Introduction to IBIS models and IBIS model making

Intel Corporation

Folsom, CA

November 3-4, 2003

Arpad Muranyi
Signal Integrity Engineering
Intel Corporation
arpad.muranyi@intel.com

intel®

CPD

Outline

- Introduction
- Transistor vs. behavioral modeling
- I-V curves
- Ramps and V-t curves
- How to obtain C_comp
- Package, PCB, connector modeling
- SPICE simulation setup for IBIS models
- Advanced I/O buffer types
- On-die termination modeling
- Differential buffer modeling
- Pre/De-emphasis buffer modeling
- Programmable buffer modeling

What is IBIS?

I I/O
B buffer
I information
S specification

IBIS, common name for any of about 30 species of long-legged, long-necked wading birds.

- **IBIS is a standard for describing the analog behavior of the buffers of digital devices using plain ASCII text formatted data**
  IBIS files are really *not* models, they just contain the data that will be used by the behavioral models and algorithms inside the simulators
- **Started in the early 90's to promote tool independent I/O models for system level Signal Integrity work**
- **It is now the ANSI/EIA-656 and IEC 62014-1 standard**

http://www.eigroup.org/ibis/ibis.htm

The origins of IBIS

- In 1991/2 PCI bus Signal Integrity simulations were ramping up at Intel, but no one had a PCI buffer designed yet
- SPICE models were very difficult to get in general
- We needed an easy way to do “what if” analysis to come up with the buffer specification
- Developed a behavioral buffer model in HSPICE to be able to simulate any buffer characteristics
- The behavioral model was so successful that Intel decided to supply these to customers
- However, not all customers used HSPICE, so a tool independent model format was desirable
- Several EDA tool vendors showed interest in a common modeling format
- The IBIS Open Forum was formed and the first IBIS specification was written

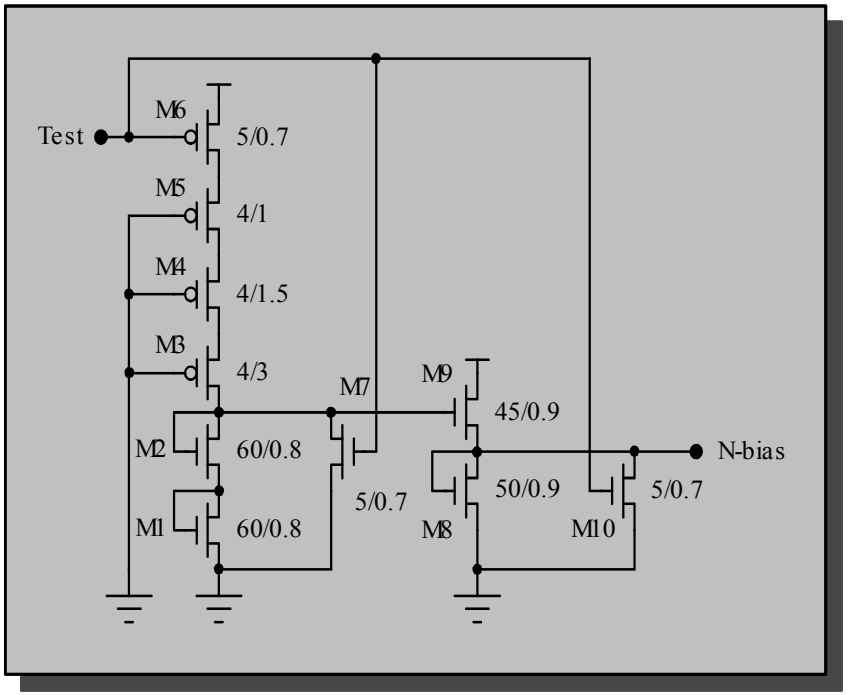
Behavioral simulation background

- **Quad Design Technology ➔ Innoveda ➔ Mentor**
  TLC, XTK - over 20 years old
- **Integrity Engineering ➔ Mentor**
  - SimNet - late 80's early 90's
- **Cadence**
  DFS (design for signoise) - early 90's
- **Siemens ➔ Incases**
  Signal Integrity Workbench - early 90's
- **Intusoft**
  IsSpice SpiceMod - early 90's
- **Hyperlinx ➔ Mentor**
  Line Sim - early 90's
- **Interconnectix ➔ Mentor**
  Interconnect Synthesis - early/mid 90's
- **VHDL-AMS, Verilog-AMS**
  Emerging new (IEEE) standards – late 90's

Behavioral vs. structural modeling

- **Behavioral models usually describe devices using electrical parameters as seen at their terminals**
  - black box approach
  - no internal information is available or needed
  - the S-parameter data of passive circuits (T-lines) is a behavioral model
  - IBIS models
- **Structural models use lots of internal detail**
  - all internal details of a buffer must be available and are needed
    - device structure, geometry and properties of material information
    - exact circuit schematic (SPICE)
  - T-line dimensions, PCB stack-up and properties of materials are used by electro-magnetic field solvers
- **Equivalent circuit representation**
  - RLGC data and circuit representation of a T-line
  - controlled source plus RC representation of semiconductors

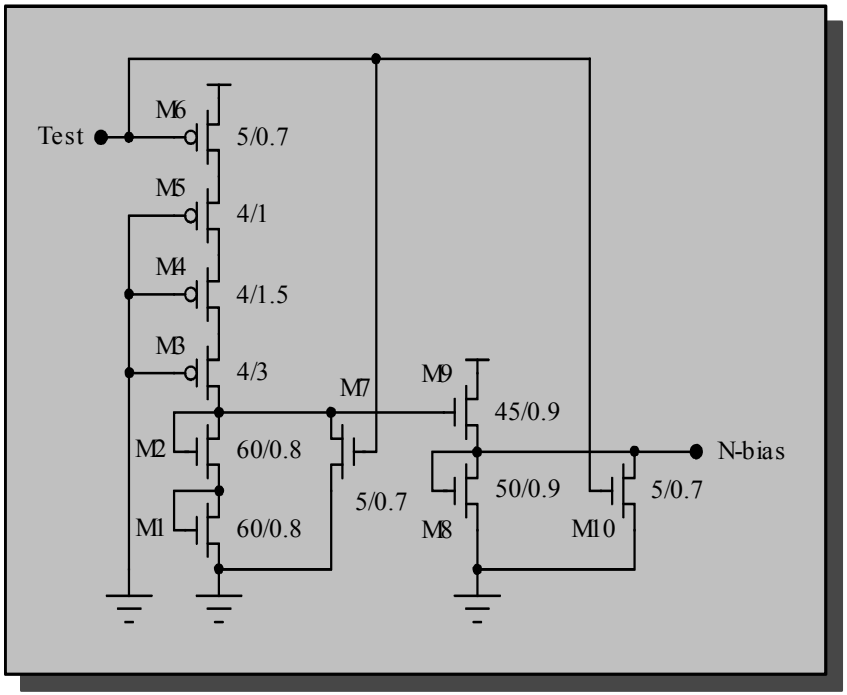
SPICE model




```
*********************************************
.MODEL NMOS NMOS
+ (LEVEL=3          UO=400.0          VTO=1.00
+  TPG=1            TOX=15E-9         NSUB=1.00E17
+  VMAX=200.0E3     RSH=50            XJ=100.0E-9
+  LD=120.0E-9      DELTA=20.0E-3     THETA=0.10
+  ETA=10.0E-3      KAPPA=20.0E-18    PB=0.40
+  CGSO=2.00E-10    CGDO=2.00E-10     CJ=0.30E-3
+  CJSW=0.20E-9     MJ=350.0E-3       MJSW=200.0E-3)
*********************************************
```

uses full schematics of the buffer
plus
manufacturing process description

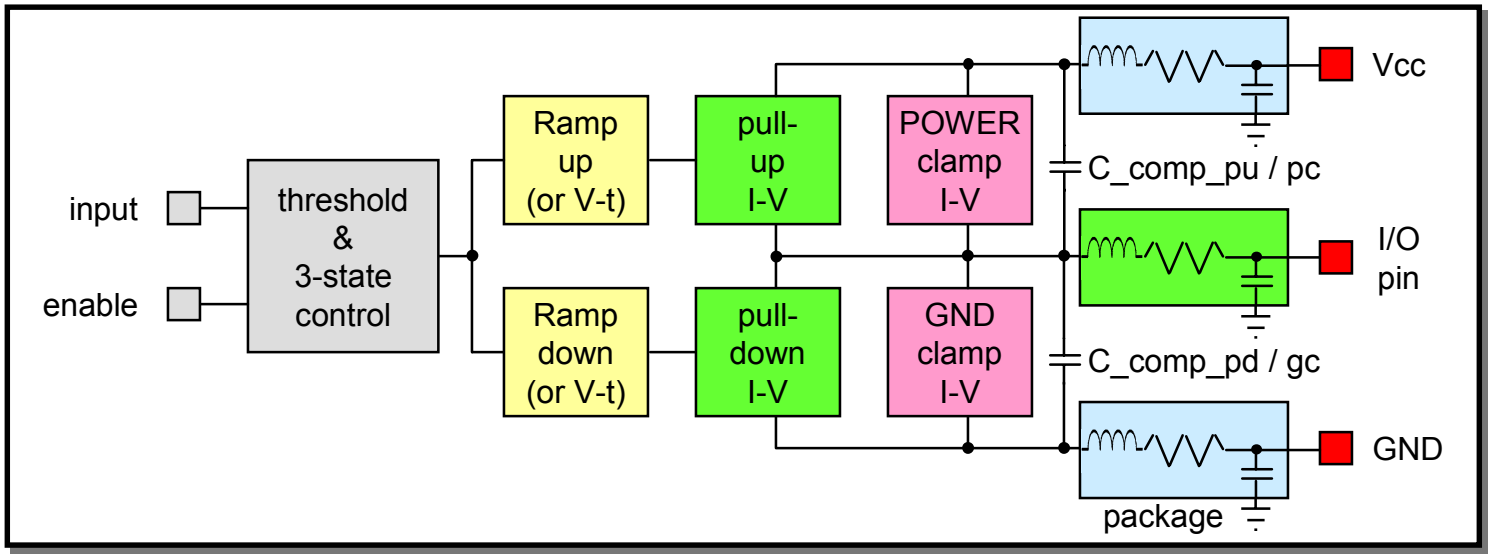
Table driven transistor model




```
*****************************************
.MODEL NMOS NMOS ...
   Vgs      Ids            ...      Cgs
  -1.000   -0.2178         ...      1.230e-12
  -0.800   -6.498e-02      ...      1.250e-12
  -0.600   -2.913e-02      ...      1.270e-12
  -0.400   -1.542e-02      ...      1.290e-12
  -0.200   -7.874e-03      ...      1.310e-12
   0.000   -4.733e-05      ...      1.330e-12
   0.200    7.643e-03      ...      1.350e-12
   0.400    1.448e-02      ...      1.370e-12
   0.600    2.062e-02      ...      1.390e-12
   0.800    2.620e-02      ...      1.410e-12
   1.000    3.086e-02      ...      1.430e-12
*****************************************
```

uses full schematics of the buffer
plus
I-V and C-V curves (tables) for each transistor

IBIS model



Block diagram of CMOS buffer

A basic IBIS model consists of:

four I-V curves: - pullup & POWER clamp
- pulldown & GND clamp

two ramps: - dV/dt_rise
- dV/dt_fall

die capacitance: - C_comp

packaging: - RLC values

for each buffer on a chip

Model characteristics

- **SPICE (transistor level) model**
  - voltage/current/capacitance relationships of device nodes are calculated with detailed equations using device geometry, and properties of materials
  - measured data is curve fitted for the equations
  - simulates very slowly, because voltage/current relationships are calculated from lower level data
  - voltages/currents are calculated for each circuit element of the buffer netlist
  - best for circuit designers
  - too slow for system level interconnect design
  - reveals process and circuit intellectual property

Model characteristics (cont'd)

- **Table driven transistor model (EIAJ)**
  - voltage/current/capacitance relationships of device nodes are based on I/C/V lookup tables
  - data tables are generated from full SPICE model simulations or die level measurements
  - data could be curve fit to equations in the future to improve efficiency and flexibility
  - simulates faster, because nodal voltage/current relationships are given directly, not because it uses data tables
  - voltages/currents are calculated for each circuit element of the buffer netlist
  - more suitable for system level design because it is faster
  - hides process, but still reveals circuit intellectual property

Model characteristics (cont'd)

- **IBIS (or behavioral) model**
  - current/voltage/time relationships of entire buffer (or building block) are based on lookup tables (I-V and V-t curves)
  - data tables are generated from full SPICE model simulations or external measurements
  - data could be curve fit to equations in the future to improve efficiency and flexibility
  - simulates fastest, because voltage/current/time relationships are given for the external nodes of the entire buffer (or building block), not because it uses data tables
  - no circuit detail involved
  - useless for circuit designers
  - ideal for system level interconnect design
  - hides both process and circuit intellectual property

All models are "behavioral" on some level!

- **Behavioral models are called that way because the information in them describes their *electrical behavior* directly**
  - it is not the data format (i.e. tables) that makes a model "behavioral"
- **The *electrical behavior* of SPICE models are calculated from device geometries and properties of materials**
- **The real difference is in the level of abstraction**
  - semiconductor device physics simulation tools go as low as describing the electron behavior in the crystal structure of the material (PISCES)
  - SPICE models use device geometries and properties of materials
  - behavioral models use direct description of terminal node voltages and currents
- **The accuracy of a model is not directly related to what kind of model it is**
  - behavioral models can have good accuracy if they describe the behavior of interest correctly
  - structural models can also be bad if the equations in them are incorrect

IBIS model as buffer specification

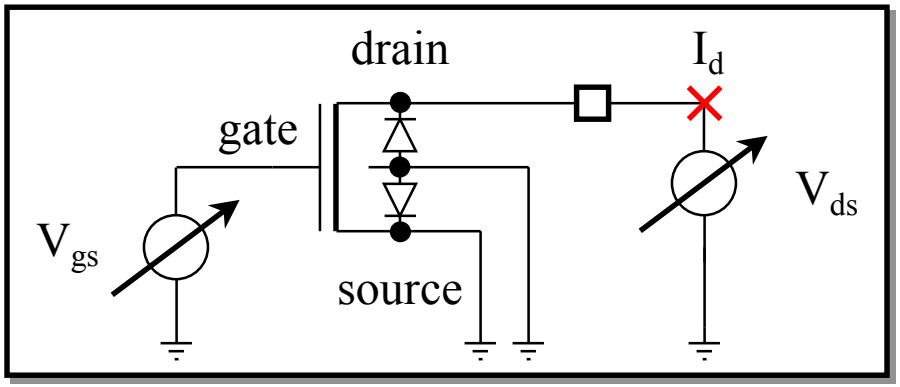
- **Signal Integrity group defines the correct buffer characteristics**
  - behavioral models lend themselves to easy “what if” analysis to find a system level solution space
  - results can be communicated to design engineering through the I-V and V-t curves and other parameters
- **Circuit designers can use these requirements as a specification for the buffer to be designed**
  - matching SPICE simulated I-V and V-t curves to IBIS model
- **Intel has used this methodology since the early 90’s on several bus interfaces**
  - PCI specification
  - AGP specification
  - 100 MHz SDRAM (PC100) specification
  - numerous internal projects

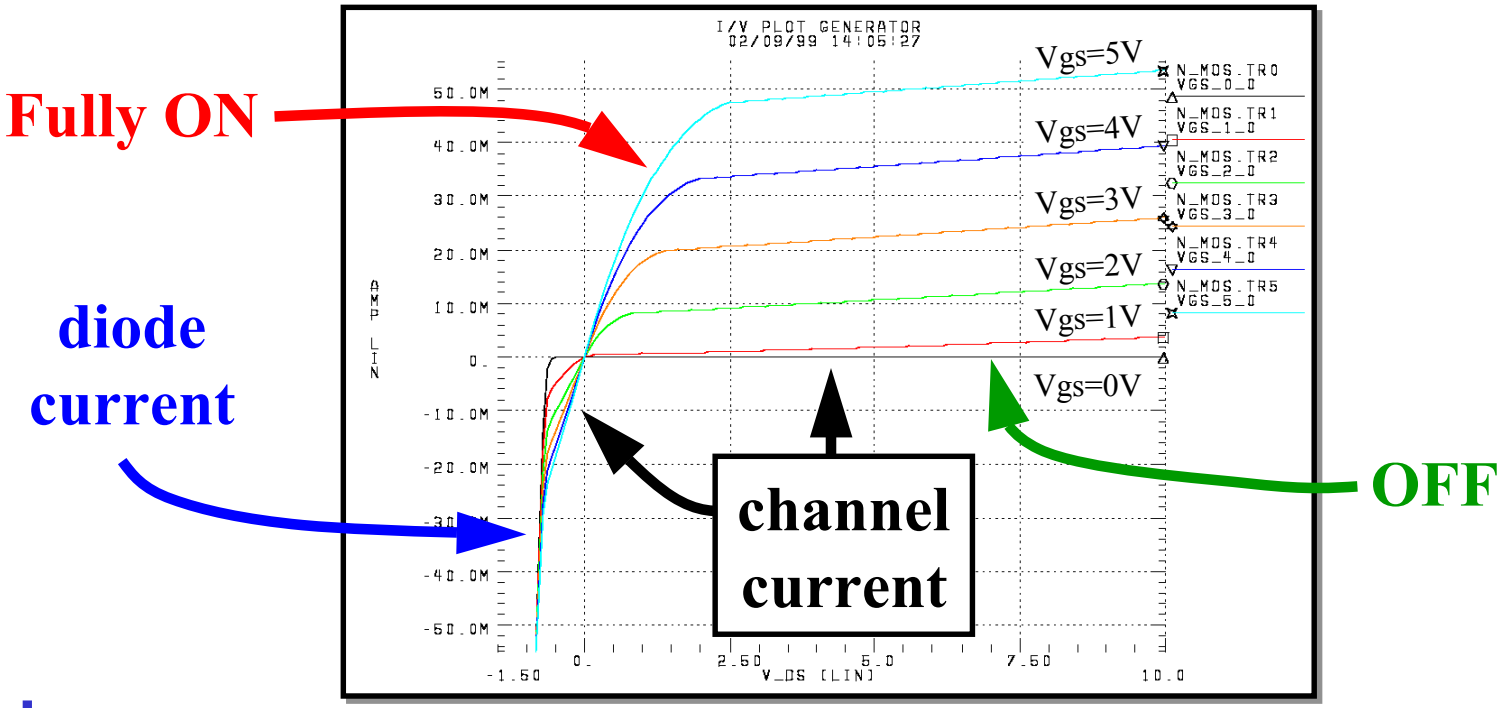
Basic structure of an IBIS file

component

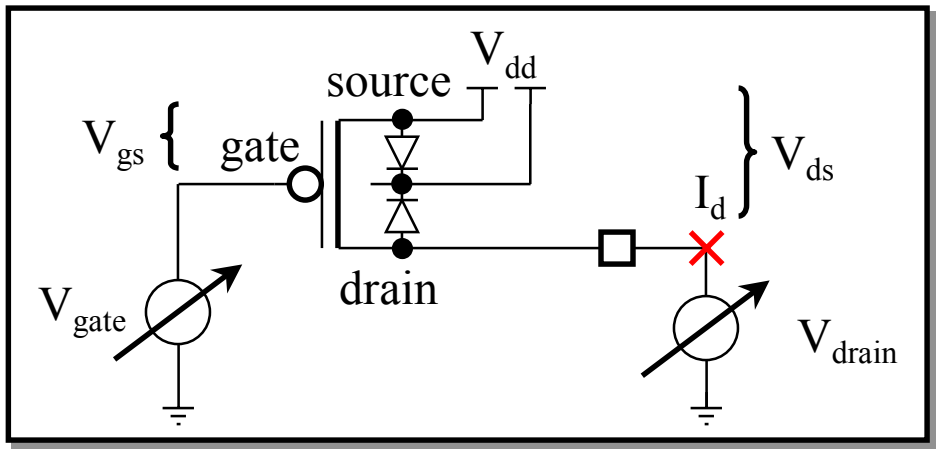
- **Header**
  - file name, date, version, source, notes, disclaimer, copyright, etc.
  - default package data (L_pkg, R_pkg, C_pkg)
  - pin list (pin name, signal name, buffer name, and optional L_pkg, R_pkg, C_pkg)
  - advanced items (differential pin associations, buffer selector, etc.)
- **Model data**
  - all models found on the chip are included in this section
  - each flavor of a programmable buffer is described under a separate [Model] section
- **One IBIS file can describe several components (chips)**
  - a new [Component] keyword can be used for each part within the same .IBS file
  - models from multiple vendors can be combined to form a "system model"

I-V curves of N-channel pulldowns





I-V curves of P-channel pullups

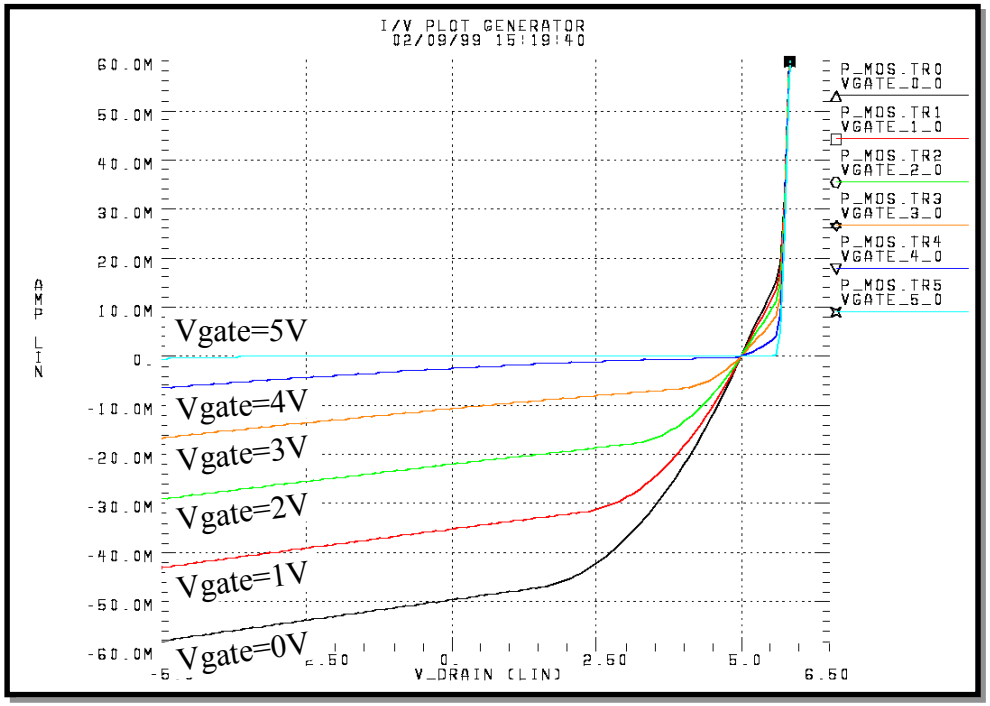


$V_{gs}=V_{dd}-V_{gate}$

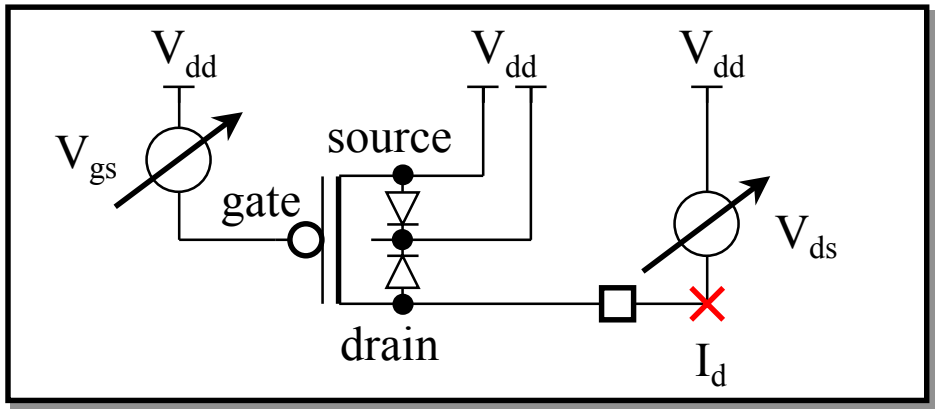
$V_{ds}=V_{dd}-V_{drain}$

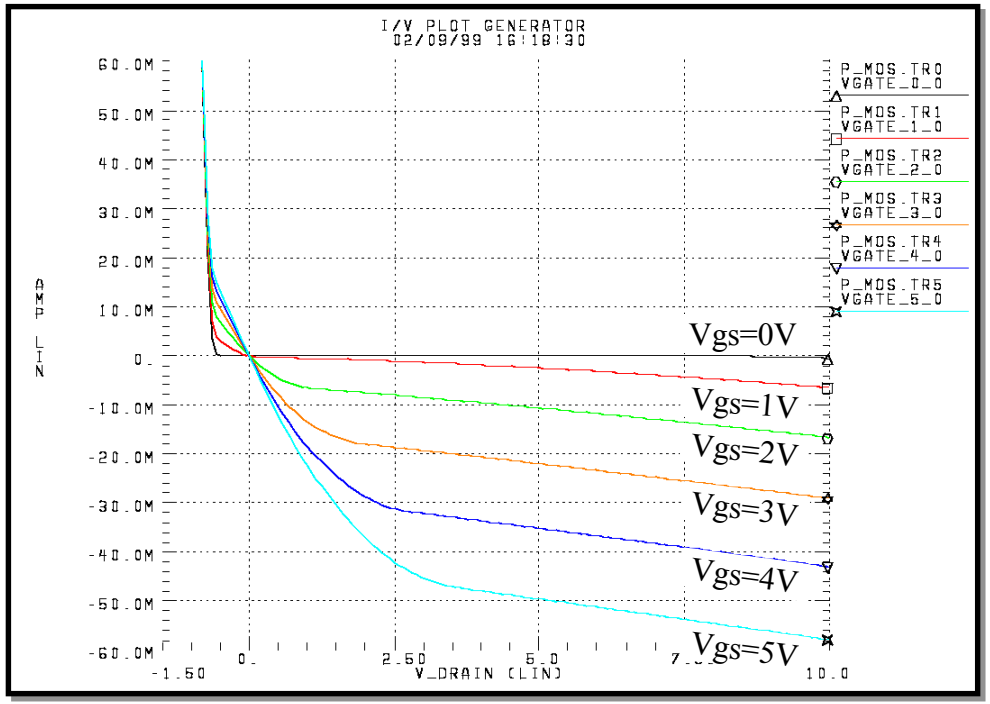
Ground relative measurements don't make sense for pullup structures because V_{gs} and V_{ds} are not related to the GND potential!

(V_{dd} varies with minimum, typical, and maximum modeling conditions, as well as in GND bounce and V_{dd} droop simulations).



V_{dd} relative pullup I-V curves

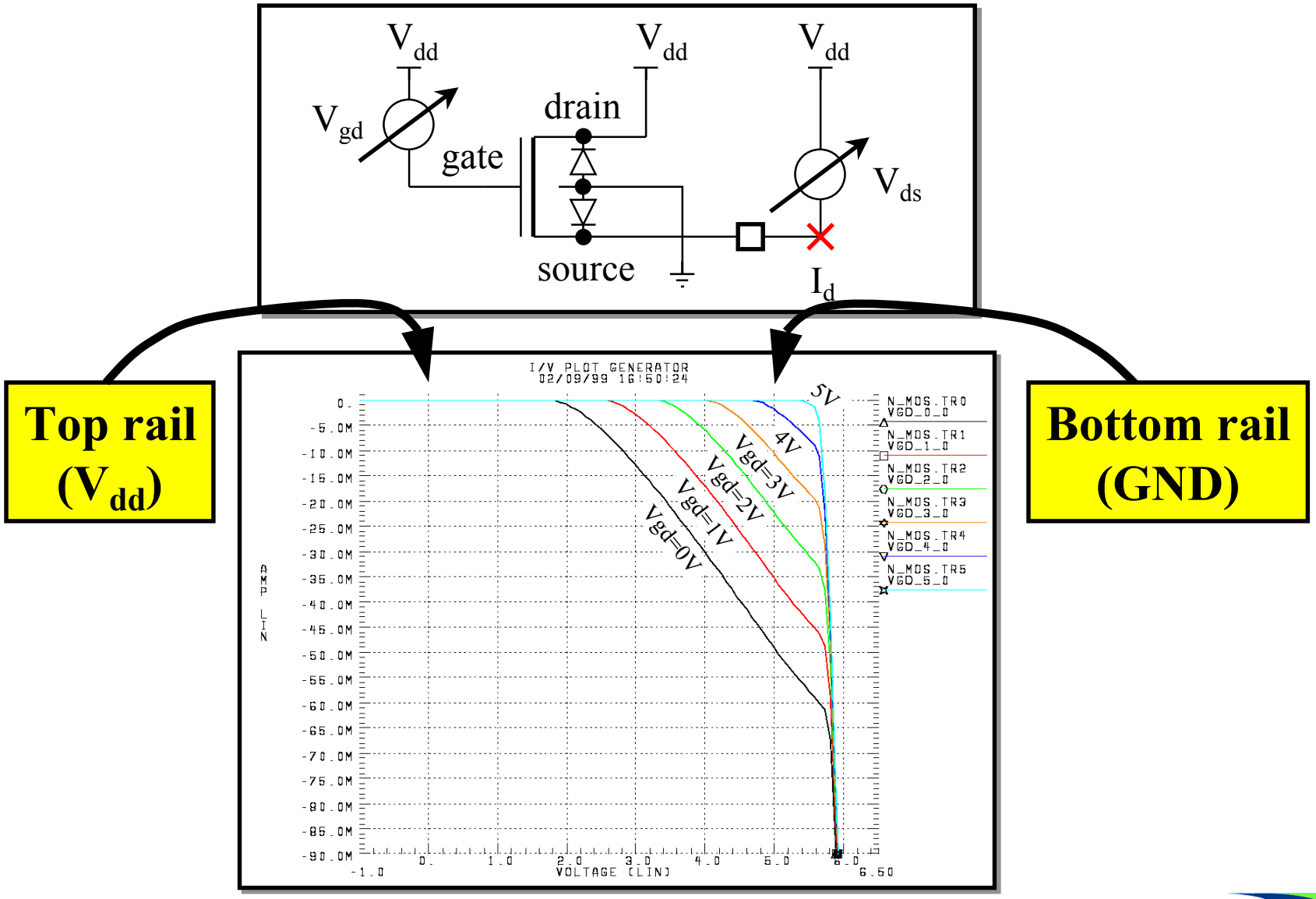




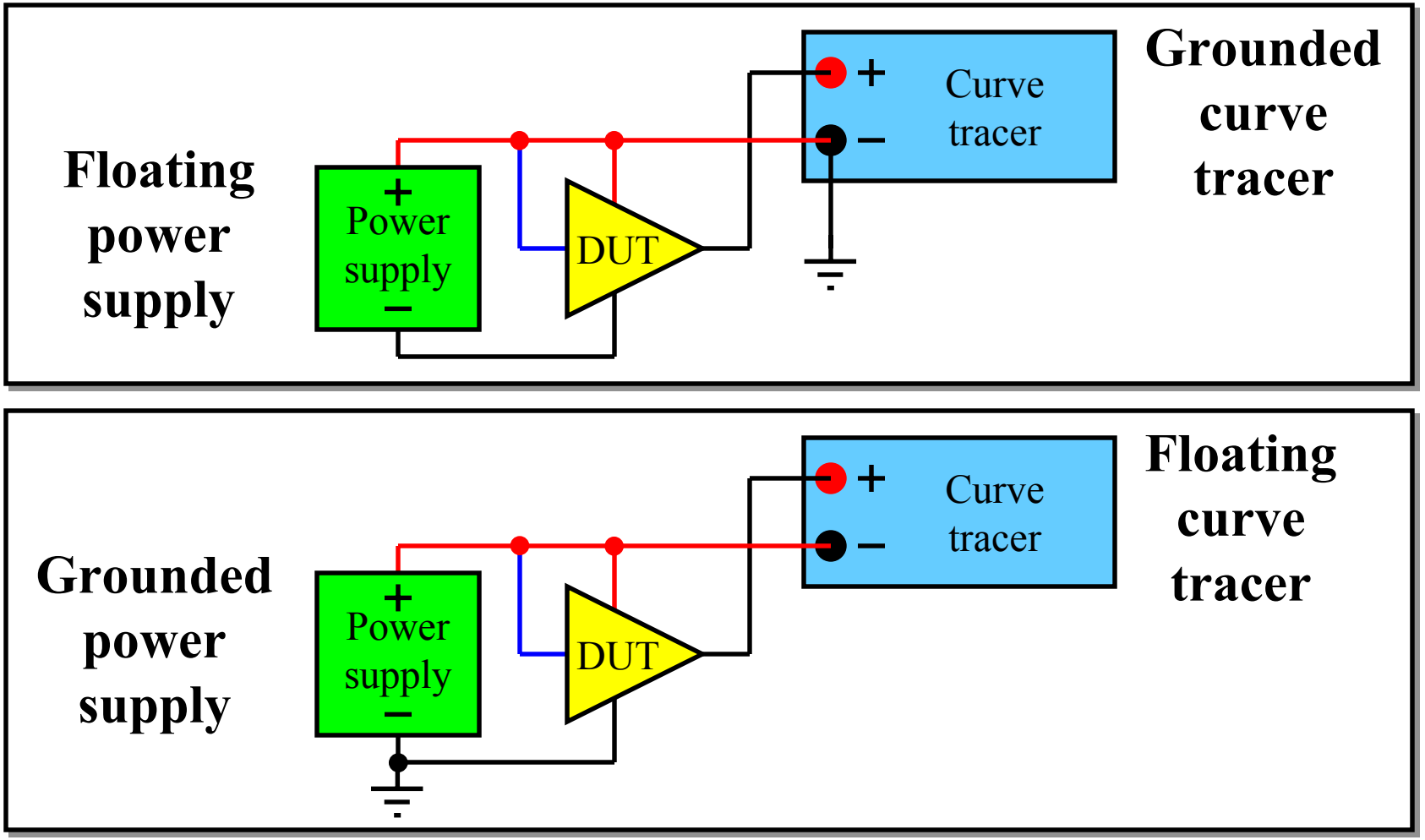
V_{dd} relative characterization of pullup structures are perfectly symmetric with the ground relative characterization of pulldown structures

Measured (or DUT) voltage conditions do not have to be recalculated for each supply voltage.

N-channel pullup I-V curves

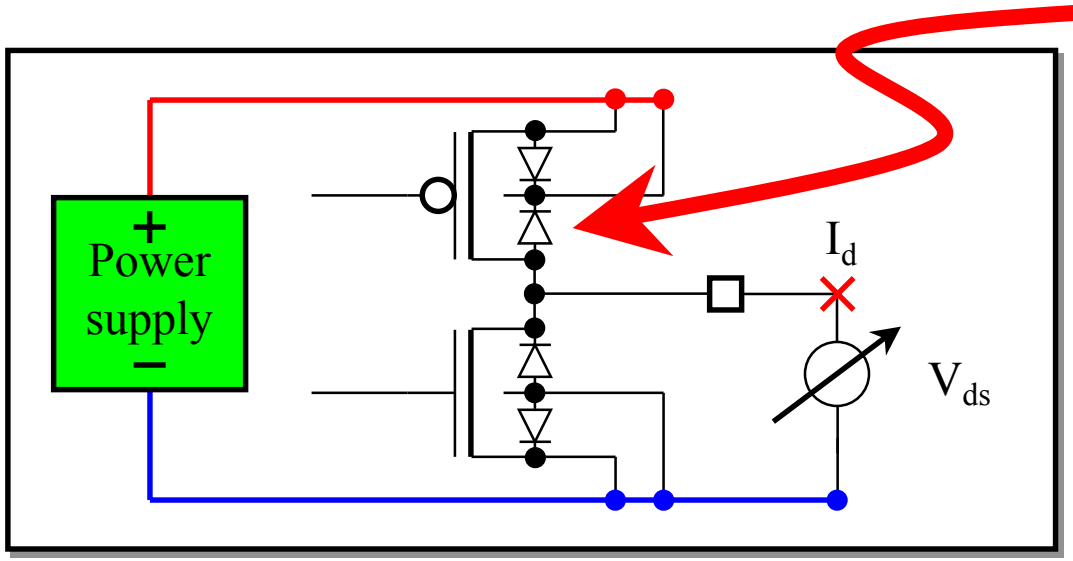


V_{dd} relative lab measurements



Don’t do this if supply and curve tracer are both grounded!

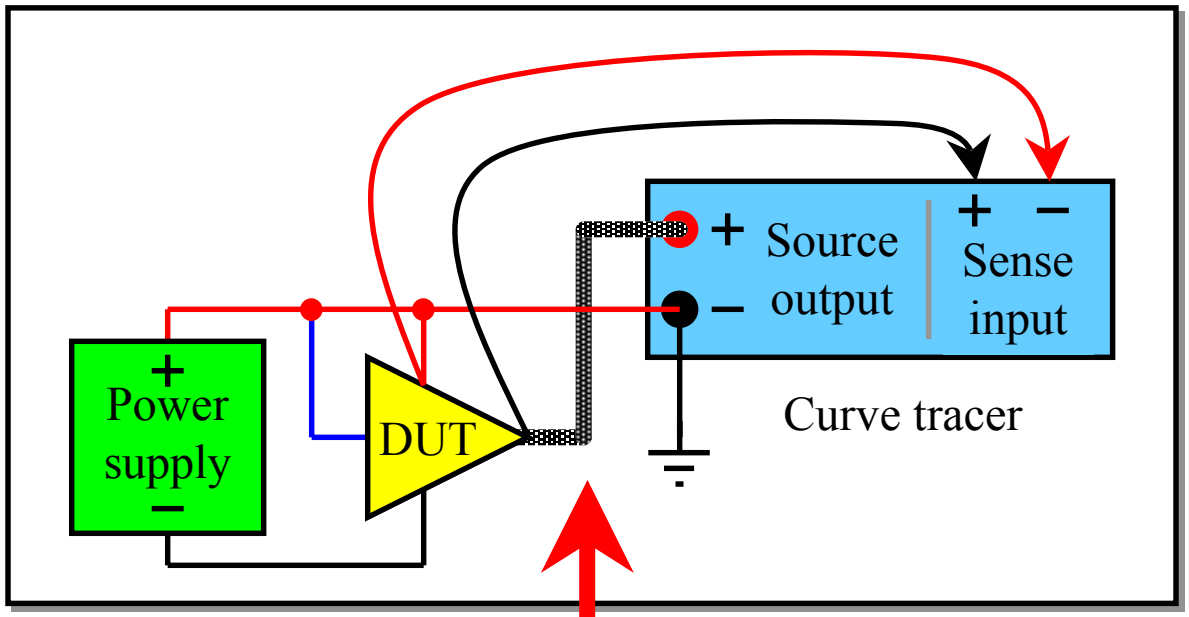
Use push and pull power supplies!



One of the parasitic diodes of the P-channel FET turns ON when V_{ds} goes beyond the supply voltage!

The power supply must be able to sink and source currents in order to keep the supply voltage at the desired level.

Precision IV curve tracing



High resistance and high current path causes large voltage drop across test fixture / wiring

What do we get in the I-V curves?

Digital buffers can only be measured in two modes (if not one)

- **Receive or 3-state mode**
  - **allows us to measure the currents of**
    parasitic diodes (for MOSFETS)
    ESD protection circuits
    pullup or pulldown “resistors”
    static overshoot protection clamps
    integrated static terminators
- **Drive mode (driving high or low)**
  - **allows us to measure the currents of all of the above, *plus***
    channel currents for pulldown and/or pullup structures
    dynamic clamps
    dynamic bus hold circuits
    integrated active terminators

Both modes are needed for SI simulations

Drive/receive mode transitioning

- **Phase out "receiver model" as the "driver model" is phased in, or vice versa**

  It may be difficult to come up with an algorithm that can cross fade the two models so that the static currents remain constant during the transitioning process.

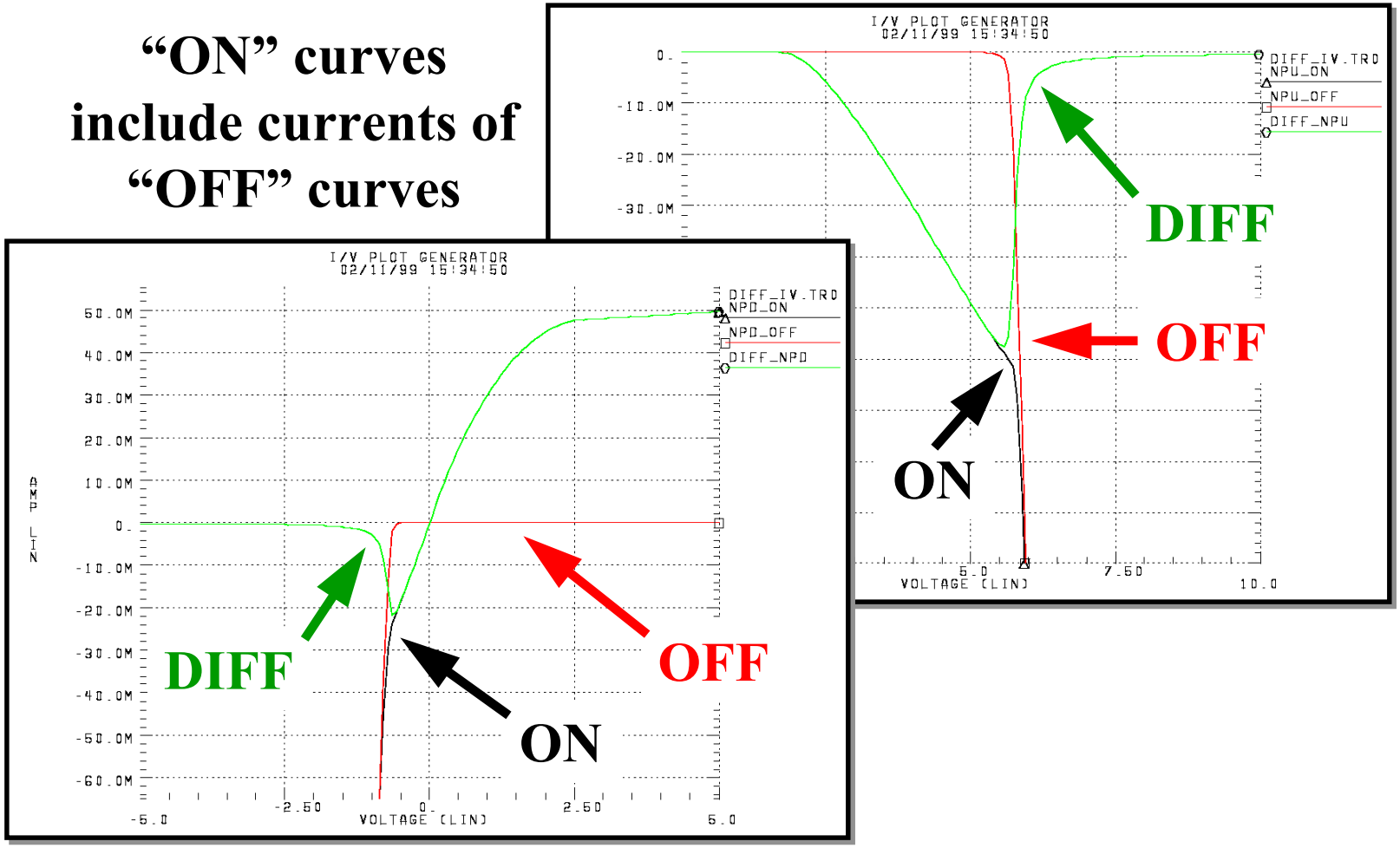
- **Keep "receiver model" constantly in the circuit and phase in/out the *difference* between the driver and receiver models**

  Note that a drive mode model includes the static currents of those circuit or device elements that make up a receive mode model because these are never turned off. These currents would be doubled if the drive mode model would simply be added to a receive mode model.

The second approach is used in IBIS

Difference I-V curve examples

“ON” curves include currents of “OFF” curves



The four I-V curves of an IBIS model

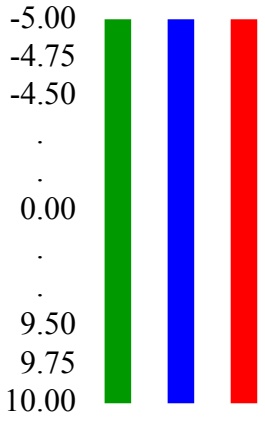
- **[Pulldown], referenced to [Pulldown Reference]**
  - contains the difference of drive and receive (3-state) mode I-V curves for driver driving low
  - the origin of the curve is usually at 0 V (GND) for normal CMOS buffers, but could be at -12 V for RS232 drivers, for example
- **[Pullup], referenced to [Pullup Reference]**
  - contains the difference of drive and receive (3-state) mode I-V curves for driver driving high
  - the origin of the curve is usually at the supply voltage (Vcc or Vdd)
- **[GND Clamp], referenced to [GND Clamp Reference]**
  - contains the receive (3-state) mode I-V curves
  - the origin of the curve is usually at 0 V (GND) for normal CMOS buffers, but could be at -12 V for RS232 drivers, for example
- **[POWER Clamp], referenced to [POWER Clamp Reference]**
  - contains the receive (3-state) mode I-V curves
  - the origin of the curve is usually at the supply voltage (V_{cc} or V_{dd}), but
  - for 5 V safe 3.3 V buffers, for example, this would be referenced to 5 V while the pullup is referenced to 3.3 V

I-V curve ranges

- **The general rule is $-V_{dd}$ to $2*V_{dd}$**
  - for a 5 V buffer this means -5 V to +10 V
- **Why is this necessary?**
  - the theoretical maximum overshoot due to a full reflection is twice the signal swing
  - since tools may not extrapolate curves the same way, it is better to define it
- **Exception: clamp curves**
  - [GND Clamp] is required to have $-V_{dd}$ to V_{dd} (-5 V to +5 V)
  - [POWER Clamp] is required to have V_{dd} to $2*V_{dd}$ (+5 V to +10 V)
- **Reason: prevent double counting**
  - the two clamp curve measurements contain the same information
  - the only difference is that one of them is measured with respect to [GND Clamp Reference] (GND relative) and the other one with respect to [POWER Clamp Reference] (V_{cc} or V_{dd} relative)
- **Note that these ranges are defined as minimum required ranges in the IBIS specification, but providing data over a wider range is not prohibited**

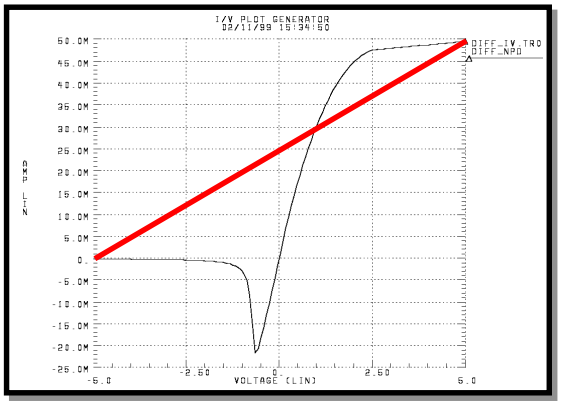
More on I-V curve ranges

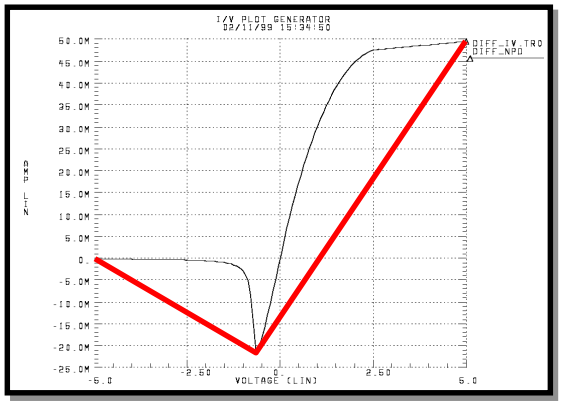
- **Mixed voltage cases**
  - a 5 V safe 3.3 volt buffer should use a -5 V to +10 V range because there is 5 V signaling
  - a 3.3 V GTL+ driver using a V_{ref} of 1.5 V should have a range of -1.5 V to 3.0 V because the signaling is 1.5 V (at most)
- **What about typ., min., and max. supply voltages?**
  - the voltage range should be based on the typical value only because there is one voltage column for each case, and a numerical value is required in the first and last points

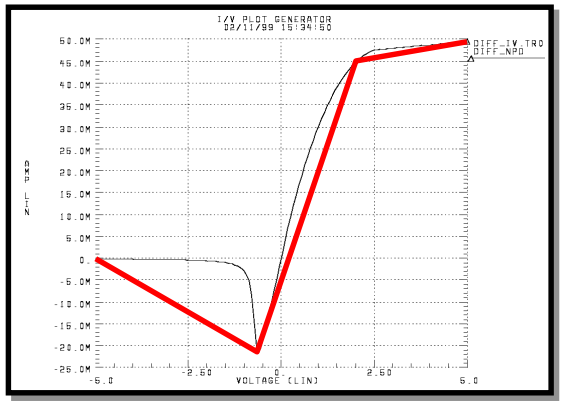


Best 100 points reduction

- **Higher resolution yields more accurate models**
  - it is desirable to have a point at least at every 100 mV, or less
- **The range of a 3.3 V device is -3.3 to 6.6 V, a span of 9.9 V**
  - at 100 mV spacing this will result in 100 points, exactly
- **For 5 V devices or higher resolution a "best points" algorithm must be used to eliminate points from the data where the curve is (almost) straight**
- **V-t curves may even need more points depending on the edge rate and time step used**







Temperature and power supply conditions for I-V and V-t curves

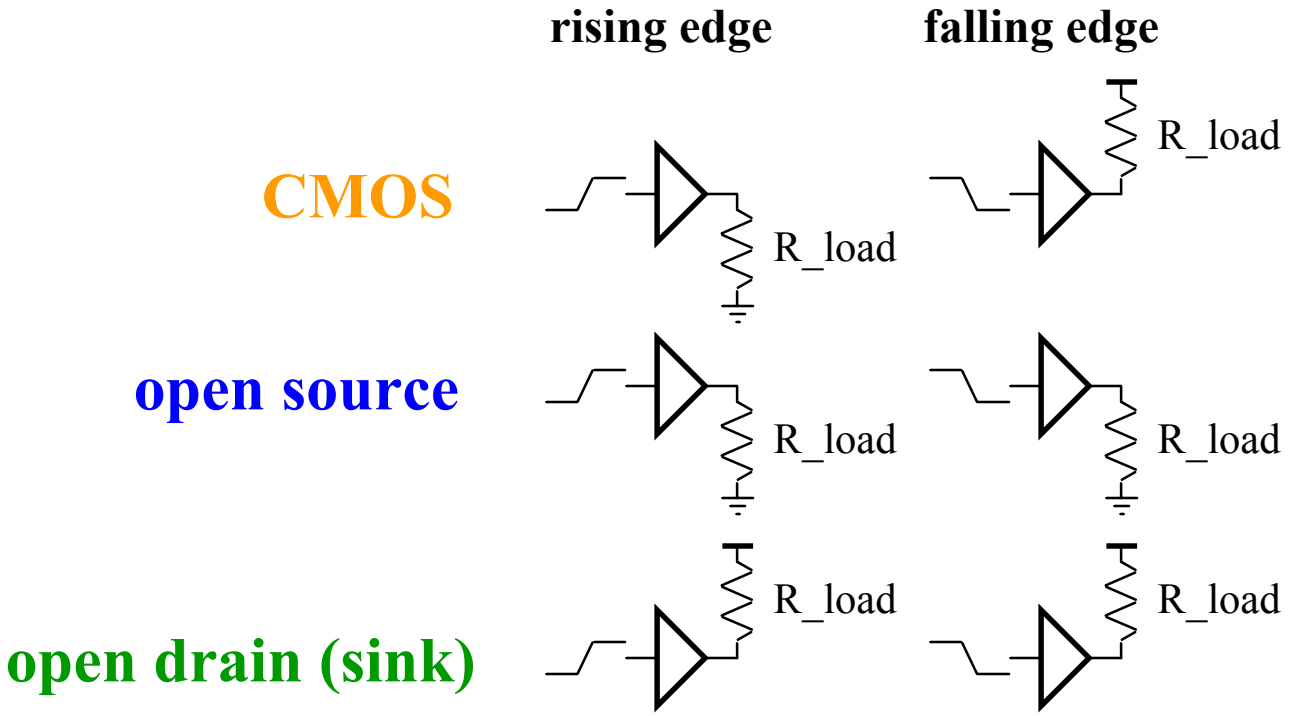
- **IBIS supports three conditions for buffer models**
  - typical values (required)
  - minimum values (optional) - not the same as worst/slow case in general
  - maximum values (optional) - not the same as best/fast case in general
- **For CMOS I-V curves**
  - minimum I-V curve conditions (worst case) require high temperature and low supply voltage
  - maximum I-V curve conditions (best case) require low temperature and high supply voltage
- **For bipolar I-V curves**
  - minimum I-V curve conditions (worst case) require low temperature and low supply voltage
  - maximum I-V curve conditions (best case) require high temperature and high supply voltage

Ramp and V-t curve measurements

- **The [Ramp] and [*** Waveform] keywords describe the transient characteristics of a buffer**
  - these keywords contain information on how fast the pullup and pulldown transistors turn on/off with respect to time
- **The effects of the package must be eliminated from these measurements**
  - remove package from SPICE circuit
  - obtain measurements from the die-pad (possibly without the bond wire connected)
  - reverse engineer a package-less waveform from packaged waveform measurement through numerical methods
- **The effects of die capacitance (C_comp) are included in the shape of the V-t curves**
  - parasitic capacitance cannot be separated from the circuit elements
  - simulation tool should have a special algorithm to avoid double counting C_comp, i.e. make it invisible from the inside out, but visible from the outside in

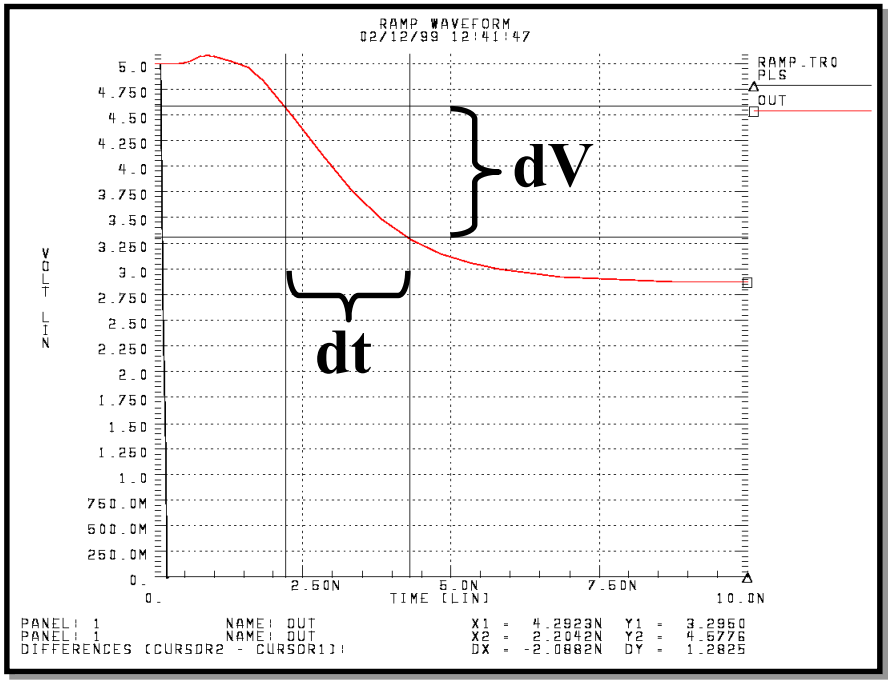
Ramp measurement setup for IBIS 1.1

Make sure R-load is connected to the appropriate supply!



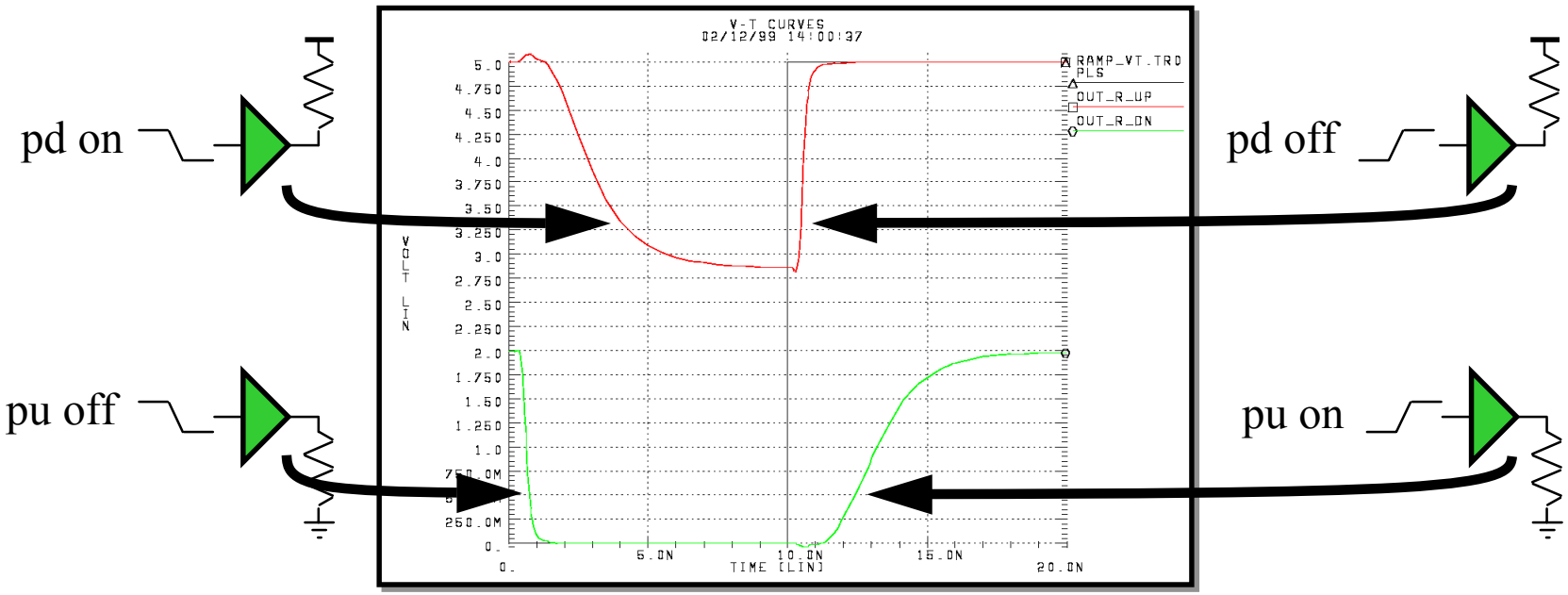
Getting dV and dt from a waveform

- **Find the 20 - 80 % points of the signal swing into Rload**
- **Read the voltage difference between these points**
  - this is your dV value
- **Read the time difference between these points**
  - this is your dt value
- **Do NOT divide out these numbers!**



The V-t curves of IBIS 2.1

- **V-t curves describe the transient characteristics of a buffer more accurately than ramps**
  - a waveform holds more detail than a straight line slope
  - slew rate controlled and multi-stage buffers may have edges which do not look like straight lines (staggered turn-on/off)
- **A minimum of four V-t curves are needed to adequately describe a CMOS buffer**



V-t curves must be time correlated!

- **Each V-t curve must have t=0 where the pulse crosses the input threshold**
  - V-t curves have to be normalized to t=0 if the buffer was not triggered at t=0 in the SPICE simulation
  - using a fast edge for the pulse helps to reduce threshold uncertainties
- **V-t curves incorporate the clock-to-out delay**
  - the accuracy of t_{co} depends on the accuracy of the SPICE model (don't trust it, most often they are not made for this)
  - you may need to tweak the leading horizontal part of the V-t curve(s) if more accurate t_{co} is known, BUT be aware that the time relationship between V-t curves must be kept right
- **The length of the V-t curve should correspond to the clock speed at which the buffer is used**
  - you may not get any output if the clock period is shorter than the length of the V-t curve (over clocking)

A note on ~0 pF loads

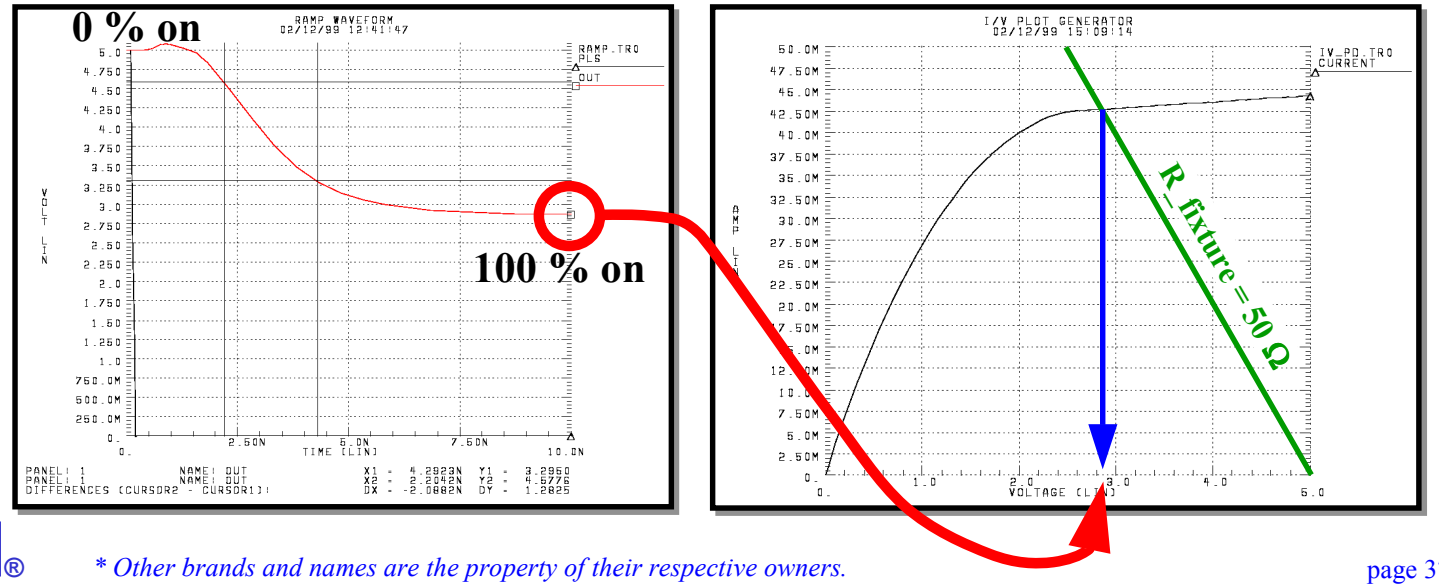
A capacitive load in buffer characterization combines the effects of speed and strength

We need a 0 pF resistive test load!

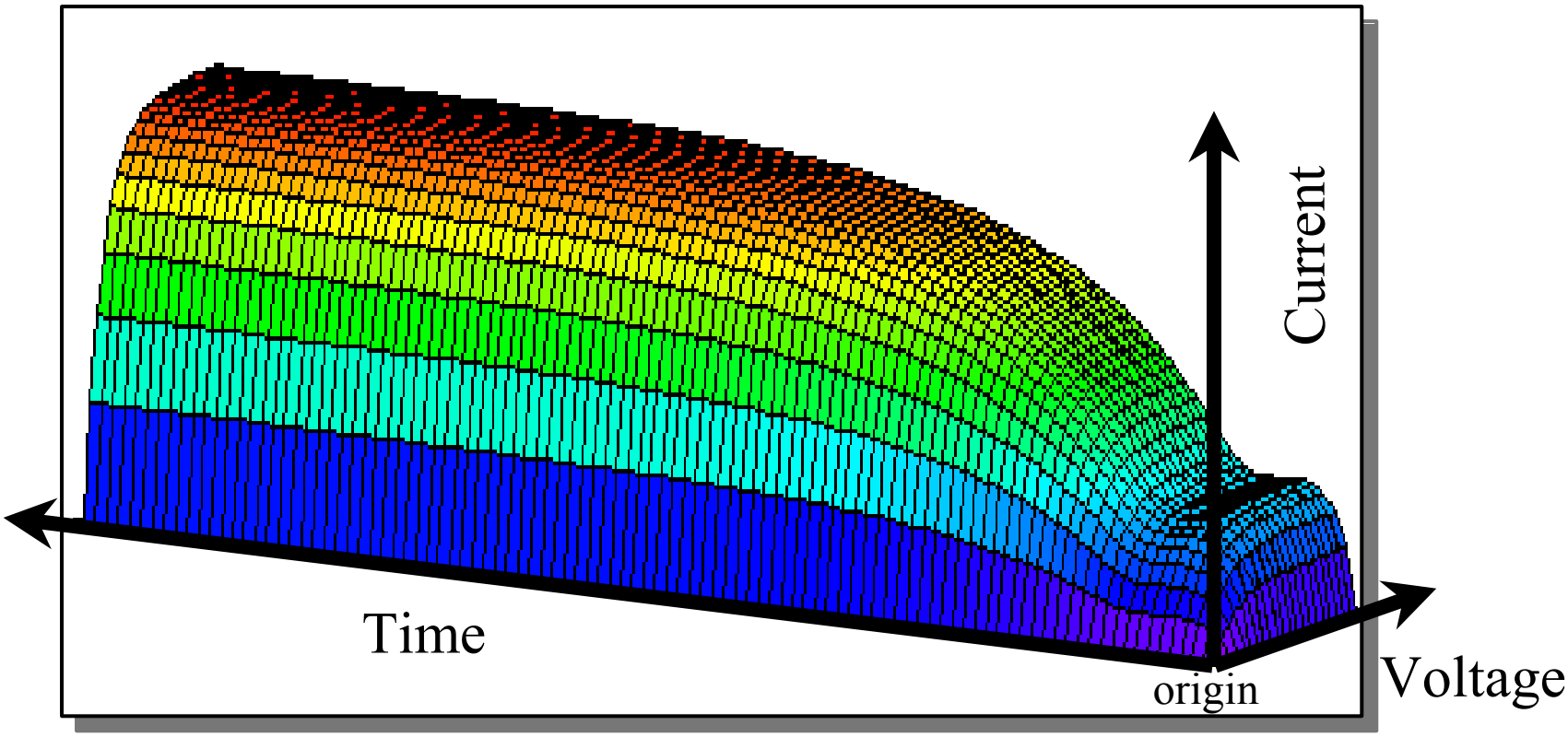
- **For fast edges transmission lines do not behave as parallel plate capacitors**
  - the driver does not see a lumped RLGC
  - a perfect (distributed) model would have infinite lumps (n=∞)
  - the RLGC values of one lump is the total RLGC divided by n
  - as n goes to infinity, the RLGC values go to zero
- **A transmission line looks like a resistor to the buffer until the reflections return**
- **Specifying the edge rates into ~0 pF load can be achieved with a transmission line or resistor**

V-t curves describe the transients

- **I-V curves provide a fully on DC characterization of the transistor**
- **V-t curves can be used to scale the I-V curves**
  - the first point of a “turn-on” V-t curve represents a driver turned OFF (0 %)
  - the last point of a “turn-on” V-t curve represents a fully ON driver (100 %)
  - anything between the end points represent a partially turned ON transistor
- **Model quality check:**
  - solving for an operating point using R_fixture across the I-V curve must yield the same voltage as the last point of the corresponding V-t curve!



3D representation of a transition



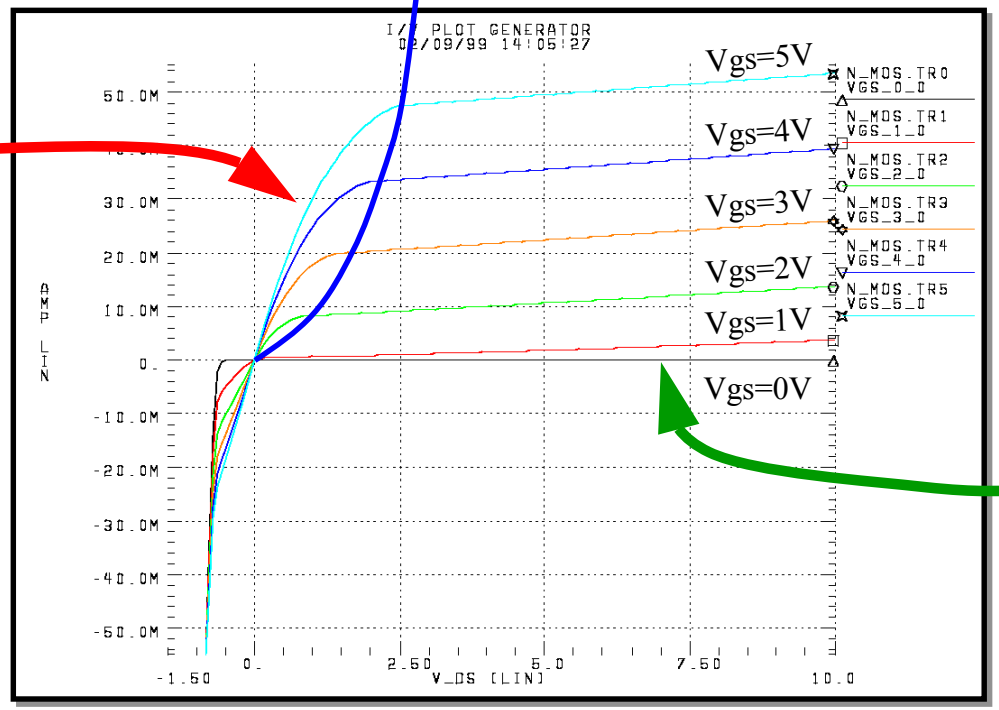
Two-stage, slew rate controlled I/O buffer

I-V curve scaling - an accuracy detail

Linear (vertical) scaling of I-V curves keeps $V_{pinchoff}$ at the same voltage. Is that a problem?

$V_{pinchoff}$ follows "square law"

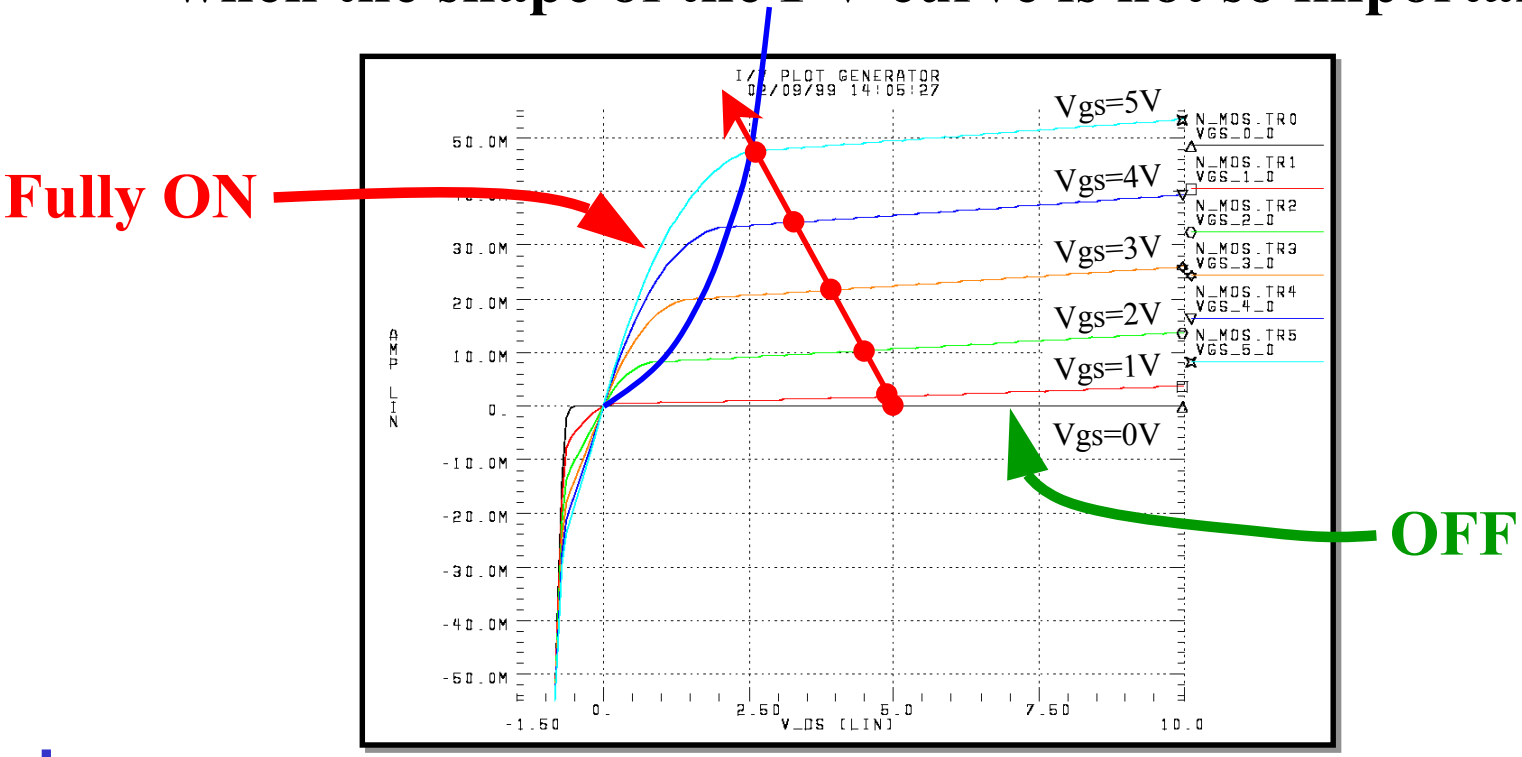
Fully ON



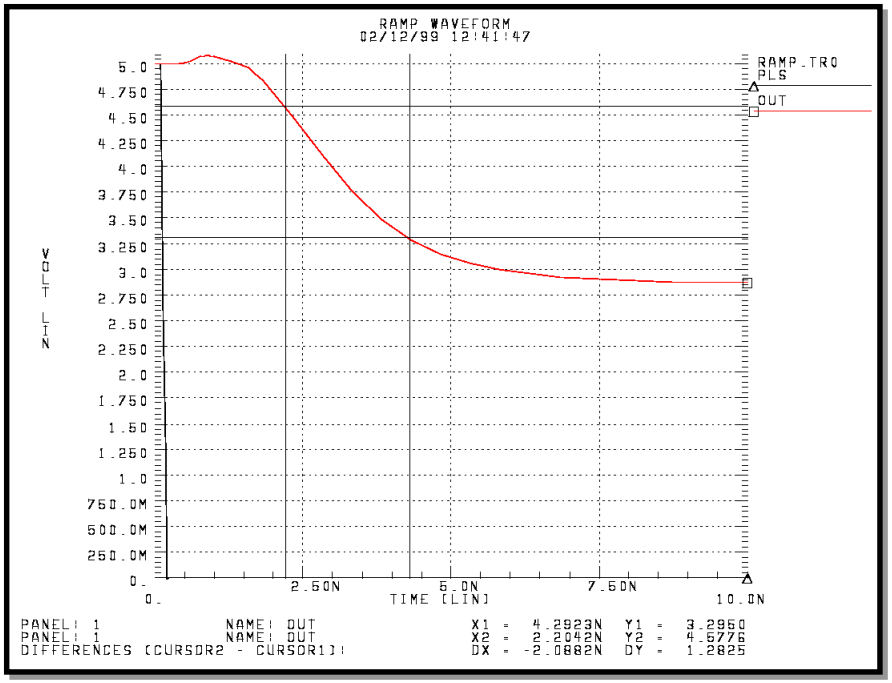
OFF

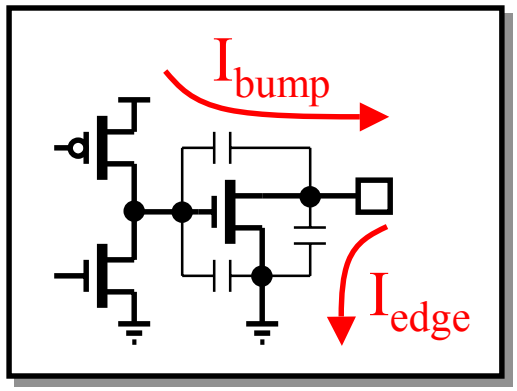
Linear scaling is usually not a problem

- Linear (vertical) scaling is most inaccurate when the buffer is barely turned on
- Walking along the load line shows that the buffer operates in the saturation region during this time when the shape of the I-V curve is not so important



Power noise accuracy problem





- **The bump preceding the falling edge is a capacitive coupling of the gate voltage to the output pad**
  - ⇨ if this V-t curve is used as a scaling coefficient to the pulldown I-V curve, the coefficient will go negative for the duration of the bump (to push up)
  - ⇨ in reality, the current for the gate to pad capacitor comes from the supply rail (for the falling edge and for an inverting transistor)
  - ⇨ if the predriver is connected to a different supply rail, this current may “bounce” another power pin

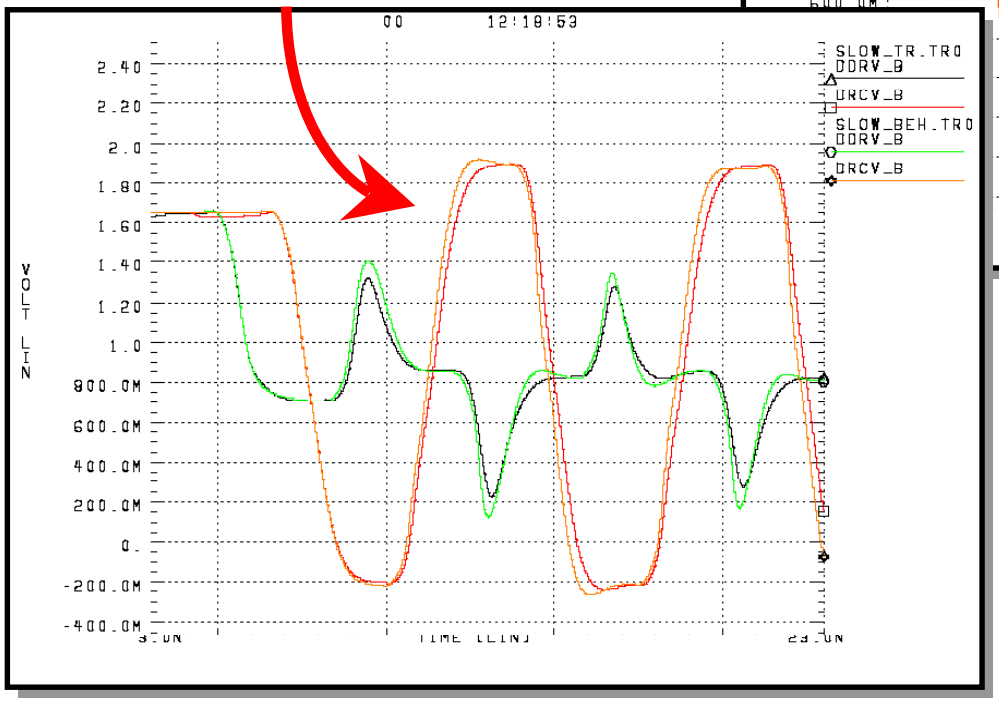
CPD

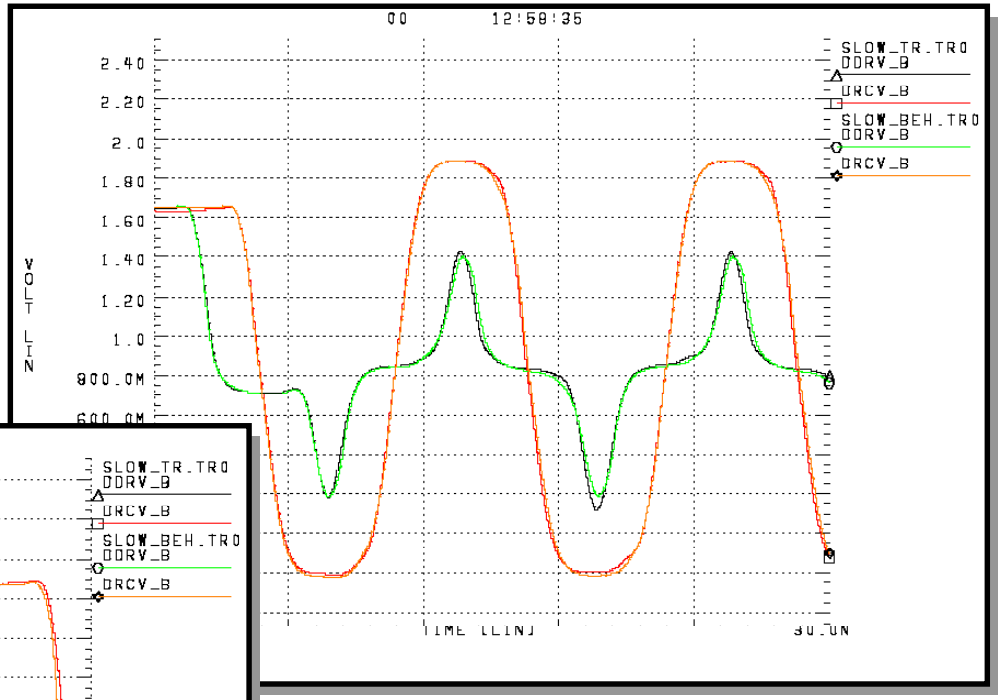
The meaning and importance of C_comp

- **C_comp is the total die capacitance as seen at the die pad**
  - parasitic capacitance of transistors and circuit elements (usually included in the process file)
  - metal capacitance connecting transistors with die pad (not necessarily included in netlist)
  - die pad capacitance (not necessarily included in netlist)
- **Do NOT include package capacitance in C_comp**
  - it is a common mistake to think of C_comp as the pin or I/O capacitance that is used in the data book which includes all capacitance as seen at the pin
- **C_comp is important even for driver models**
  - C_comp plays an important role in shaping the reflected waveforms at the driver
  - make sure C_comp matches what is in the SPICE model when correlating SPICE and IBIS models

Results of a correlation study

C_comp too large in *driver* model



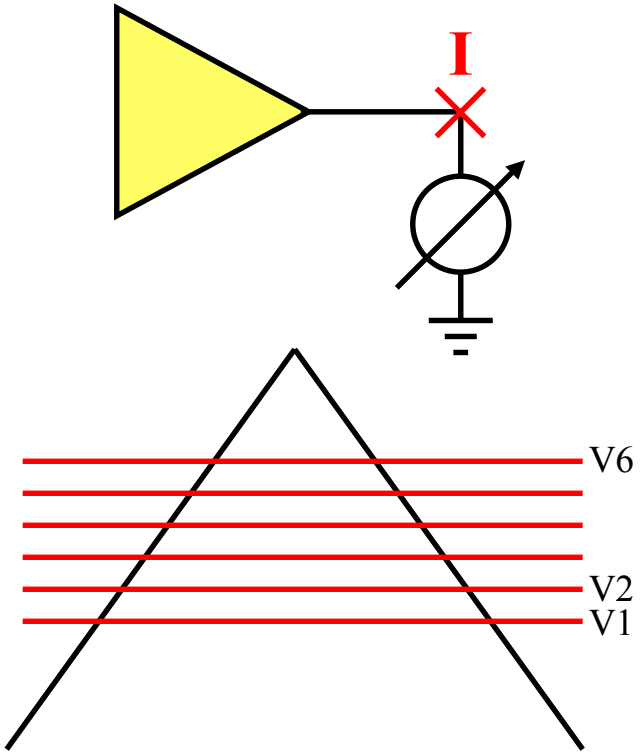


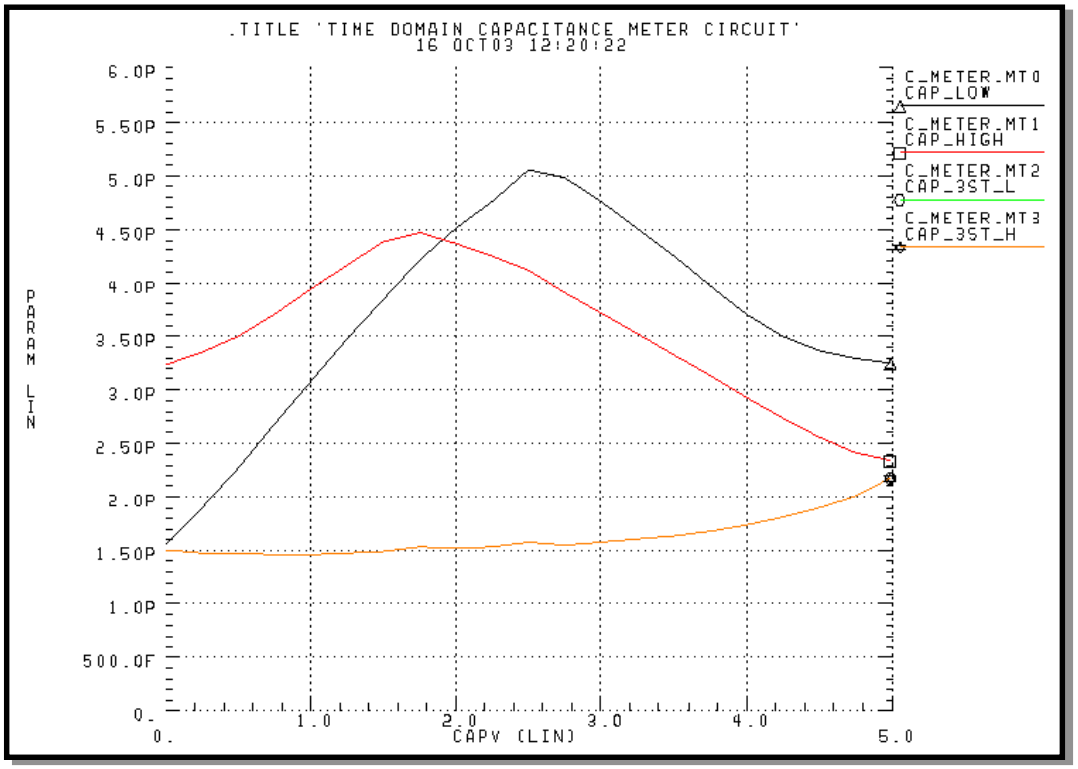
C_comp correct

Methods for measuring C_comp

- **RC time constant with a step function**
  - does not show voltage or frequency dependencies
- **Time Domain forced saw tooth voltage**
  - can provide voltage dependent capacitance curves
  - does not show frequency dependence
- **TDR technique (Time Domain)**
  - does not show voltage or frequency dependencies
- **Frequency Domain sweep with tank circuit**
  - can provide voltage dependent capacitance curves
  - does not show frequency dependence(?)
- **Frequency Domain imaginary current**
  - can provide voltage and frequency dependent capacitance curves
  - shows frequency dependence very well
- **Frequency Domain pole/zero method**
  - See BIRD 79 on IBIS web page for details

Forced saw tooth technique (TD)



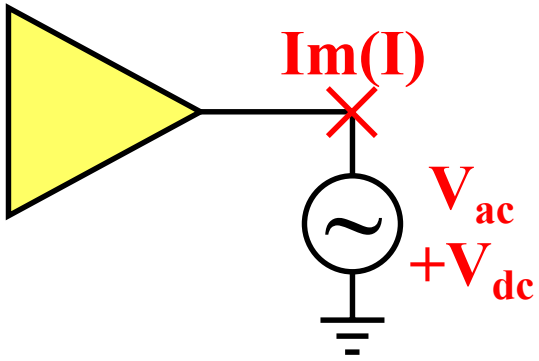


Using the same dV/dt for rise and fall, measure current on both slopes

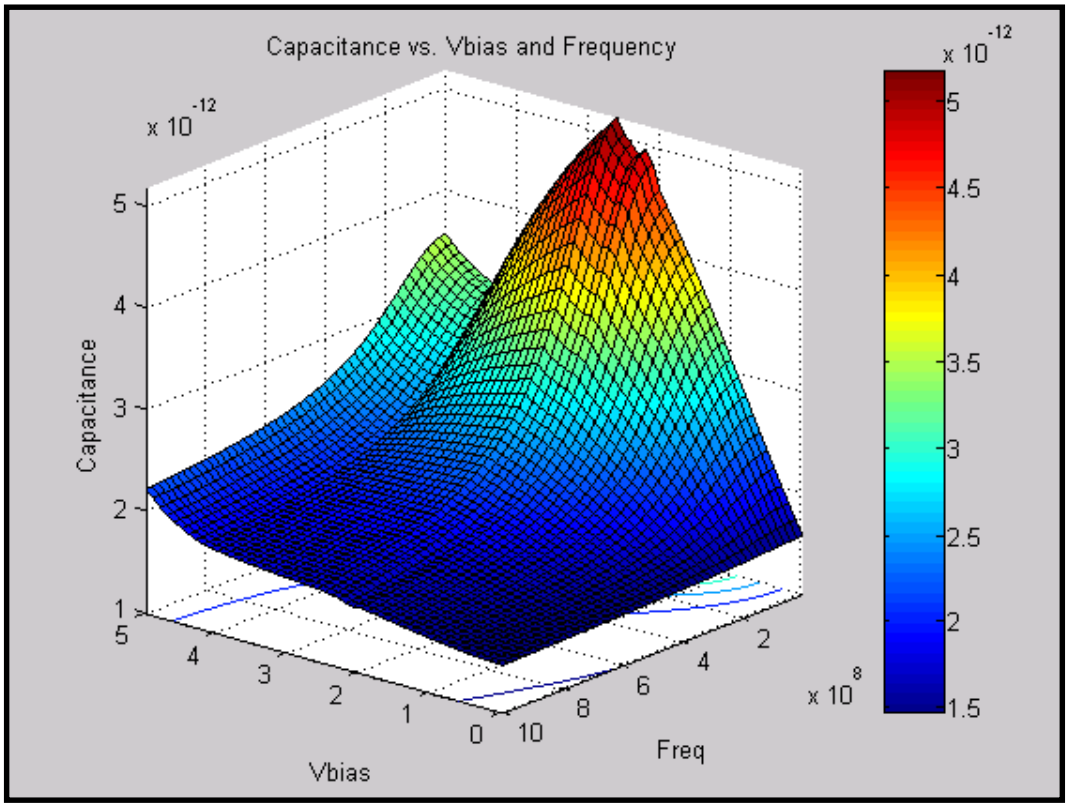
(I1-I2)/2 = C*dV/dt

This technique works whether the buffer is 3-stated or not!

Biased frequency sweep technique (FD)

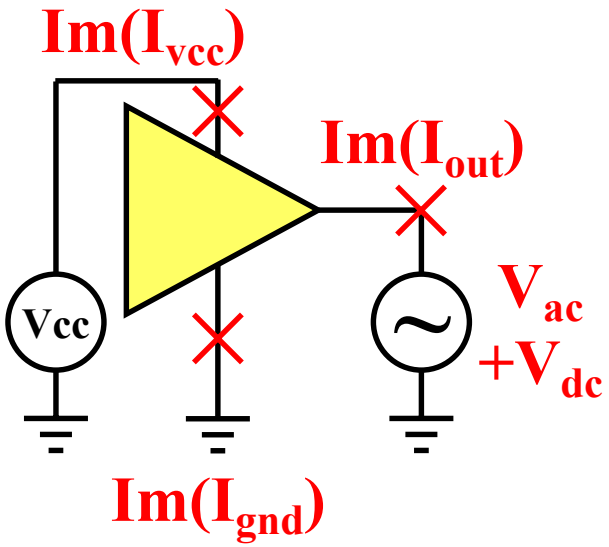


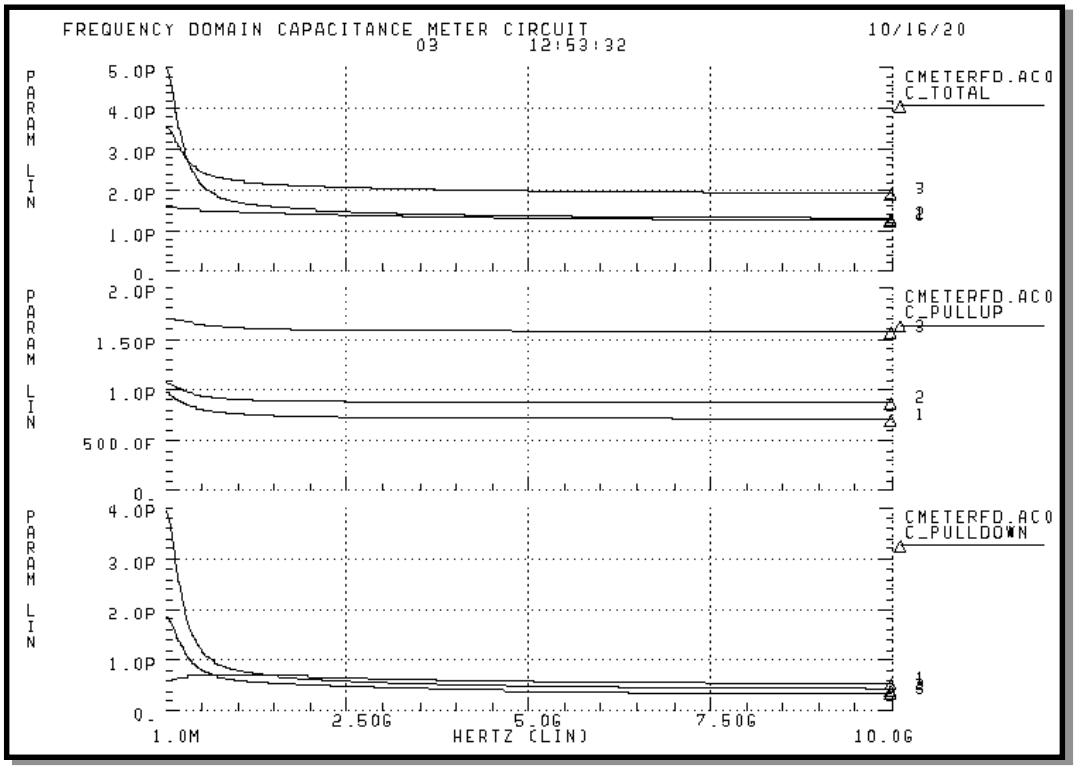
$$C = Im(I)/(2*\pi*f*V_{ac})$$



This technique will also work regardless whether the buffer is 3-stated or not!

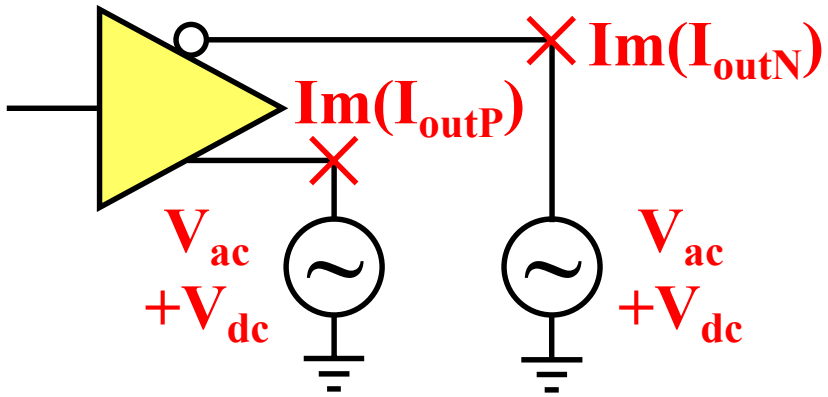
Separating C_{pu} and C_{pd} (for IBIS v4.0)





$$C_{total} = Im(I_{out})/(2*\pi*f*V_{ac})$$
$$C_{pu} = Im(I_{vcc})/(2*\pi*f*V_{ac})$$
$$C_{pd} = Im(I_{gnd})/(2*\pi*f*V_{ac})$$

Measuring C_{common} and C_{diff}



- **Run simulations with the above circuit**
  - Give one of the AC sources 0 V AC amplitude (makes it a DC source)
  - Give the other AC source a small AC amplitude (1 mV)
  - Give both of the sources an appropriate DC bias
- **Calculate capacitance using: $C = Im(I) / (2*\pi*f*Ampl)$**
  - For C_{diff} use the current of the "DC" source
  - For C_{common} use the current of "AC" source minus "DC" source
- **Repeat the above on both pads, drive high / low / 3-state**
- **Repeat everything at different DC bias voltages**

Limitations in IBIS regarding C_comp

- **The capacitance measurement setup on the previous slides provide frequency and voltage dependent capacitance**
- **Since C_comp in IBIS (through 4.0) can only use a "single" value we need to make some guesses for picking the best value from the available data**
- **Using the *-AMS language extensions in future versions of IBIS, it will be possible to write models which make use of all of this data**

Advanced features in IBIS v3.2

- **Multi-section uncoupled package description**
  - transmission line, package stubs
- **Single-section (lumped) full matrix coupled package description**
- **Electrical board description (EBD)**
- **Multi-stage buffer [Driver Schedule]**
- **Dynamic clamping and bus hold capabilities (on-die termination)**
- **Series pin to pin and FET bus switch modeling**
- **Model selector for programmable buffers**
- **Extended model specifications and simulation hooks**
  - ringback and hysteresis specifications

New features in IBIS v4.0

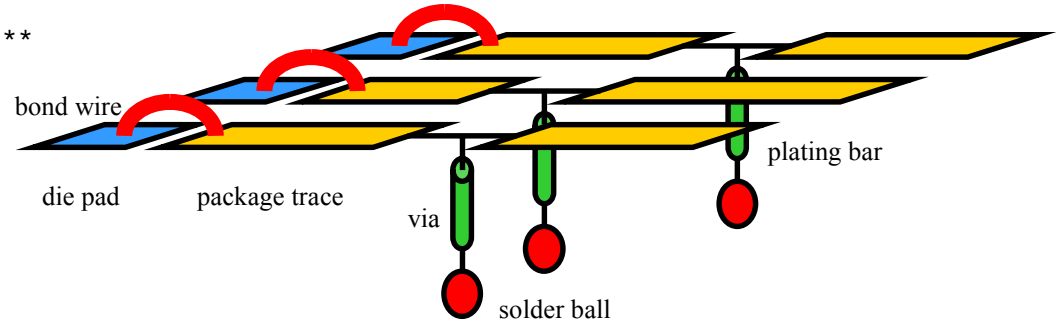
- **Enhanced receiver threshold specification**
  - Vth, Vinh_ac, Vinh_dc, Tslew_ac, Threshold_sensitivity, etc…
- **Alternate package models**
  - similar to [Model Selector]
- **C_comp refinements**
  - split C_comp into four parts for better return current accounting
- **Vref added to [Model Spec]**
- **Vt tables can be up to 1000 points long**
- **Golden waveforms**
- **More elaborate timing test loads**
  - to support PCI, PCI-X, and similar test loads
- **Series MOSFET allows N and P in parallel**
- **Fallback submodel**

Package modeling in IBIS

- **[Package] keyword**
  - this is a required section for each component
  - contains typical, minimum and maximum values for R_pkg, L_pkg and C_pkg
- **[Pin] keyword**
  - each pin can have a distinct R_pin, L_pin or C_pin value
  - these override the values under the [Package] keyword
  - only a single value can be used (no typ., min., max.)
- **[Package Model] keyword**
  - this can reference an external file or [Define Package Model] within the same .IBS file
  - overrides the values under the [Package] keyword
  - two methods possible currently
    - coupled single lump RLC matrix description (full, banded, and sparse matrix formats available)
    - uncoupled multi-section description using RLC, length

BGA package examples

```
|*****************************************
|
[Define Package Model] BGA example
[Manufacturer]         Noname Corp.
[OEM]                  Noname Corp.
[Description]          BGA
[Number Of Sections]   5
[Number Of Pins]       3
[Pin Numbers]
|
```




```
A1 Len=0 R=0.160 L=5.0n /        | bond wire
Len=4.00 L=0.170n C=0.250p /     | package trace
Fork
  Len=0.80 L=0.200n C=0.100p /   | plating bar
  Endfork
Len=0.010 L=0.010n C=0.020p /    | via
Len=0 C=2.0p /                   | Ball 1
|
A2 Len=0 R=0.160 L=5.0n /        | bond wire
Len=2.50 L=0.170n C=0.250p /     | package trace
Fork
  Len=0.80 L=0.200n C=0.100p /   | plating bar
  Endfork
Len=0.010 L=0.010n C=0.020p /    | via
Len=0 C=2.0p /                   | Ball 2
|
A3 Len=0 R=0.160 L=5.0n /        | bond wire
Len=4.00 L=0.170n C=0.250p /     | package trace
Fork
  Len=0.80 L=0.200n C=0.100p /   | plating bar
  Endfork
Len=0.010 L=0.010n C=0.020p /    | via
Len=0 C=2.0p /                   | Ball 3
```

```
| The resistance matrix for this package has no coupling
|
[Resistance Matrix]      Banded_matrix
[Bandwidth]              0
[Row]    1
10.0
[Row]    2
15.0
[Row]    3
15.0
|
[Inductance Matrix]      Full_matrix
[Row]    1
3.04859e-07      4.73185e-08      1.3428e-08
[Row]    2
3.04859e-07      4.73185e-08
[Row]    3
3.04859e-07
|
|
[Capacitance Matrix]     Sparse_matrix
[Row]    1
```

Single Line Equivalent Method

- **Multi section package description is uncoupled**
  - even/odd mode model must be made separately to account for the coupling effects

$$Z=\mathrm{Sqrt}(L/C) \quad Z_{even}=\mathrm{Sqrt}(L_{even}/C_{even}) \quad Z_{odd}=\mathrm{Sqrt}(L_{odd}/C_{odd})$$

$$v=1/\mathrm{Sqrt}(L*C) \quad v_{even}=1/\mathrm{Sqrt}(L_{even}*C_{even}) \quad v_{odd}=1/\mathrm{Sqrt}(L_{odd}*C_{odd})$$

$$L_{even}=L_{11}+L_{12} \quad L_{odd}=L_{11}-L_{12}$$

$$C_{even}=C_{11}-C_{12} \quad C_{odd}=C_{11}+C_{12}$$

Where C_{11} is the total capacitance of line 1 ($C_{line1\_to\_GND} + C_{12}$).

Electrical Board Description (EBD)

- **Even though the main focus of IBIS was buffer modeling, this capability became necessary for memory modules, processor cartridges, multi chip modules, etc.**
  - these kind of devices are mostly sold as a “canned unit”
  - very difficult to get routing information (Gerber files) for memory modules,
  - it is increasingly more important to simulate traces as transmission lines
- **The EBD syntax is very similar to the uncoupled multi-section package description**
  - uncoupled sections using RLC and length
  - two additional features are “node” and “pin”

The IBIS InterConnect Modeling Specification

- **The original connector specification idea was turned into a general purpose interconnect specification (ICM)**
  - it can be used to describe connectors, packages, and PCBs
- **ICM v1.0 has been ratified on 9/12/2003 and is available from the official IBIS web site**

  http://www.eda.org/pub/ibis/connector/
- **The ICM parser program is well under way**
- **Main features:**
  - supports single/multi line lossless and lossy RLGC models, and frequency dependent S-parameters
  - supports cascaded model matrixes, and arbitrary irregular structures
  - supports swath matrixes
  - supports multiple configurations of similar structures (mated / unmated, board edge / solder tail / press fit, etc.)

What does the future hold?

- **BIRD 75 approved by IBIS Open Forum on Jan. 10, 2003**
  - the goal is to remove the rigidity of the IBIS specification so that we would not need a new keyword every time a new buffer behavior type appears
  - BIRD 75 proposes a hook to SPICE, Verilog-AMS and VHDL-AMS languages
  - the *-AMS languages are equation based which become the algorithms for IBIS simulators, so no assumptions or hard coded algorithms will be needed any more
  - backwards compatible by keeping existing IBIS intact
  - BIRD 75 will become part of IBIS version 4.1
- **SPICE has similar problems**
  - model equations are hard coded in most SPICE versions (MOSFET level=xx)
  - users can only supply coefficients to these equations through the process files
  - new, deep sub-micron devices may need new model equations
  - implementation of new model equations can only be done by the tool vendor and takes a long time
  - many companies have proprietary models
  - Code Based Models were proposed at Intel to overcome these issues in SPICE

SPICE to IBIS checklist

1) Prepare a pin list for the component (.PIN file)
2) Prepare a clean SPICE netlist of the buffer for the I-V and V-t curve simulations
3) Run simulations for each buffer (.LIS files)
4) Convert each buffer's simulation output to IBIS format (.MRx files)
5) Combine individual buffer's IBIS models (.MRx) files into one IBIS model
6) Run IBISchk3 to verify the new IBIS model
7) Run SPICE vs. IBIS simulations to correlate new model

Preparing SPICE models

- **Prepare a clean SPICE netlist of the buffer for the I-V and V-t curve simulations**
  - make sure that you have the SPICE netlist and the process files for the buffer
  - remove all packaging, stimulus sources, transmission lines, test loads, etc. from “netlist” if there are any in it
  - make sure you understand the way the buffer needs to be connected to input, enable, output power, GND, reference voltages, and anything else it may need (strength selector, etc.)
  - find out the operating conditions of the buffer
    temperature (0 - 100 C, or anything else?)
    supply voltage (5.0 V, 3.3 V)
    supply voltage tolerance (± 5, or ±10 %)

Prepare the SPICE simulations (I-V)

- **I-V curve simulations**
  - could be done as a .DC sweep, or .TRAN simulation
  - .TRAN may work better in some cases (flip-flops, etc.)
  - if running in .TRAN mode use a slow sweep ramp
    parasitic capacitance of buffer can alter the actual currents if swept too fast (I=C*dV/dt)
  - set appropriate temperatures, supply voltages, sweep voltages, and time step for typical, minimum, and maximum curves
  - make sure you run the buffer in each mode
    driving high/low and 3-stated
  - measure pullup and power clamp curves relative to their supply rail
    (GND relative curves can be converted later if desired)
  - generate difference curves if necessary
    (this step could be done later also if so desired)

Prepare the SPICE simulations (V-t)

- **V-t curve simulations**
  - must be done with .TRAN simulation
  - select a small enough time step to get enough detail
  - make sure the length of the simulation is long enough to arrive to a steady state
    last point must match with the I-V curve / load line operating point solution
  - set the *same* temperatures, supply voltages, for typical, minimum, and maximum curves as for the I-V curves
  - select a proper value for R_fixture
    use the transmission line impedance value the buffer was designed to drive, or
    the voltage swing of the buffer loaded with R_fixture should be about 1/2 to 2/3 of the full swing without the load
  - make sure you run the rising / falling edges with R_fixture connected to each rail once (for complementary buffers)
    a minimum of four V-t curves per buffer are highly desirable
  - make sure that each V-t curve has a common time reference

Convert simulated data to IBIS format

- **Format simulation data to accommodate post processing tool (if any)**
  - this step may or may not be necessary
  - the HSPICE simulation results (.LIS files) of this course can be read directly by Cadence's Model Integrity tool (or IBIS Center)
- **Post process simulated data**
  - I-V curve subtraction
  - clamp I-V curve adjustments (to eliminate double counting)
  - reduce number of points to 100 per table (1000 for Vt)
  - guardband, derate if necessary
- **Convert data to IBIS syntax**
  - this can be done with MI, IBIS Center, or by hand
- **Concatenate individual buffer models to form an IBIS model for a complete component**
  - you will need additional information about the buffer for the last two steps that was not available from simulation

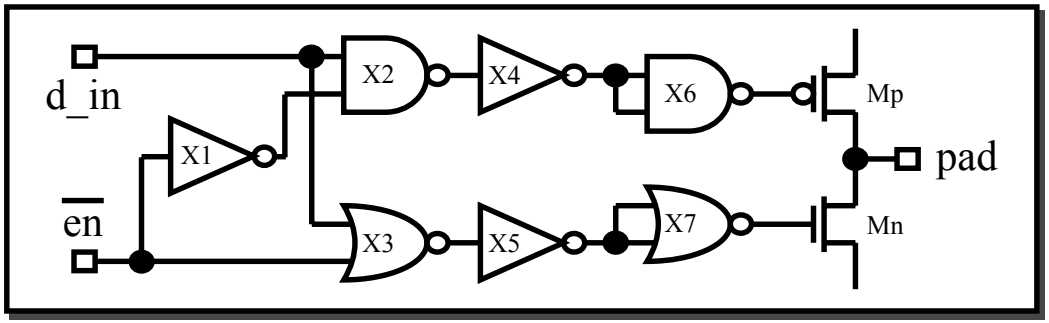
Process files and typ., min., max.

- **Process files cover a very wide range too often**
  - six sigma coverage is usual
  - design engineers do need this for their work
- **It may be more useful to make IBIS models with a smaller range**
  - system designer may never find a solution with such variations
  - customers may never get a part that falls outside a 3-4 sigma range due to testing, sorting and QA
- **Use "realistic" fast / slow process files, or**
- **Use typical process files with derating factors**
  - run minimum (worst case) and maximum (best case) simulations with temperature and supply voltage variations and apply a "fudge factor" during post processing the data to arrive to a 3-4 sigma "realistic" range

Word of caution when derating curves

- **I-V curves can be scaled easily to adjust min., and max. curves**
- **V-t curves must match the adjusted I-V curves!**
  - new swing amplitude must be calculated from new I-V curve
- **Edge rate adjustment on V-t curves means horizontal shrinking / stretching of the curve**
  - be careful with normalizing the “lead in” part of the V-t curves
- **Don’t forget to adjust the [Ramp] numbers to match the new V-t curves**

HSPICE buffer model example




```
*********************************************************************************
.SUBCKT  IO_buf  d_in pad power p_clamp ground g_clamp enable
*********************************************************************************
X1  enable  en_b            power  ground  INVERTER
X2  d_in    en_b    pre_p1  power  ground  NAND2
X3  d_in    enable  pre_n1  power  ground  NOR2
X4  pre_p1  pre_p2          power  ground  INVERTER  mult_p=2  mult_n=2
X5  pre_n1  pre_n2          power  ground  INVERTER  mult_p=2  mult_n=2
X6  pre_p2  pre_p2  gate_p  power  ground  NAND2     mult_p=4  mult_n=2
X7  pre_n2  pre_n2  gate_n  power  ground  NOR2      mult_p=2  mult_n=4
*
Mp  pad gate_p power  p_clamp  PMOS  L=0.800U W=43.40U NRD=0.0897 NRS=0.0737
+                              AS=434.0P AD=217.0P PS=106.8U PD=53.40U
+                              M=12
Mn  pad gate_n ground g_clamp  NMOS  L=0.800U W=43.40U NRD=0.0897 NRS=0.0714
+                              AS=434.0P AD=217.0P PS=106.8U PD=53.40U
+                              M=6
```

HSPICE template example (part 1)

```
********************* V-t curve simulations *********************
.SUBCKT BUFFER  1      2      3      4      5      6      7
*               d_in   pad    puref  PCLref pdref  GCLref /en
X0 1 2 3 4 5 6 7 IO_buf           $ <<<------ Change buffer name here
.ENDS
*********************************************************************************
.PROTECT
.LIB  'Process.lib'  Typ
.TRAN 25.0ps 15.0ns
*********************************************************************************
.TEMP   50                        $ Temperature of typical case
*--------------------------------------------------------------------------------*
.PARAM  PUref_typ  =  5.000V      $ Pullup reference voltage, typ.
.PARAM  PUref_min  =  4.750V      $ Pullup reference voltage, min.
.PARAM  PUref_max  =  5.250V      $ Pullup reference voltage, max.
.PARAM  PCLref_typ =  PUref_typ   $ Power clamp reference voltage, typ.
.PARAM  PCLref_min =  PUref_min   $ Power clamp reference voltage, min.
.PARAM  PCLref_max =  PUref_max   $ Power clamp reference voltage, max.
*--------------------------------------------------------------------------------*
.PARAM  PDref_typ  =  0.000V      $ Pulldown reference voltage, typ.
.PARAM  PDref_min  =  0.000V      $ Pulldown reference voltage, min.
.PARAM  PDref_max  =  0.000V      $ Pulldown reference voltage, max.
.PARAM  GCLref_typ =  PDref_typ   $ GND clamp reference voltage, typ.
.PARAM  GCLref_min =  PDref_min   $ GND clamp reference voltage, min.
.PARAM  GCLref_max =  PDref_max   $ GND clamp reference voltage, max.
*********************************************************************************
.PARAM  PD_ref     = PDref_typ    $ Reference voltages for typical case
.PARAM  GCL_ref    = GCLref_typ
.PARAM  PU_ref     = PUref_typ
.PARAM  PCL_ref    = PCLref_typ
*--------------------------------------------------------------------------------*
.PARAM  Ven        = PD_ref       $ Active-low enable
*.PARAM  Ven        = PU_ref       $ Active-high enable
*********************************************************************************
```

HSPICE template example (part 2)

```
*******************************************************************************
.PARAM  Vfx_pd_on  = PU_ref
.PARAM  Vfx_pd_off = PU_ref
.PARAM  Vfx_pu_on  = PD_ref
.PARAM  Vfx_pu_off = PD_ref
*
.PARAM  Rfx_pd_on  = 50
.PARAM  Rfx_pd_off = 50
.PARAM  Rfx_pu_on  = 50
.PARAM  Rfx_pu_off = 50
*
.PARAM  Cfx_pd_on  = 0.0pF
.PARAM  Cfx_pd_off = 0.0pF
.PARAM  Cfx_pu_on  = 0.0pF
.PARAM  Cfx_pu_off = 0.0pF
*******************************************************************************
.MEASURE TRAN  Vpower           PARAM = 'PU_ref-PD_ref'
.MEASURE TRAN  Vfixture_pd_on   PARAM = Vfx_pd_on
.MEASURE TRAN  Vfixture_pd_off  PARAM = Vfx_pd_off
.MEASURE TRAN  Vfixture_pu_on   PARAM = Vfx_pu_on
.MEASURE TRAN  Vfixture_pu_off  PARAM = Vfx_pu_off
.MEASURE TRAN  Rfixture_pd_on   PARAM = Rfx_pd_on
.MEASURE TRAN  Rfixture_pd_off  PARAM = Rfx_pd_off
.MEASURE TRAN  Rfixture_pu_on   PARAM = Rfx_pu_on
.MEASURE TRAN  Rfixture_pu_off  PARAM = Rfx_pu_off
.MEASURE TRAN  Cfixture_pd_on   PARAM = Cfx_pd_on
.MEASURE TRAN  Cfixture_pd_off  PARAM = Cfx_pd_off
.MEASURE TRAN  Cfixture_pu_on   PARAM = Cfx_pu_on
.MEASURE TRAN  Cfixture_pu_off  PARAM = Cfx_pu_off
*******************************************************************************
.OPTIONS BRIEF INGOLD NUMDGT=8 CO=132 ACCT=0 NOWARN ACCURATE RMAX=0.5
*.OPTIONS POST=1 POST_VERSION=9007 PROBE
.PRINT TRAN Pulldown_on  = V(Out_pd_on)
+           Pulldown_off = V(Out_pd_off)
+           Pullup_on    = V(Out_pu_on)
+           Pullup_off   = V(Out_pu_off)
*******************************************************************************
```

HSPICE template example (part 3)

```
*****************************************************************************
Vrefpd      pdref    0   DC = PD_ref
VrefGNDcl   GCLref   0   DC = GCL_ref
Vrefpu      puref    0   DC = PU_ref
VrefPOWcl   PCLref   0   DC = PCL_ref
*
Von         Von      0   DC = Ven
Voff        Voff     0   DC = 'PU_ref-(Ven-PD_ref)'
*
Vpls_r   Pls_r   0   PWL   0.0ns PD_ref 1.0ps PU_ref 100ns PU_ref
Vpls_f   Pls_f   0   PWL   0.0ns PU_ref 1.0ps PD_ref 100ns PD_ref
************************************************************** pull-down on
Xdrv_pd_on    Pls_f      Out_pd_on    puref   PCLref   pdref   GCLref   Von   BUFFER
Rfxt_pd_on    V_pd_on    Out_pd_on    Rfx_pd_on
Cfxt_pd_on    V_pd_on    Out_pd_on    Cfx_pd_on
Vfxt_pd_on    V_pd_on    0            Vfx_pd_on
*------------------------------------------------------------- pull-down off
Xdrv_pd_off   Pls_r      Out_pd_off   puref   PCLref   pdref   GCLref   Von   BUFFER
Rfxt_pd_off   V_pd_off   Out_pd_off   Rfx_pd_off
Cfxt_pd_off   V_pd_off   Out_pd_off   Cfx_pd_off
Vfxt_pd_off   V_pd_off   0            Vfx_pd_off
*--------------------------------------------------------------- pull-up on
Xdrv_pu_on    Pls_r      Out_pu_on    puref   PCLref   pdref   GCLref   Von   BUFFER
Rfxt_pu_on    V_pu_on    Out_pu_on    Rfx_pu_on
Cfxt_pu_on    V_pu_on    Out_pu_on    Cfx_pu_on
Vfxt_pu_on    V_pu_on    0            Vfx_pu_on
*-------------------------------------------------------------- pull-up off
Xdrv_pu_off   Pls_f      Out_pu_off   puref   PCLref   pdref   GCLref   Von   BUFFER
Rfxt_pu_off   V_pu_off   Out_pu_off   Rfx_pu_off
Cfxt_pu_off   V_pu_off   Out_pu_off   Cfx_pu_off
Vfxt_pu_off   V_pu_off   0            Vfx_pu_off
*****************************************************************************
```

HSPICE template example (part 4)

```
*******************************************************************************
.UNPROTECT
*******************************************************************************
.ALTER                              $ Minimum case
.PROTECT
.DEL LIB  'Process.lib'  Typ
.LIB      'Process.lib'  Slow
.TEMP   100                         $ Temperature for minimum case
.PARAM  PD_ref     = PDref_min      $ Reference voltages for minimum case
.PARAM  GCL_ref    = GCLref_min
.PARAM  PU_ref     = PUref_min
.PARAM  PCL_ref    = PCLref_min
.UNPROTECT
*******************************************************************************
.ALTER                              $ Maximum case
.PROTECT
.DEL LIB  'Process.lib'  Slow
.LIB      'Process.lib'  Fast
.TEMP   0                           $ Temperature for maximum case
.PARAM  PD_ref     = PDref_max      $ Reference voltages for maximum case
.PARAM  GCL_ref    = GCLref_max
.PARAM  PU_ref     = PUref_max
.PARAM  PCL_ref    = PCLref_max
.UNPROTECT
*******************************************************************************
.END
```

HSPICE template example (part 5)

```
********************* I-V curve simulations *********************
.SUBCKT BUFFER  1       2       3       4       5       6       7
*               d_in    pad     puref   PCLref pdref   GCLref /en
X0 1 2 3 4 5 6 7 IO_buf             $ <<<------ Change buffer name here
.ENDS
*********************************************************************************
.PROTECT
.LIB  'Process.lib'  Typ
*********************************************************************************
.TEMP   50                          $ Temperature of typical case
*---------------------------------------------------------------------------------*
.PARAM  Resolution =  5.0mV         $ Voltage resolution of sweep
*********************************************************************************
.PARAM  PUref_typ  =  5.000V        $ Pullup reference voltage, typ.
.PARAM  PUref_min  =  4.750V        $ Pullup reference voltage, min.
.PARAM  PUref_max  =  5.250V        $ Pullup reference voltage, max.
.PARAM  PCLref_typ =  PUref_typ     $ Power clamp reference voltage, typ.
.PARAM  PCLref_min =  PUref_min     $ Power clamp reference voltage, min.
.PARAM  PCLref_max =  PUref_max     $ Power clamp reference voltage, max.
*---------------------------------------------------------------------------------*
.PARAM  PDref_typ  =  0.000V        $ Pulldown reference voltage, typ.
.PARAM  PDref_min  =  0.000V        $ Pulldown reference voltage, min.
.PARAM  PDref_max  =  0.000V        $ Pulldown reference voltage, max.
.PARAM  GCLref_typ =  PDref_typ     $ GND clamp reference voltage, typ.
.PARAM  GCLref_min =  PDref_min     $ GND clamp reference voltage, min.
.PARAM  GCLref_max =  PDref_max     $ GND clamp reference voltage, max.
*********************************************************************************
.PARAM  PD_ref     = PDref_typ      $ Reference voltages for typical case
.PARAM  GCL_ref    = GCLref_typ
.PARAM  PU_ref     = PUref_typ
.PARAM  PCL_ref    = PCLref_typ
*---------------------------------------------------------------------------------*
.PARAM  Ven        = PD_ref         $ Active-low enable
*.PARAM  Ven        = PU_ref        $ Active-high enable
*********************************************************************************
```

HSPICE template example (part 6)

```
*******************************************************************************
.MEASURE TRAN  Vpower        PARAM = 'PU_ref-PD_ref'
.MEASURE TRAN  Pulldown_ref  PARAM = PD_ref
.MEASURE TRAN  GND_cl_ref    PARAM = GCL_ref
.MEASURE TRAN  Pullup_ref    PARAM = PU_ref
.MEASURE TRAN  POWER_cl_ref  PARAM = PCL_ref
*******************************************************************************
.TRAN  Step_t  Sweep_t  START= Sweep_d
.OPTIONS BRIEF INGOLD NUMDGT=10 CO=132 ACCT=0 NOWARN ACCURATE RMAX=0.5
*.OPTIONS POST=1 POST_VERSION=9007 PROBE
.PRINT TRAN
+ V_sweep      = PAR('Resolution*(TIME-Sweep_d)/Step_t + SWstart')
+ I_pulldown   = I(Vpulldown)
+ I_gndclamp   = I(Vgnd_clamp)
+ I_pullup     = I(Vpullup)
+ I_powerclamp = I(Vpower_clamp)
*******************************************************************************
Vrefpd     pdref    0  DC = PD_ref
VrefGNDcl  GCLref   0  DC = GCL_ref
Vrefpu     puref    0  DC = PU_ref
VrefPOWcl  PCLref   0  DC = PCL_ref
*
Von        Von      0  DC = Ven
Voff       Voff     0  DC = 'PU_ref-(Ven-PD_ref)'
*-------------------------------------------------------------------------------*
```

HSPICE template example (part 7)

```
*-------------------------------------------------------------------------------*
.PARAM
+ DRspan         = 'PUref_typ-PDref_typ + Resolution'
+ CLspan         = 'PCLref_typ-GCLref_typ + Resolution'
*
+ PCLstart_typ   = 'PCLref_typ - max(PUref_typ +   (DRspan) , PCLref_typ +   (CLspan))'
+ PCLend_typ     = 'PCLref_typ - min(PUref_typ - 2*(DRspan) , PCLref_typ - 2*(CLspan))'
+ GCLstart_typ   = 'min(PDref_typ -   (DRspan) , GCLref_typ -   (CLspan)) - GCLref_typ'
+ GCLend_typ     = 'max(PDref_typ + 2*(DRspan) , GCLref_typ + 2*(CLspan)) - GCLref_typ'
*
+ PCLstart_min   = 'PCLref_min - max(PUref_min +   (DRspan) , PCLref_min +   (CLspan))'
+ PCLend_min     = 'PCLref_min - min(PUref_min - 2*(DRspan) , PCLref_min - 2*(CLspan))'
+ GCLstart_min   = 'min(PDref_min -   (DRspan) , GCLref_min -   (CLspan)) - GCLref_min'
+ GCLend_min     = 'max(PDref_min + 2*(DRspan) , GCLref_min + 2*(CLspan)) - GCLref_min'
*
+ PCLstart_max   = 'PCLref_max - max(PUref_max +   (DRspan) , PCLref_max +   (CLspan))'
+ PCLend_max     = 'PCLref_max - min(PUref_max - 2*(DRspan) , PCLref_max - 2*(CLspan))'
+ GCLstart_max   = 'min(PDref_max -   (DRspan) , GCLref_max -   (CLspan)) - GCLref_max'
+ GCLend_max     = 'max(PDref_max + 2*(DRspan) , GCLref_max + 2*(CLspan)) - GCLref_max'
*
+ nPCLstart      = 'min(PCLstart_typ,min(PCLstart_min,PCLstart_max))'
+ xPCLend        = 'max(PCLend_typ  ,max(PCLend_min  ,PCLend_max))'
+ nGCLstart      = 'min(GCLstart_typ,min(GCLstart_min,GCLstart_max))'
+ xGCLend        = 'max(GCLend_typ  ,max(GCLend_min  ,GCLend_max))'
*
+ SWstart        = 'min(-1*DRspan,min(nPCLstart,nGCLstart))'
+ SWend          = 'max(   DRspan,max(xPCLend  ,xGCLend))'
*
+ Step_t         = 1.0ms                $ Step size of sweep
+ Sweep_d        = 1.0us                $ Delay before sweep begins
+ Sweep_t        = 'Sweep_d + Step_t*(SWend-SWstart)/Resolution'
*********************************************************************************
```

HSPICE template example (part 8)

```
*****************************************************************************
Vpulldown     pdref   pulldown     PWL
+  0ms                                                                    0V
+  'Sweep_d + Step_t *                      max(0,-1*DRspan-SWstart) /Resolution'  'DRspan'
+  'Sweep_d + Step_t *          (3*DRspan + max(0,-1*DRspan-SWstart))/Resolution'  '-2*DRspan'
+  'Sweep_t + Sweep_d'                                                    0V
Vgnd_clamp    GCLref  gnd_clamp    PWL
+  0ms                                                                    0V
+  'Sweep_d + Step_t *                      max(0,nGCLstart-SWstart) /Resolution'  '-1*nGCLstart'
+  'Sweep_d + Step_t * (xGCLend-nGCLstart + max(0,nGCLstart-SWstart))/Resolution'  '-1*xGCLend'
+  'Sweep_t + Sweep_d'                                                    0V
Vpullup       puref   pullup       PWL
+  0ms                                                                    0V
+  'Sweep_d + Step_t *                      max(0,-1*SWstart-DRspan) /Resolution'  '-1*DRspan'
+  'Sweep_d + Step_t *          (3*DRspan + max(0,-1*SWstart-DRspan))/Resolution'  '2*DRspan'
+  'Sweep_t + Sweep_d'                                                    0V
Vpower_clamp  PCLref  power_clamp  PWL
+  0ms                                                                    0V
+  'Sweep_d + Step_t *                      max(0,nPCLstart-SWstart) /Resolution'  'nPCLstart'
+  'Sweep_d + Step_t * (xPCLend-nPCLstart + max(0,nPCLstart-SWstart))/Resolution'  'xPCLend'
+  'Sweep_t + Sweep_d'                                                    0V
*****************************************************************************
Xpulldown   pdref  pulldown     puref  PCLref  pdref  GCLref  Von   BUFFER
Xgnd_clamp  pdref  gnd_clamp    puref  PCLref  pdref  GCLref  Voff  BUFFER
*-----------------------------------------------------------------------------*
Xpullup     puref  pullup       puref  PCLref  pdref  GCLref  Von   BUFFER
Xpow_clamp  puref  power_clamp  puref  PCLref  pdref  GCLref  Voff  BUFFER
*****************************************************************************
.UNPROTECT
*****************************************************************************
```

HSPICE template example (part 9)

```
*********************************************************************************
.ALTER                              $ Minimum case
.PROTECT
.DEL LIB  'Process.lib'  Typ
.LIB      'Process.lib'  Slow
.TEMP   100                         $ Temperature for minimum case
.PARAM  PD_ref     = PDref_min      $ Reference voltages for minimum case
.PARAM  GCL_ref    = GCLref_min
.PARAM  PU_ref     = PUref_min
.PARAM  PCL_ref    = PCLref_min
.UNPROTECT
*********************************************************************************
.ALTER                              $ Maximum case
.PROTECT
.DEL LIB  'Process.lib'  Slow
.LIB      'Process.lib'  Fast
.TEMP   0                           $ Temperature for maximum case
.PARAM  PD_ref     = PDref_max      $ Reference voltages for maximum case
.PARAM  GCL_ref    = GCLref_max
.PARAM  PU_ref     = PUref_max
.PARAM  PCL_ref    = PCLref_max
.UNPROTECT
*********************************************************************************
.END
```

Additional information

- **Collect data for those parameters which are not available directly from SPICE simulations**
  - model type (I/O can be used as input or output only)
  - Vinh, Vinl, etc…
  - C_comp (may need to run a separate simulation for this)
  - package L, R, C
  - Vmeas, Vref, Rref, Cref
  - component name
  - copyright note
  - source of the model (measured, simulated)
  - disclaimers
  - etc…

Lab #1 - make an IBIS model

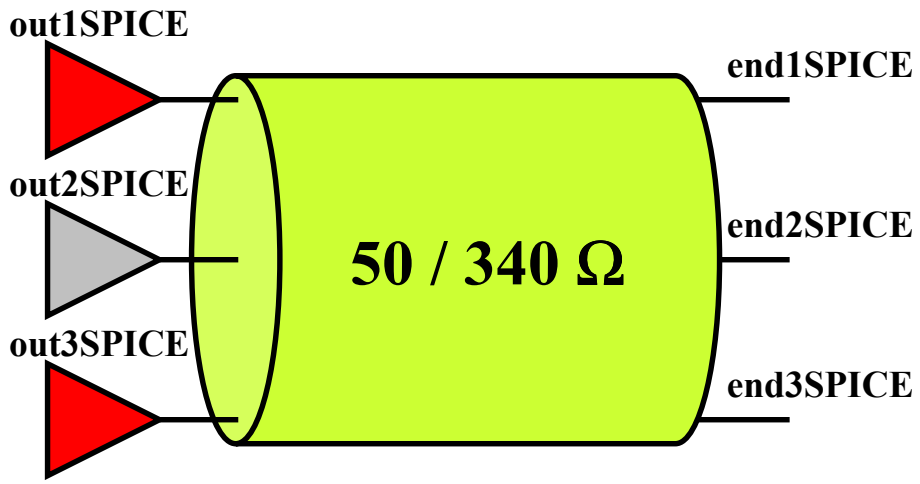
- **Run the HSPICE template for each buffer**
  - observe the templates (.SP files) and the SPICE models (.INC files) of the buffers with a text editor (and make changes if needed or desired)
  - run an HSPICE simulation with each .SP file in the directory
    in Windows Explorer right click on .SP file and select "Run HSPICE"
    at UNIX command line type: *hspice -i filename.sp -o filename.lis*
- **Make an IBIS equivalent for each .LIS file using Model Integrity (or IBIS Center)**
  - have the tool read each of the .LIS files you generated with HSPICE in the previous step
  - convert each .LIS file to IBIS format (.BUF file with MI or .MR1 file with IBIS Center
- **Make a complete .IBS file from the individual buffer files (.BUF or .MR1)**
  - observe the contents of the .PIN file (check buffer names in pin list)
  - have the tool concatenate all of the buffer files into a complete IBIS file using the .PIN file

Lab #2 - verify the new IBIS model

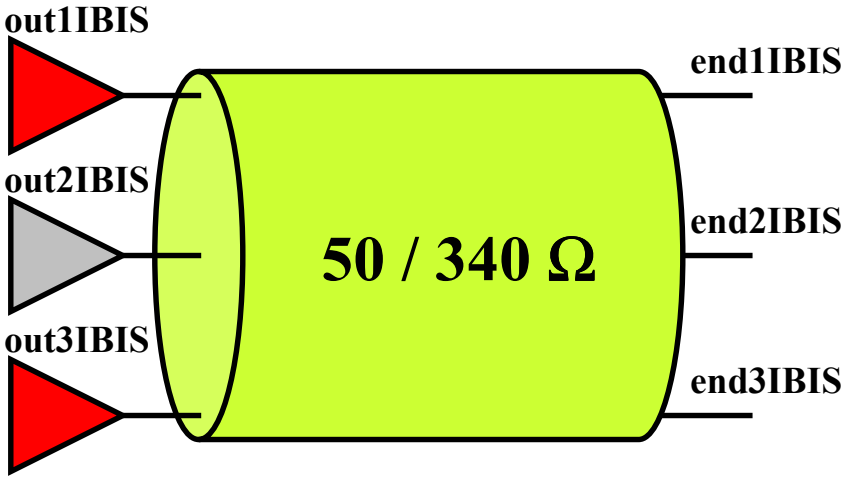
- **Visual check**
  - open the IBIS buffer files (.BUF or .MR1) and/or the .IBS file with MI or IBIS Center and overlay the I-V and V-t curves with the I-V and V-t curves in the .LIS file(s)
- **Syntax check**
  - run the IBIS checker program on the IBIS file(s) you made

    Model Integrity can do this automatically for you
    in Windows Explorer right click on .IBS file and select "Run IBISchk3"
    at UNIX command line type: *path/ibischk3 IBIS_file [> output_file]*
- **Functional check**
  - run a simulation with the IBIS model directly in MI using the the IBIS model's own loading information
  - run a simulation with the original SPICE model and the new IBIS model using the compare.sp file and compare the waveforms of the two buffers by overlaying them in AWAVES

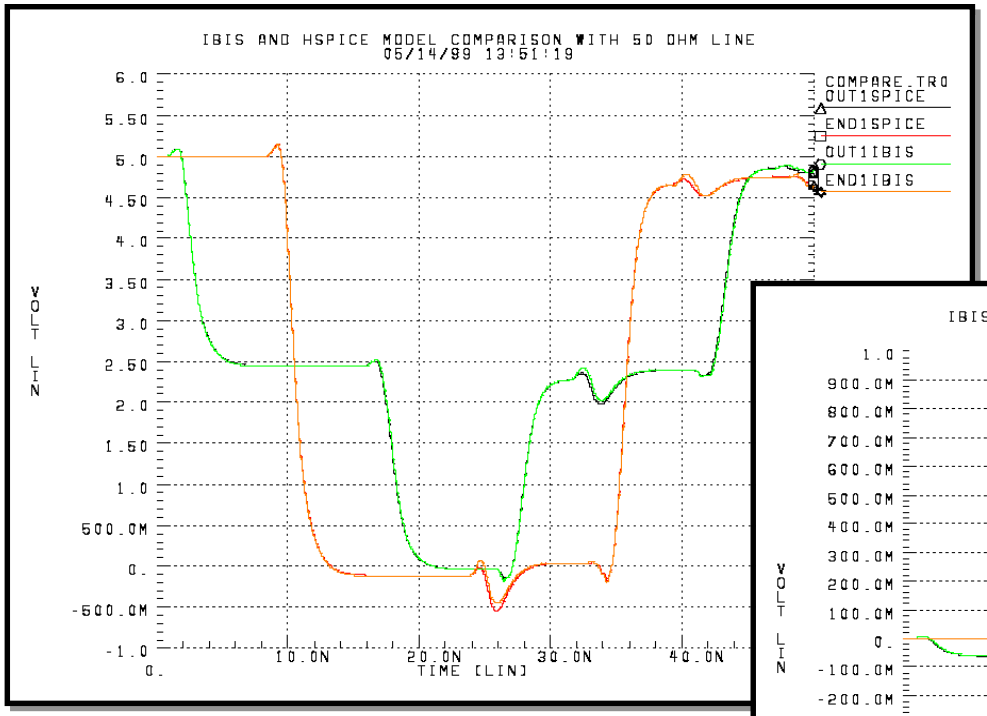
SPICE and IBIS simulation comparison



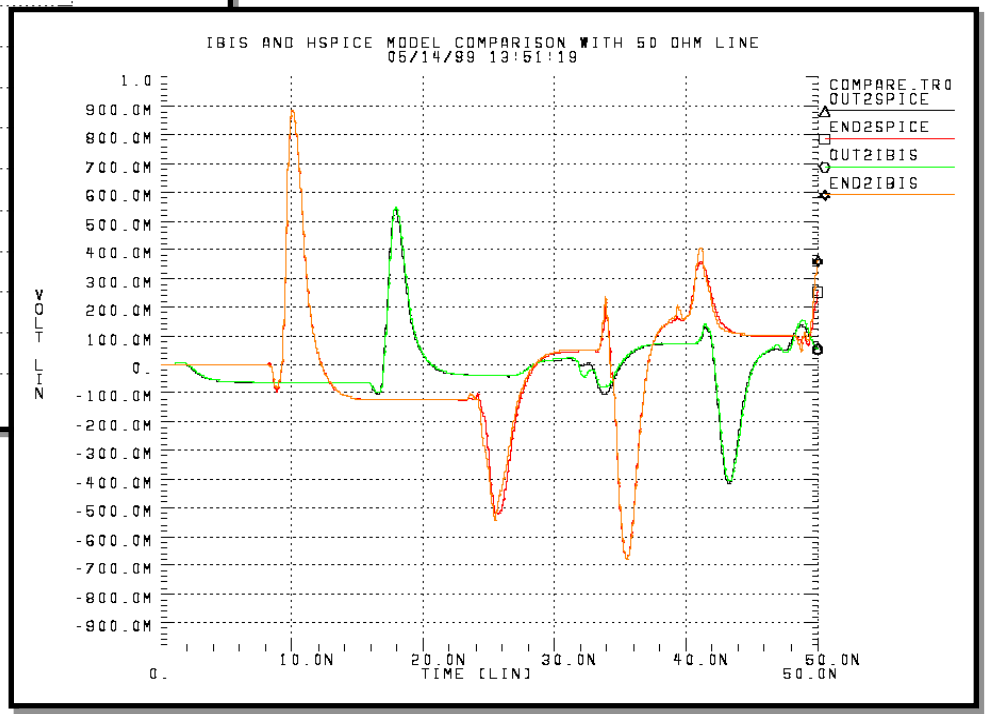
The IBIS model's waveforms were generated with a 50 Ω load for these buffers



Waveforms with 50 Ω Line

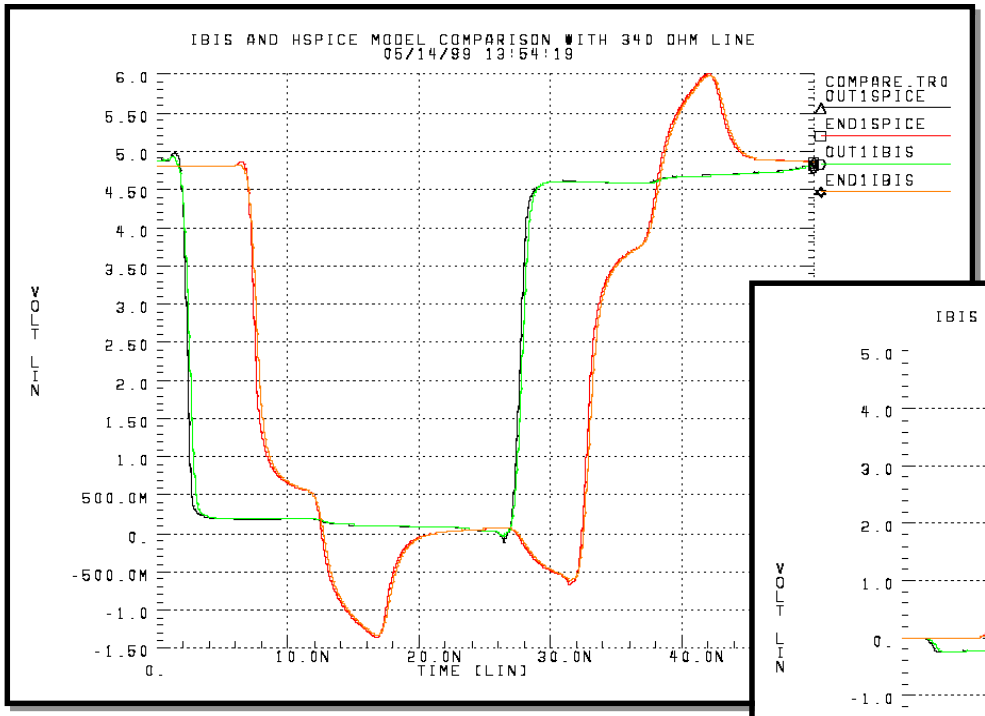


Driven line

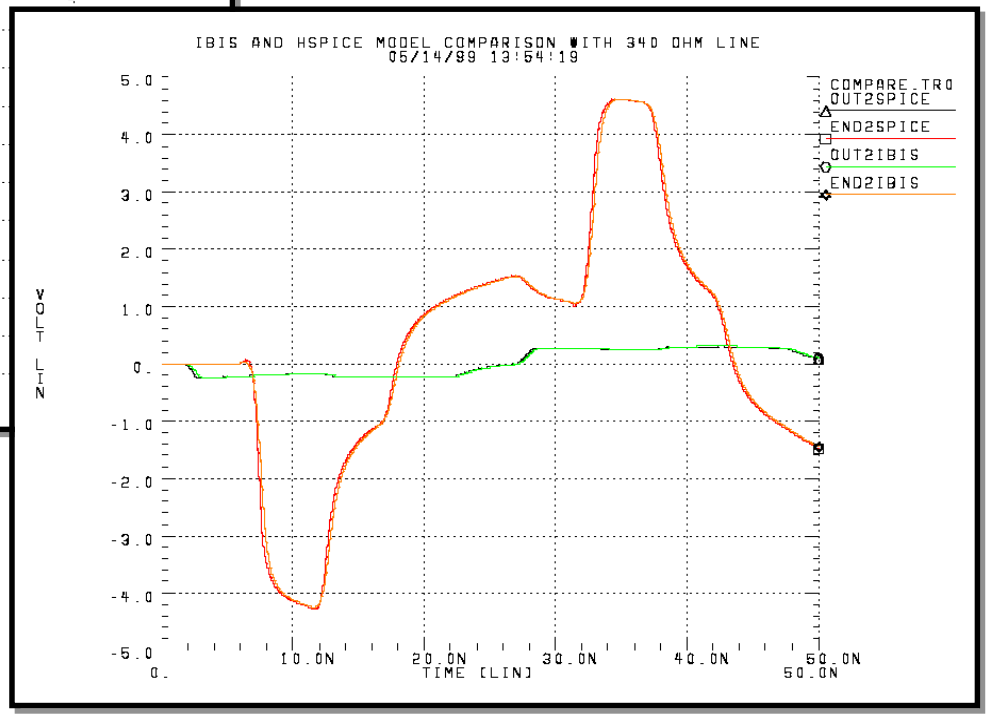


Cross talk on quiet line

Waveforms with 340 Ω Line



Driven line



Cross talk on quiet line

Advanced buffer modeling

- **Pullup or pulldown "resistors"**
  - they prevent 3-stated buses from floating around the threshold voltages
  - usually in the kΩ range (I_{sat} in μA range)
  - usually implemented as a transistor turned on constantly
- **Integrated terminators**
  - static transmission line termination (low impedance)
  - dynamic implementations designed to save power
- **Bus hold circuits (may be dynamic)**
  - similar to pu/pd resistor idea, but usually has a lower impedance
  - could be time, edge or level dependent if dynamic
- **Dynamic clamping mechanisms**
  - strong clamps turn on momentarily to prevent excessive overshoot
- **Staged buffers**
  - mostly used in slew rate controlled drivers
- **Kicker circuits**
  - transition boosters and then turn off
- **Anything else you can invent goes here...**

Modeling static advanced features

- **Anything that is ON constantly should be modeled using the [Power Clamp] or [GND Clamp] I-V curves**
  - pullup or pulldown “resistors”
  - static integrated terminators
  - static clamps
  - static bus hold circuits
- **Make sure you are using the appropriate rail for correct power and GND bounce simulation purposes**
  - use [Power Clamp] for pullup resistor
  - [GND Clamp] for pulldown resistor, etc.
- **Some additional post processing may be required to avoid double counting**

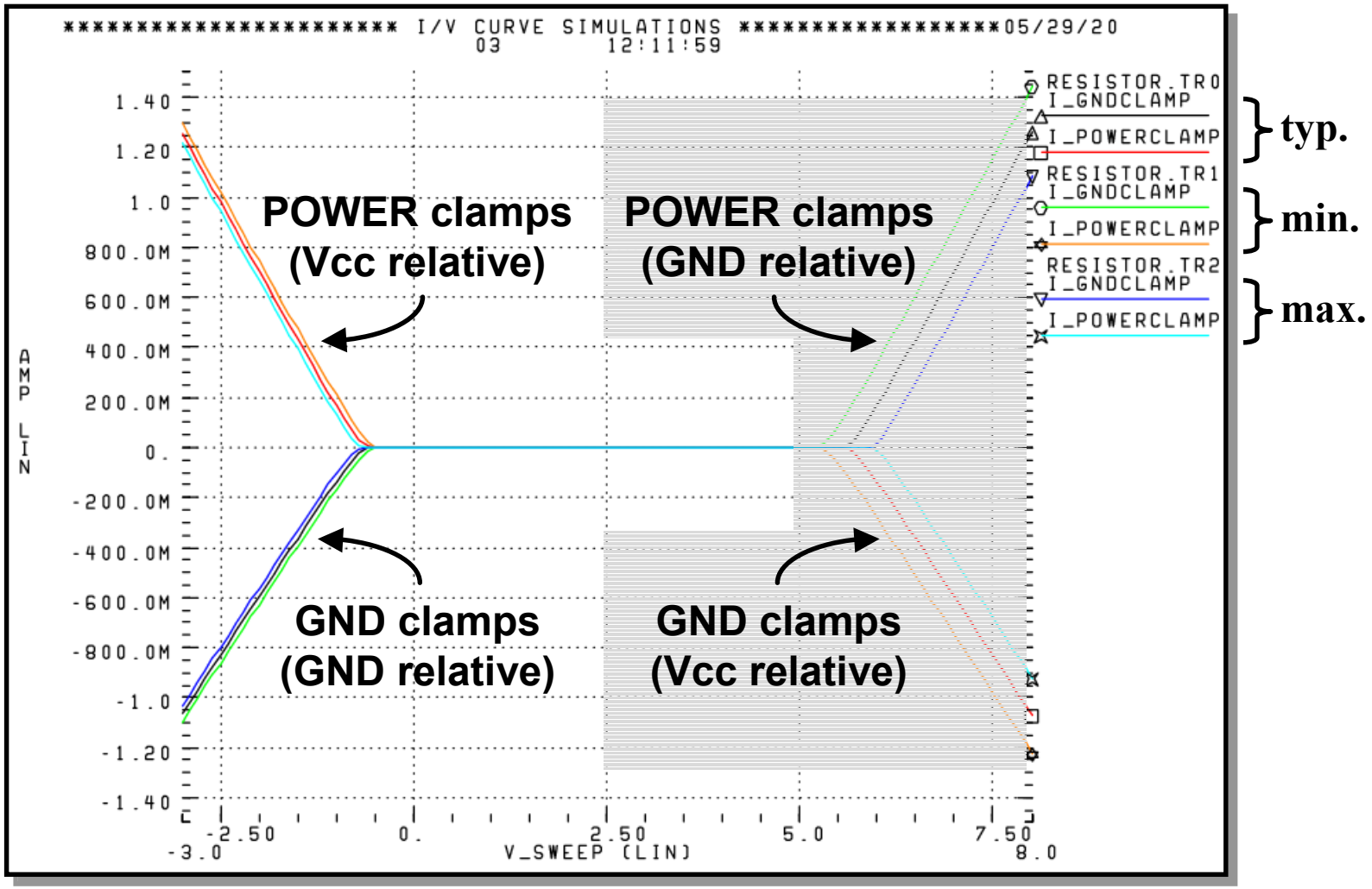
Modeling dynamic advanced features

- **Use IBIS version 3.2 features**
  - keywords: [Driver Schedule], [Add Submodel], [Submodel], [Submodel Spec]
  - subparameters: Dynamic_clamp, Bus_hold, Fall_back
- **Detailed knowledge of circuit behavior is required**
- **Familiarity with buffer's SPICE netlist required**
- **May have to dissect or modify SPICE netlist to generate necessary data in separate steps**
- **It may not be possible to make such models from simple and/or direct lab measurements**

On-die terminations

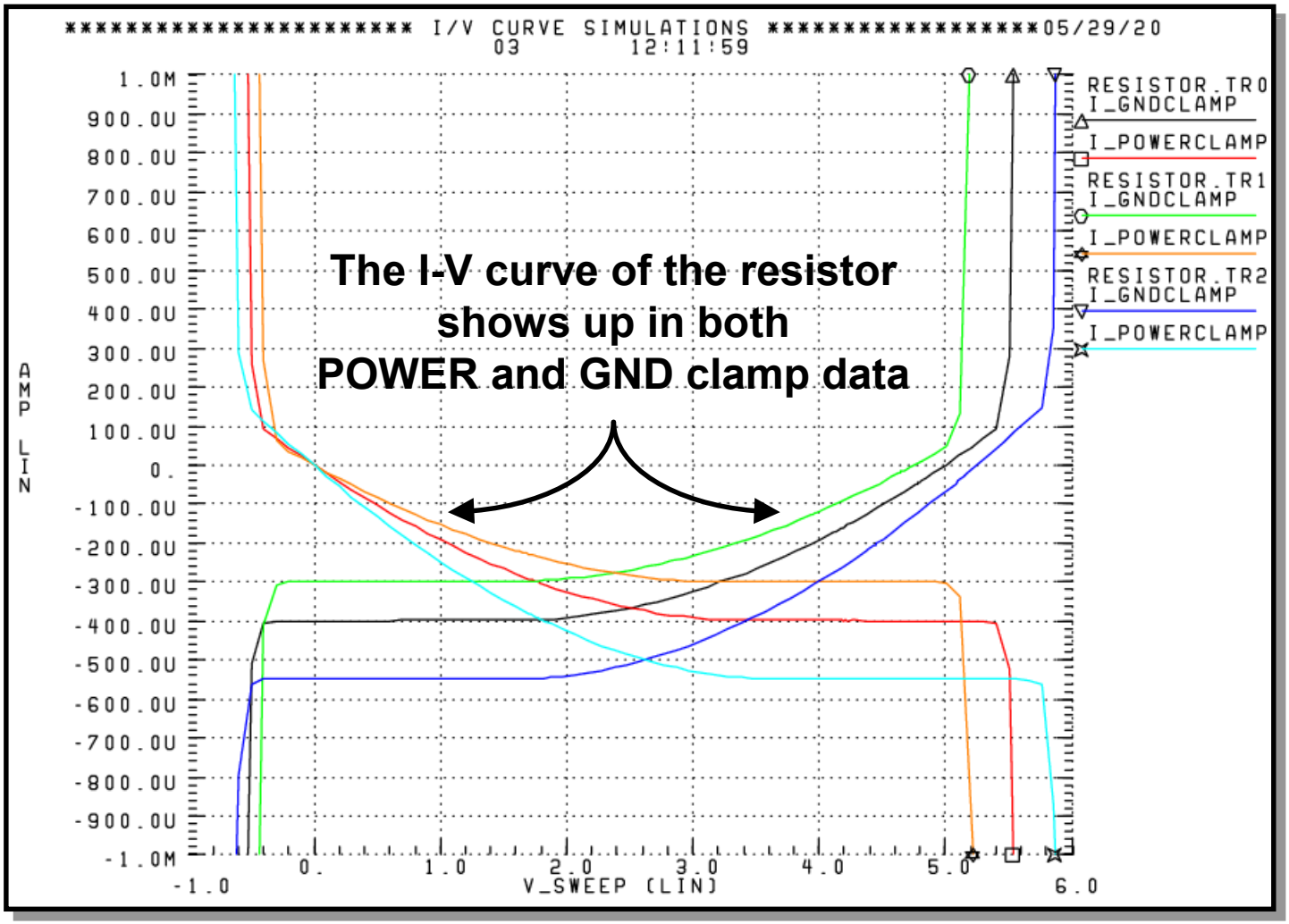
- **Series termination**
  - does not require any special work because it is described by the shape of the I-V curve
- **Parallel termination**
  - if the parallel termination is on all the time, use the method described for pullup/pulldown resistors
- **Switched parallel termination**
  - the parallel termination device is turned off while the opposite half of the buffer is driving
  - make a normal complementary model for the driver portion of the buffer
  - make a difference I-V curve for the terminator device and use the **[Add Submodel]** keyword in **non-driving** mode with the **[Submodel]** keyword's **dynamic_clamp** in static mode (without a pulse)

Pullup resistor example

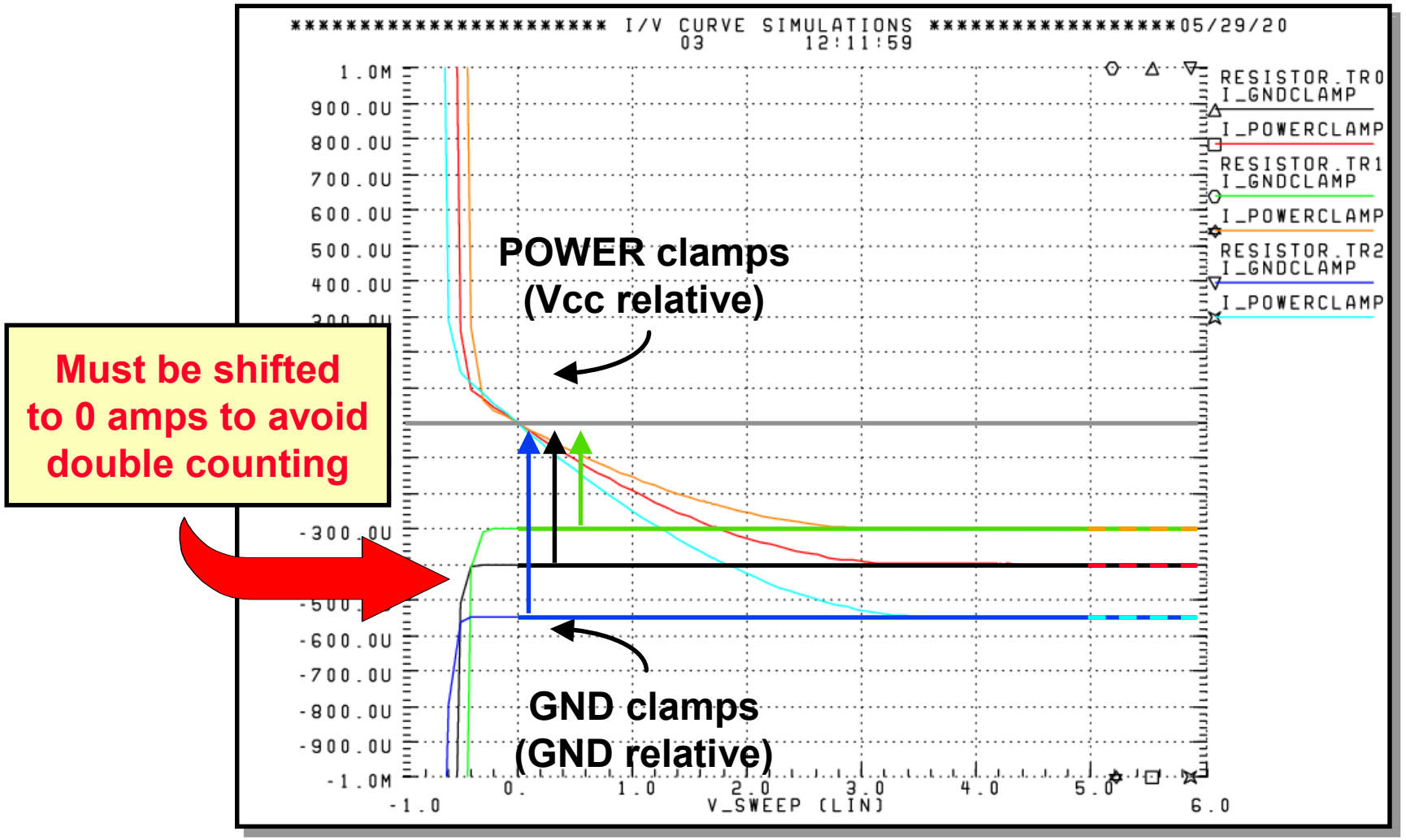


I-V curves of a 3-stated buffer with pullup R

Zooming in on I-V curves



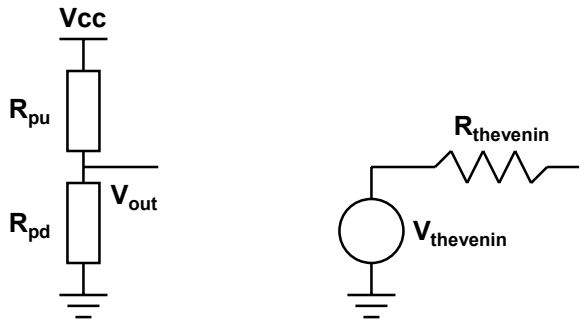
Algorithm in pictures



Algorithm in words

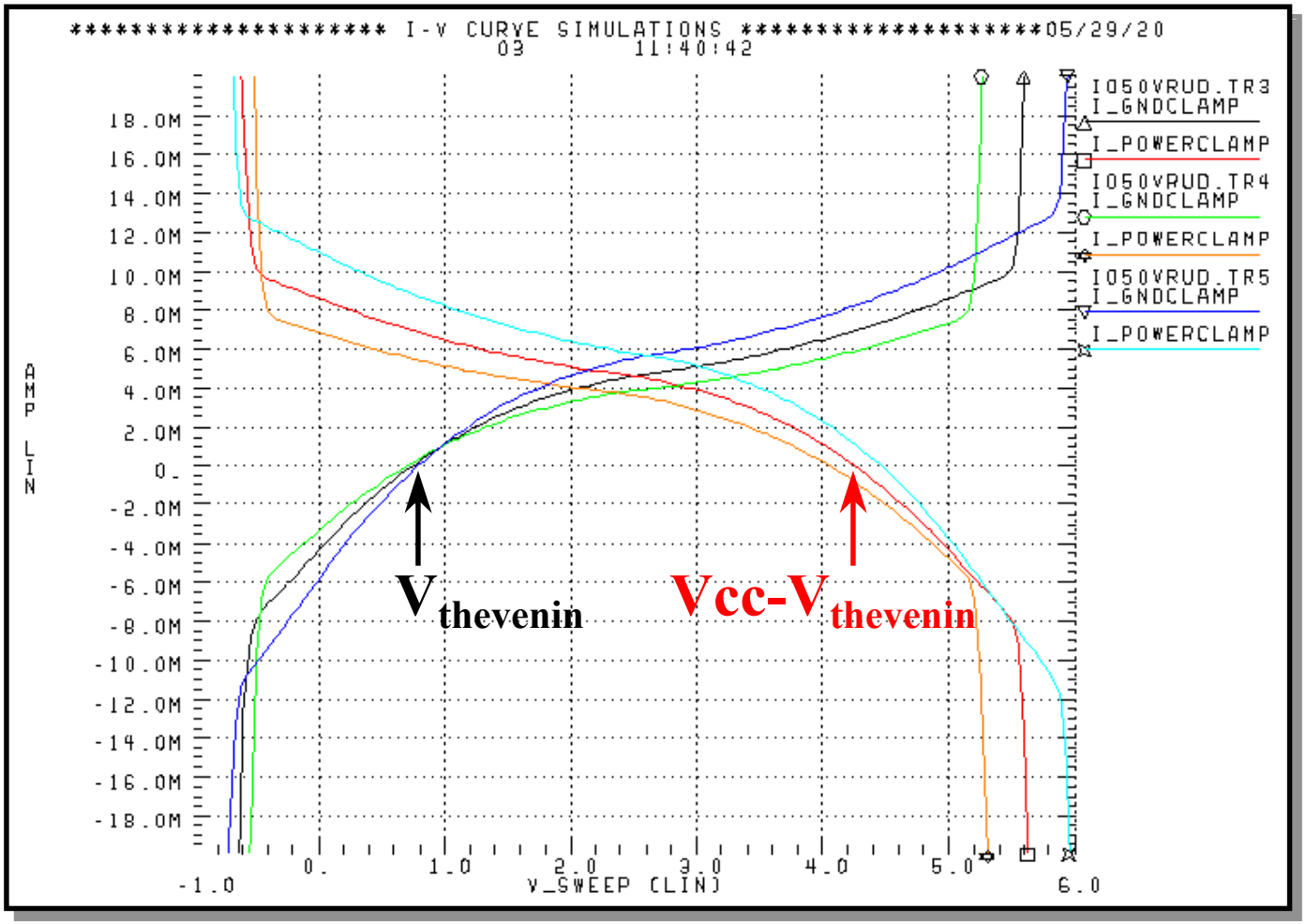
- Sweep device from $-V_{cc}$ to $2*V_{cc}$ twice: GND and V_{cc} relative
- Cut clamp curve which will include the resistor at V_{cc}
  - This can be automated by detecting which group of IV curves goes through the origin
- Cut other clamp curve at 0V
- Normalize (shift) the clamp curve which will not include the resistor to zero current at 0V
- Extrapolate both clamp curves horizontally to $2*V_{cc}$

Pullup and pulldown resistor example



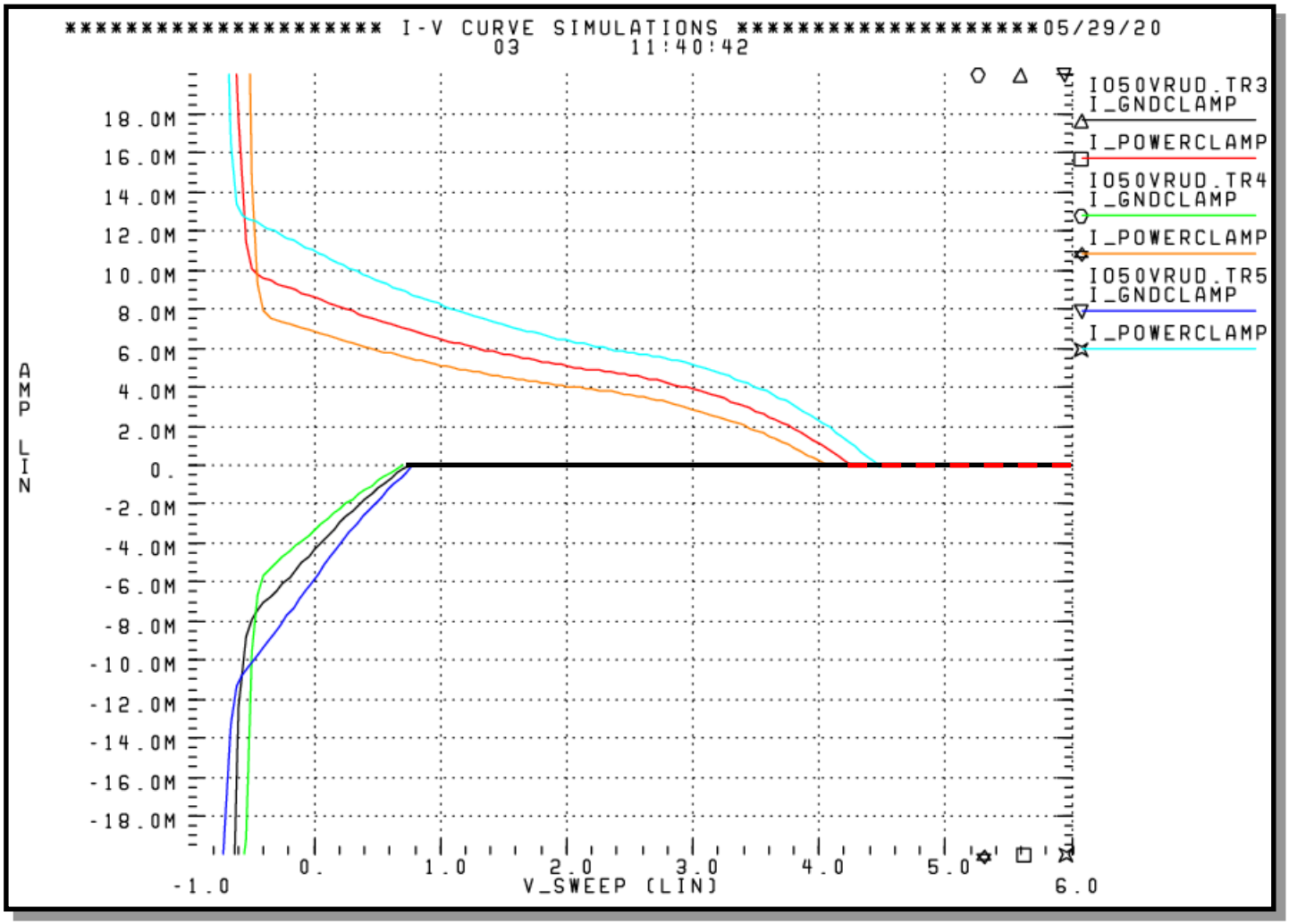
- Looking into the output pad we see $R_{thevenin}$
- It is not possible to separate $R_{thevenin}$ into R_{pu} and R_{pd} from a single measurement at the pad
- The algorithm described on the following pages is only a crude approximation, but it may be better than leaving everything in one IV curve
  - Useful for POWER and GND bounce simulations

IV curves of pu and pd R example



I-V curves of a 3-stated buffer with both pu and pd R

Algorithm in pictures



Algorithm in words

- Sweep device from $-V_{cc}$ to $2*V_{cc}$ twice: GND and V_{cc} relative
- Cut clamp curves where they reach zero current going left to right
- Extrapolate all clamp curves horizontally to $2*V_{cc}$

Lab #3a - buffer with pullup R

- **Look at the IO_buf_R.INC file and notice the change**
  - there is a P-channel MOSFET element called MpR who's gate is connected to ground (it is always on)
- **Run HSPICE with the IO50V_R.SP file**
- **Read the IO50V_R.LIS file into MI (or IBIS Center)**
- **Make a .BUF (or.MR1) file from it**
  - you may get a warning about having both pu and pd resistors in the buffer
  - you have to edit the .LIS file and make sure that the clamp columns of the I-V curves have zeroes at 0 volt, or make R_detection less sensitive
- **Display the I-V curves of both the .LIS and .BUF (or .MR1) files and observe the differences**
  - you should see the GND clamp curve shifted up to 0 amp
  - the GND clamp curve should be flat above 0 volt
  - the POWER clamp should be extended horizontally above the power supply voltage (5 V)

Lab #3b – OD buffer with on-die term.

- **Look at the OD_buf_R.INC file and notice the change**
  - the P-channel MOSFET is turned on all the time (on-die terminator)
- **Run HSPICE with the OD33V_R.SP file**
- **Read the OD33V_R.LIS file into MI (or IBIS Center)**
- **Make a .BUF (or .MR1) file from it**
  - you may get a warning about having both pu and pd resistors in the buffer
  - you have to edit the .LIS file and make sure that the clamp columns of the I-V curves have zeroes at 0 volt, or make R_detection less sensitive
- **Display the I-V curves of both the .LIS and .BUF (or .MR1) files and observe the differences**
  - the pulldown I-V curve should go through the origin in the .BUF (or .MR1) file
  - you should see the GND clamp curve shifted up to 0 amp
  - the GND clamp curve should be flat above 0 volt
  - the POWER clamp should be extended horizontally above the power supply voltage (3.3 V)

Switched parallel termination example

- **The buffer is a normal CMOS driver, but its pullup is ON in receive mode acting as a parallel terminator**

```
|*****************************************************************************
|
[Add Submodel]
| Submodel name          Mode
ParTerm                  Non-Driving
|
|*****************************************************************************
|
[Submodel]      ParTerm
Submodel_type   Dynamic_clamp
|
|*****************************************************************************
|
[POWER Clamp]
|
|       Voltage             I(typ)              I(min)              I(max)
|
     -1.79999995E+0     14.23263550E-3     17.10075140E-3     12.31312752E-3
|    . . .
|    . . .
|   The I-V curve table of the [Pullup] is repeated here, because the
|   terminator is actually the pullup left on in receive mode.
|    . . .
|    . . .
      3.59999990E+0    -44.34032738E-3    -44.32120919E-3    -48.62782359E-3
|
|*****************************************************************************
```

Quasi differential buffer example

- **Make a regular CMOS model for both pins**
  - simple differential buffers consist of two regular CMOS drivers (hence the name “quasi differential”)
  - since differential input currents are usually minimal, defining the clamp currents and differential threshold is sufficient for most differential receivers
- **Associate the drivers and/or receivers as a differential pair using the [Diff Pin] keyword**
  - use the inv_pin, vdiff, tdelay_xxx subparameters to define polarity, differential threshold and differential skews

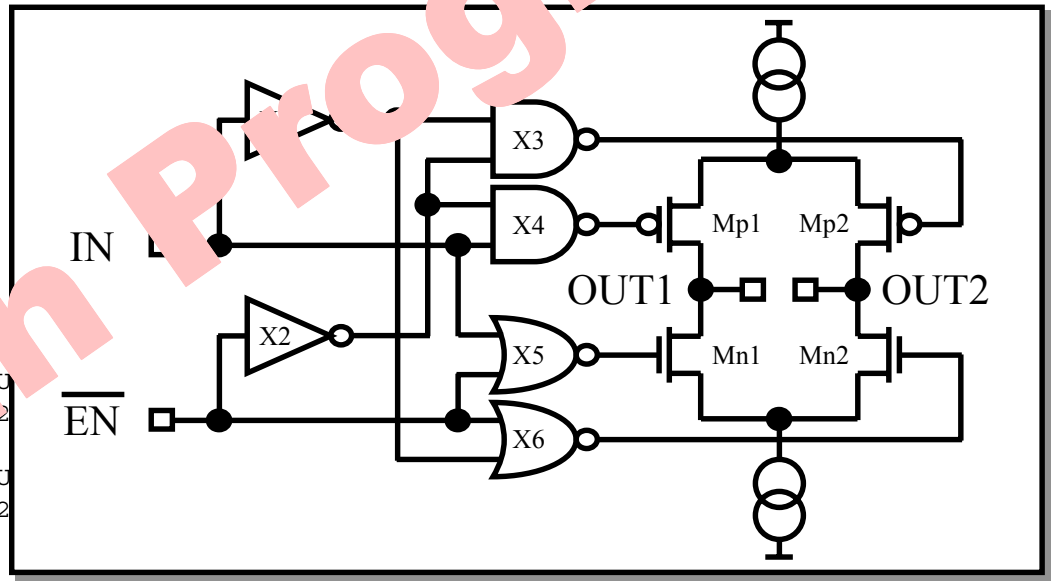
```
[Diff Pin]  inv_pin  vdiff  tdelay_typ tdelay_min tdelay_max
|
 3           4       150mV    -1ns       0ns       -2ns   | Input or I/O pair
 7           8         0V      1ns        NA         NA   | Output* pin pair
 9          10         NA       NA        NA         NA   | Output* pin pair
16          15       200mV     1ns     | Input or I/O pin pair
20          19         0V       NA     | Output* pin pair, tdelay = 0
22          21         NA       NA     | Output*, tdelay = 0
                                       | * Could be Input or I/O with vdiff = 0
```

Lab #4a - quasi differential buffer

- **Open the IBIS file you made in Lab #1 with a text editor and add a differential driver and/or receiver to it manually**
  - add two or more pins to the pin list under the [Pin] keyword
  - fill out the pin number, signal and buffer name columns
  - add a new keyword [Diff Pin]
  - fill in the headers next to the keyword
  - type the pin numbers of those additional pins you entered under the [Pin] keyword
  - fill in the values for the subparameters (use NAs if appropriate)
- **Generate a new IBIS file using this .PIN file with MI (or IBIS Center)**

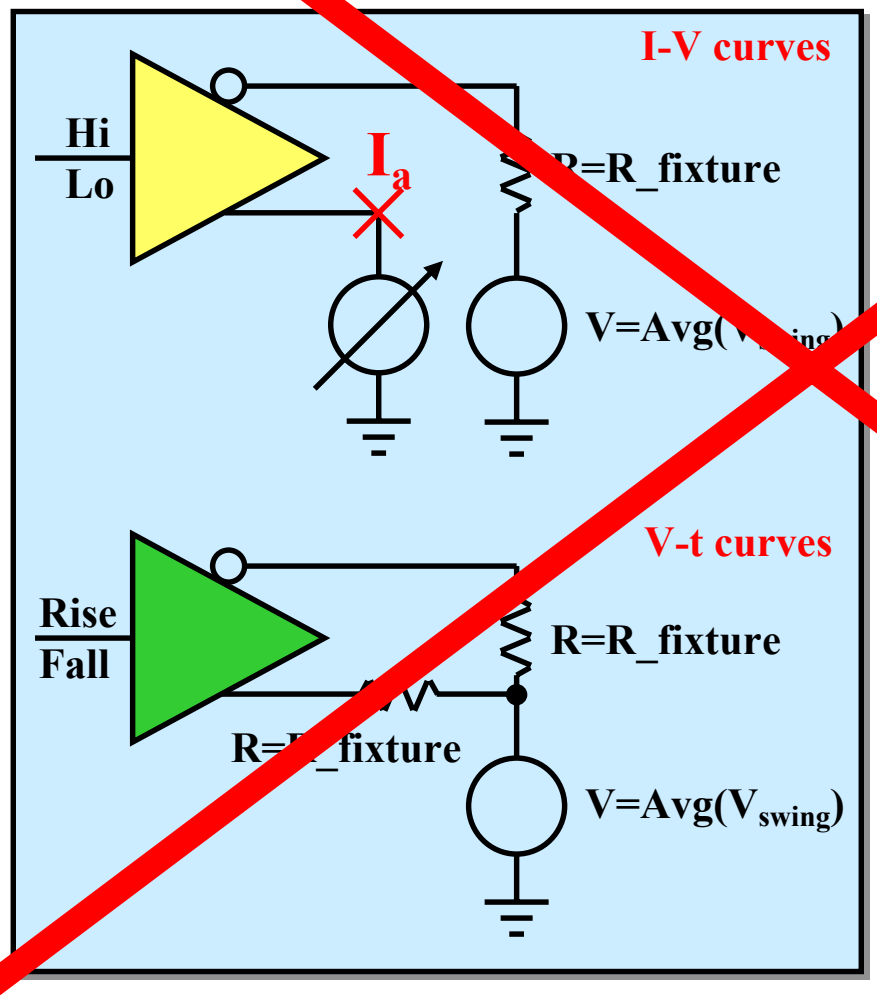
True differential buffer templates

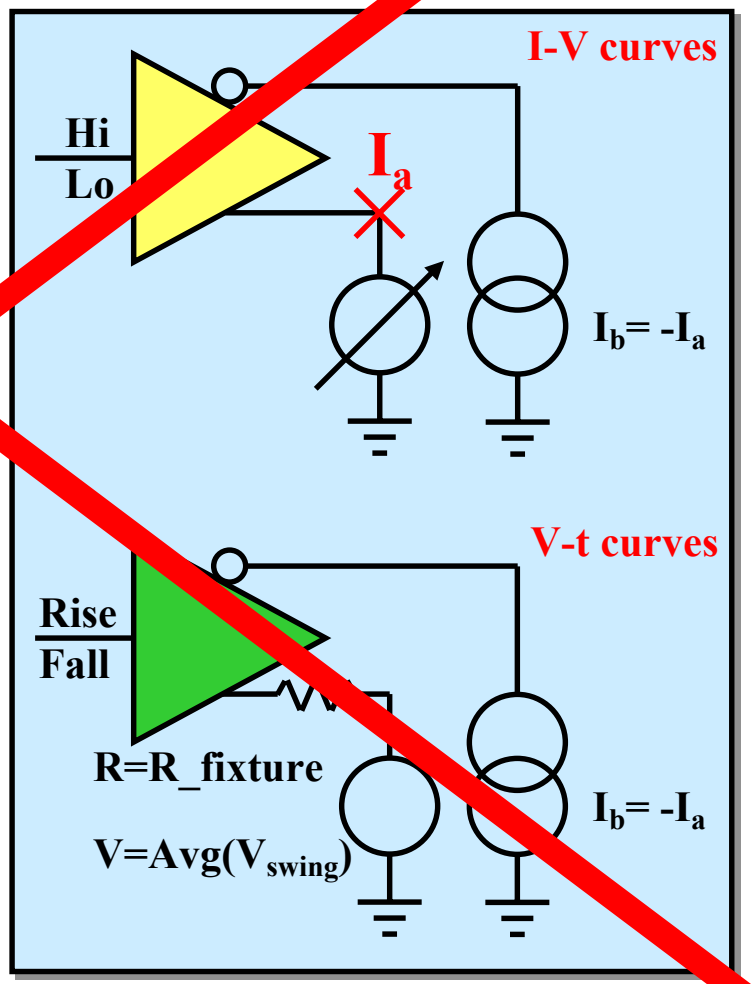
```
*******************************************************************************
.SUBCKT IO_buf input output power ground enable
*******************************************************************************
X1  input   in_b    power  ground  INV_IDEAL
X2  enable  en_b    power  ground  INV_IDEAL
*
X3  in_b    en_b    gt_p2  power  ground  NAND2  mult_p=  24  mult_n=  12
X4  input   en_b    gt_p1  power  ground  NAND2  mult_p=  24  mult_n=  12
X5  input   enable  gt_n1  power  ground  NOR2   mult_p=   8  mult_n=   5
X6  in_b    enable  gt_n2  power  ground  NOR2   mult_p=   8  mult_n=
*
Gt  power  top  VCVS  PWL(1)  power  top
+  -5      '-Itop'
+  -0.5    '-Itop'
+   0.5    Itop
+   5      'Itop+(4.5/Rtop)'
Gb  bot  ground  VCCS  PWL(1)  bot  ground
+  -5      '-Ibot'
+  -0.5    '-Ibot'
+   0.5    Ibot
+   5      'Ibot+(4.5/Rbot)'
*
Mp1 output1 gt_p1 top power  PMOS  L=0.   0U
+                            AS=  34.0P A   2
+                            M=
Mn1 output1 gt_n1 bot grou a          =0.800U
+                            A   434.0P AD=2
+                            M=6
*
Mp2 output2 g   2   p   wer  PMOS  L=0.800U W=43.40U NRD=0.0897 NRS=0.0737
+                            AS=434.0P AD=217.0P PS=106.8U PD=53.40U
+                            M=12
Mn2 out     gt_  bot ground NMOS  L=0.800U W=43.40U NRD=0.0897 NRS=0.0714
+                            AS=434.0P AD=217.0P PS=106.8U PD=53.40U
+                            M=6
```



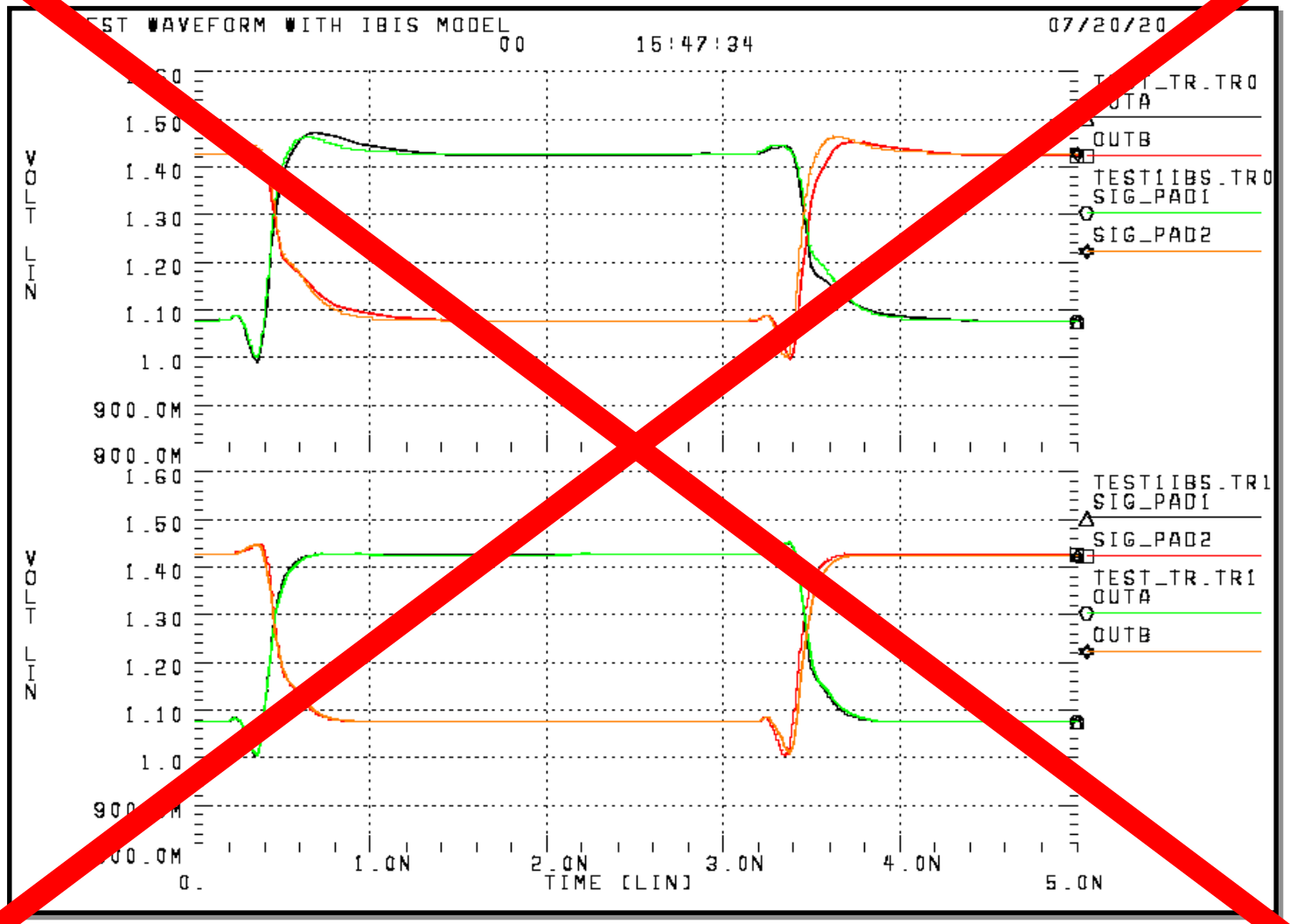
True differential buffers (LVDS)

IBIS doesn't support them directly but they can be approximated with two different methods



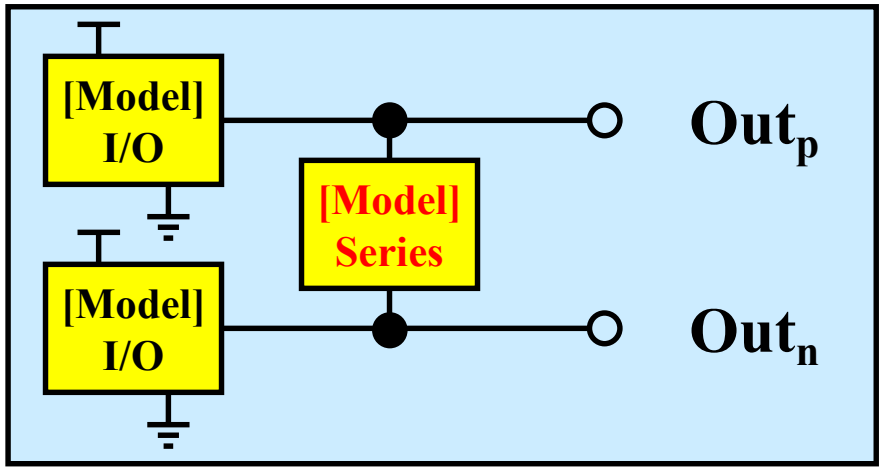


LVDS SPICE and IBIS model overlay

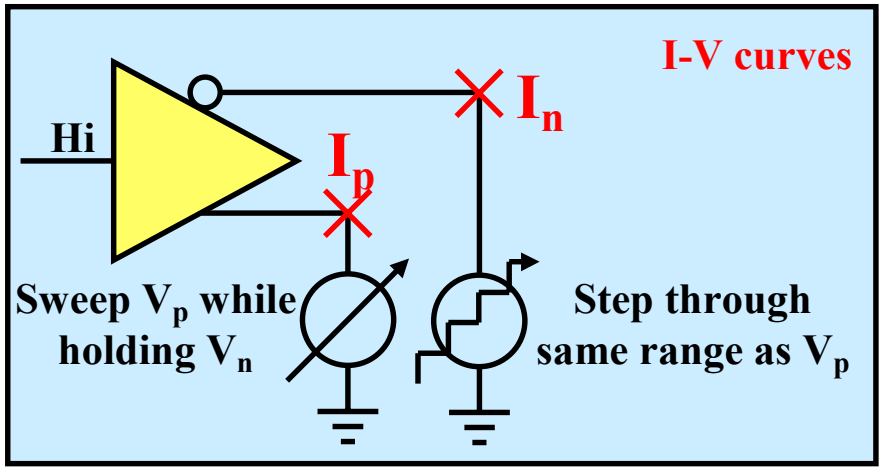


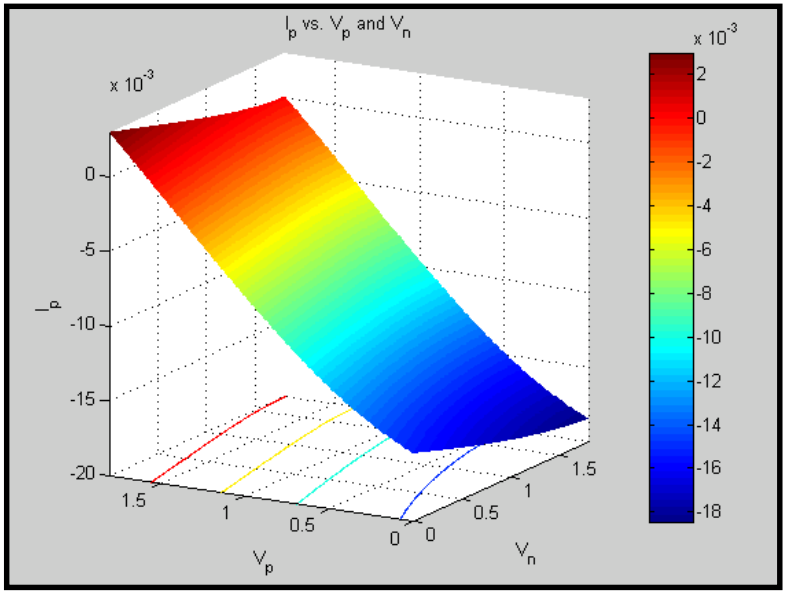
Keywords for true differential buffers

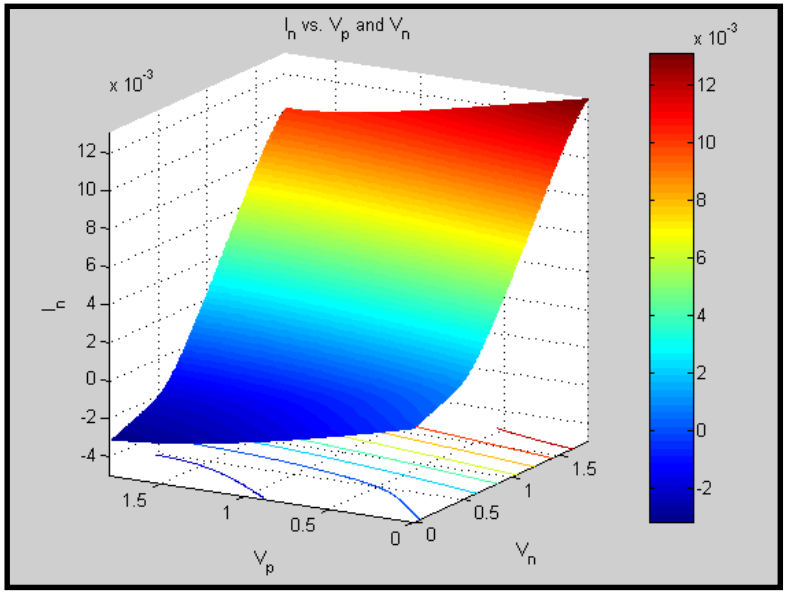
- **[Series Pin Mapping]**
  - allows the mapping of series elements, such as R, L, C and current tables to pins which *already have* a [Model] assignment
  - a series current (and/or) R, C elements could be used to account for the *differential currents*
  - the regular [Model] associated with the same two pins could be used to account for the *common currents*
- **[Series Current]**
- **[Series MOSFET]**
- **[R Series]**
- **[C Series]**
- **[Rc Series]**



IV curve measurements







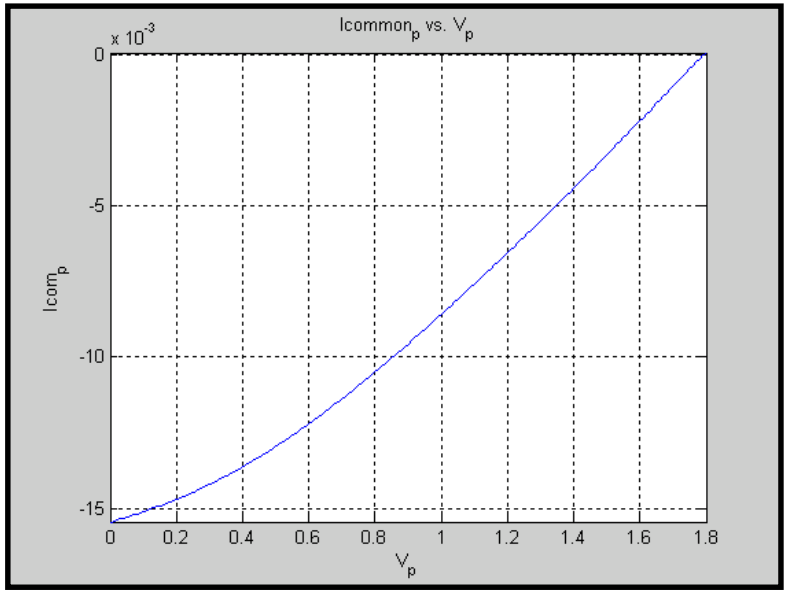
Separating the common mode current

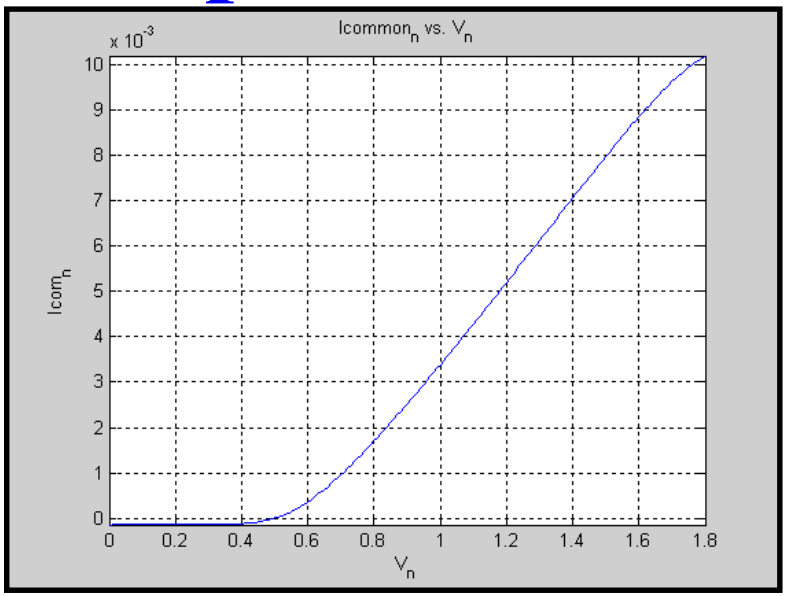
- **Assume that $I_{diff} = 0$ when $V_{out\_p} = V_{out\_n}$**
  - I.e. no energy is stored, and/or we sweep slowly enough, so that the stored energy is not visible even if it is there
- **$V_{out\_p} = V_{out\_n}$ is along the diagonal of the IV surface plot**
- **The current values along the diagonal represent the common mode current of the differential buffer**
  - Extract the current along the diagonal and use it for the common mode [Model]'s pullup and/or pulldown IV curve(s)

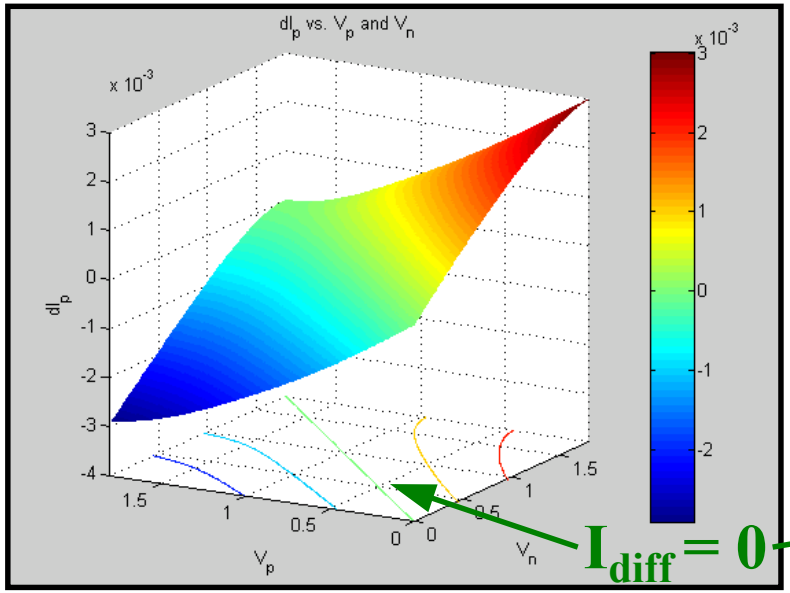
Separating the differential mode current

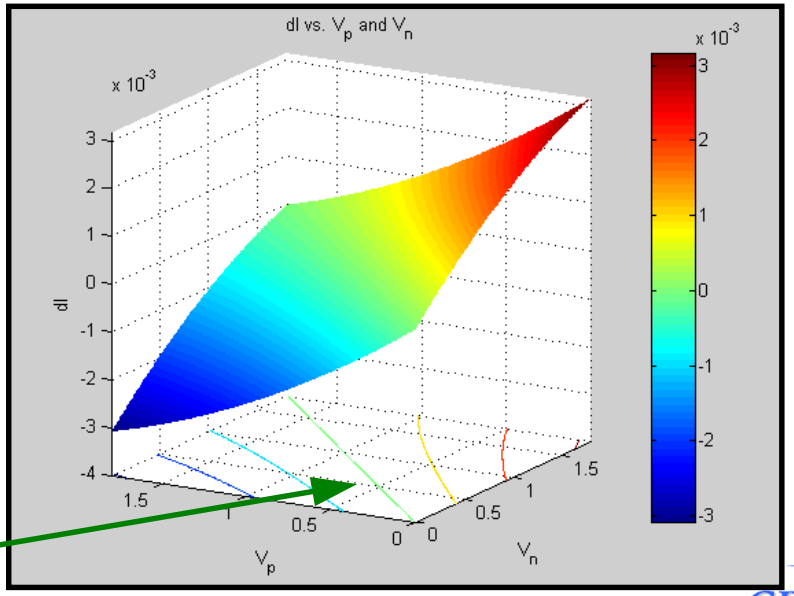
- **The off-diagonal current values represent the sum of the common and differential mode currents**
- **To obtain the differential mode currents alone, "normalize" the surface so that its diagonal values become zero**
  - Subtract the common mode component from the surface and use it for the Series [Model]'s [R Series], [Series Current], [Series MOSFET], etc… keywords
  - If the surface is linear (flat) [R Series] is sufficient
  - Otherwise use the [Series Current] or [Series MOSFET] keywords
  - Slice the surface along the necessary voltage value(s) to satisfy the syntax requirement of the IBIS keyword used

Results of decomposition



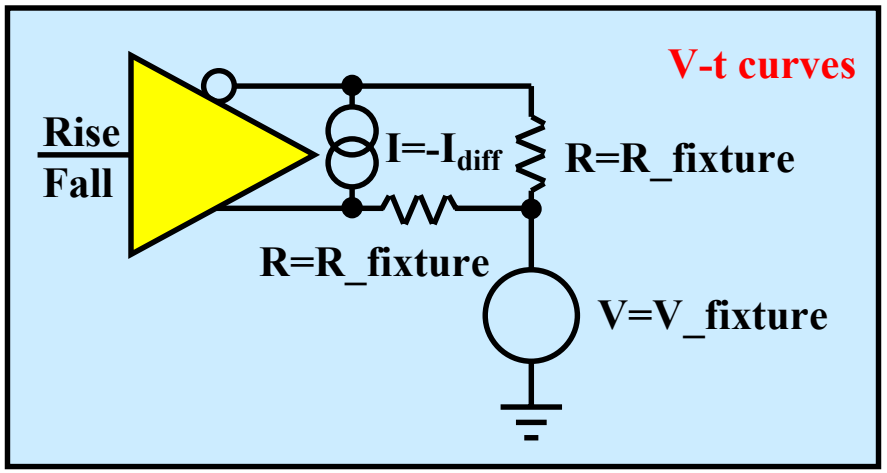






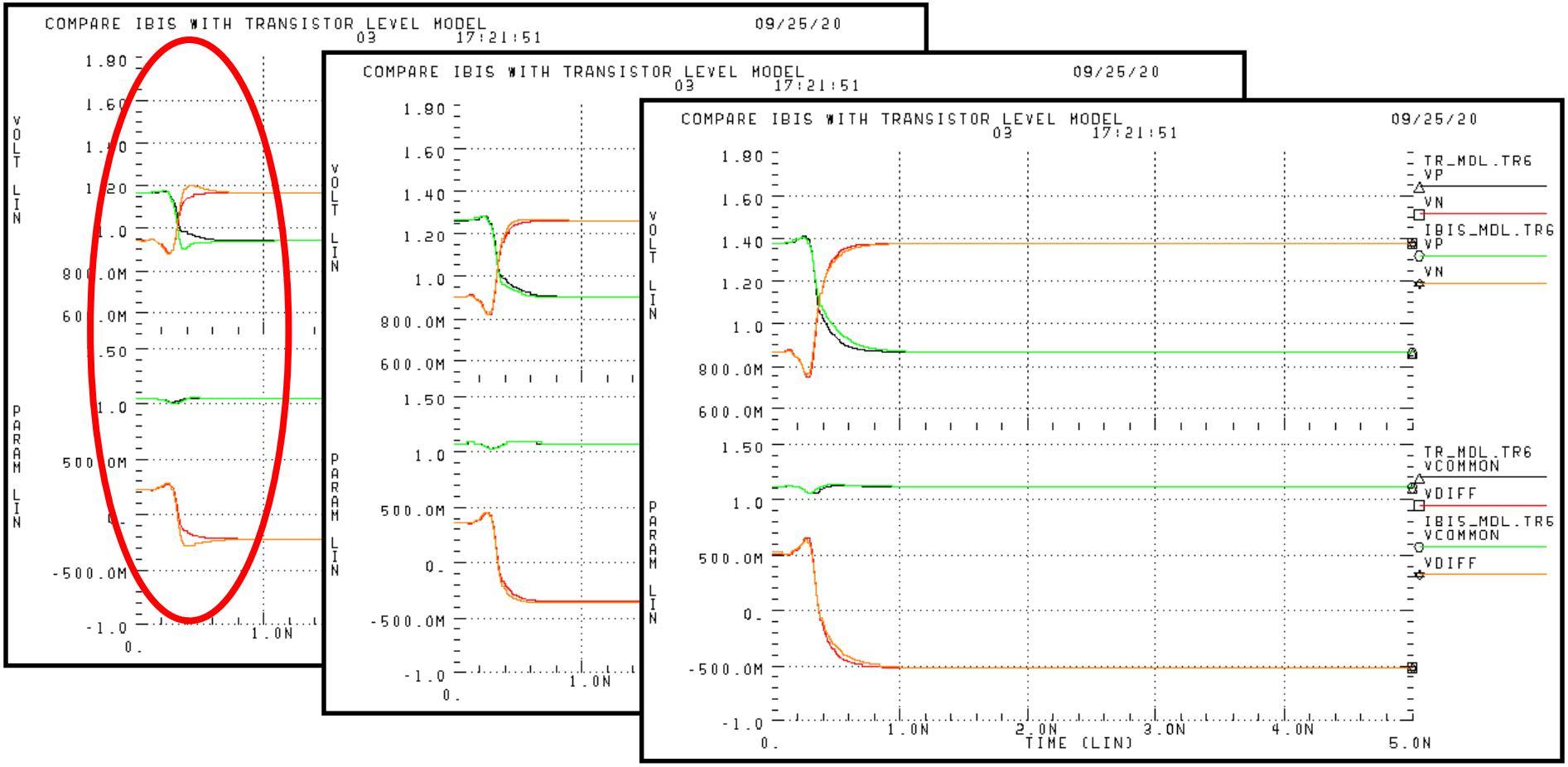
$I_{diff} = 0$

Vt curve measurements

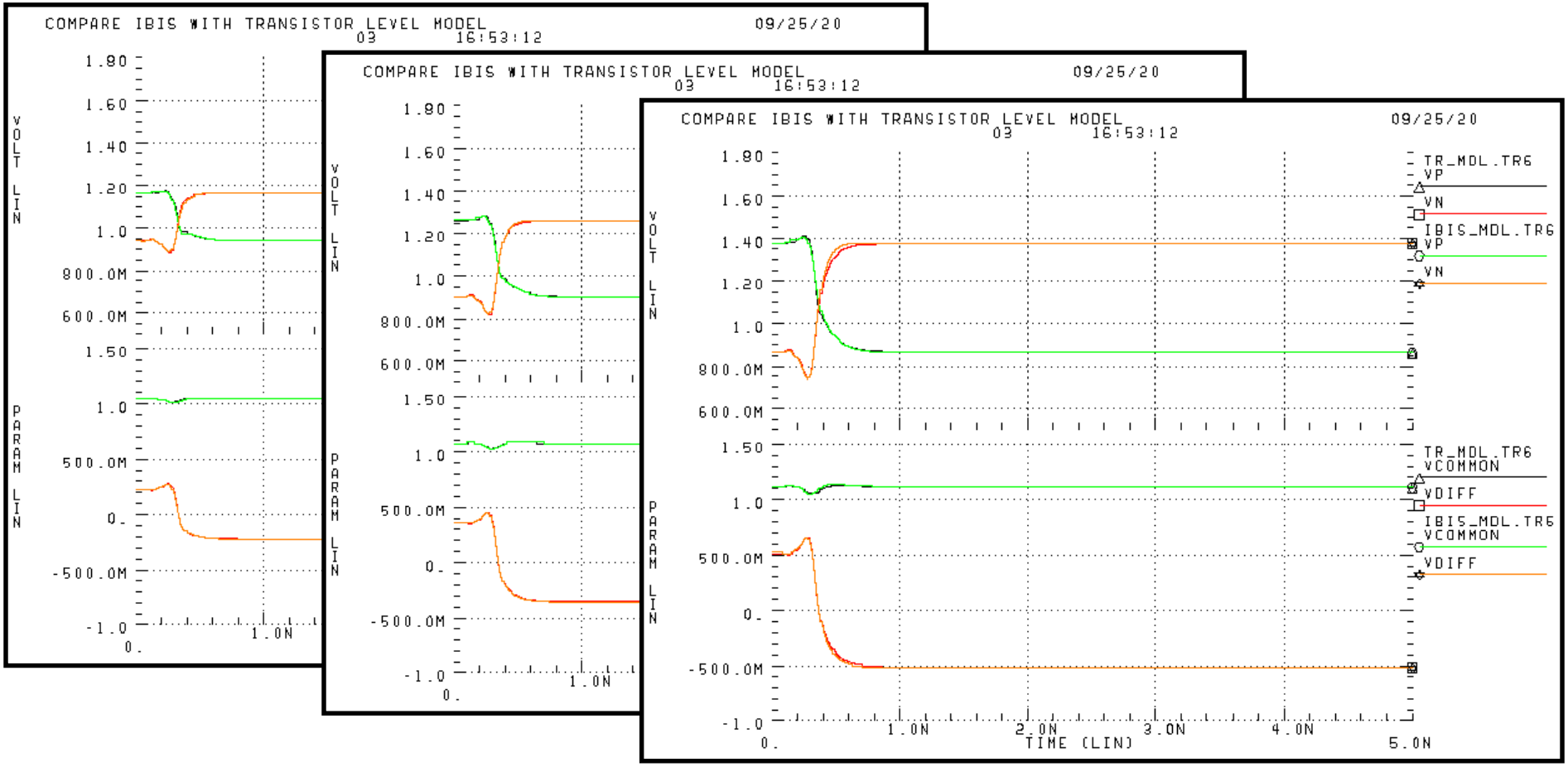


- **Run time domain simulations (.TRAN) using the above circuit**
  - R_fixture should be close to Z_{common} of the T-line the buffer will drive
  - V_fixture should be close to the DC high and low levels of the signal swing the buffer will achieve
  - Use current source to cancel differential current
- **Generate at least four waveforms**
  - One rising and falling waveform with at least two V_fixture values
- **More than four waveform tables per [Model] makes the model more accurate with tools that support it**

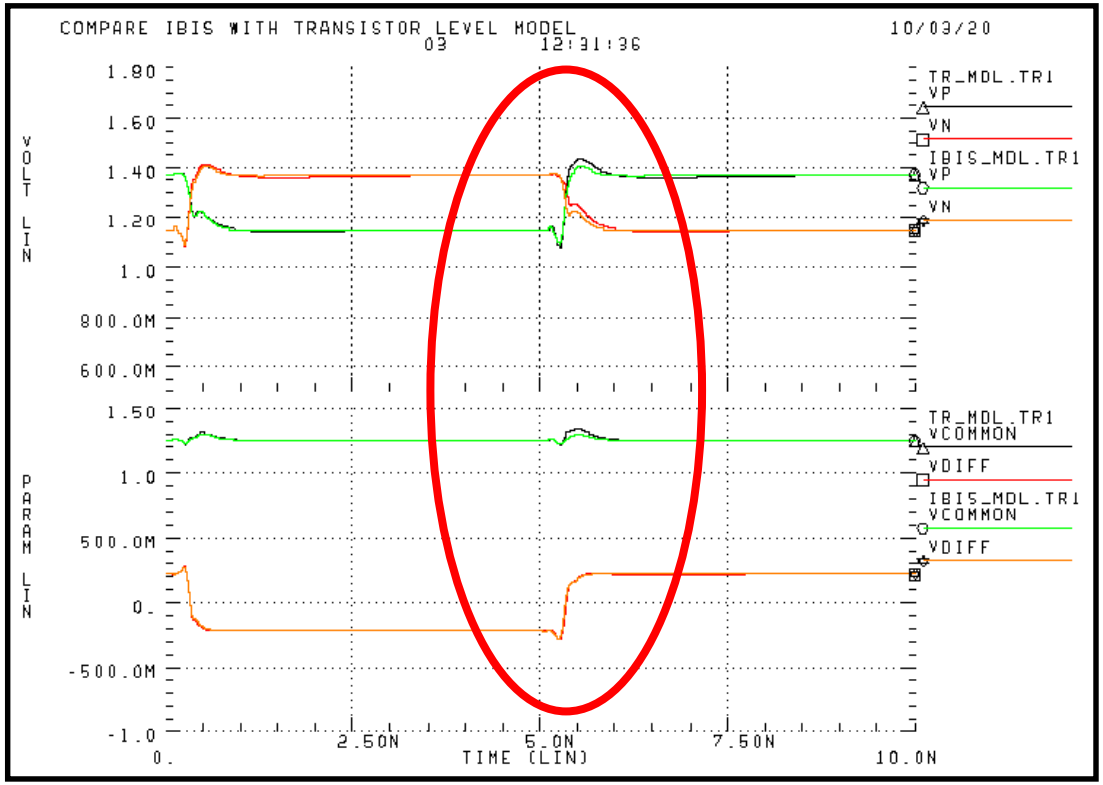
Miscorrelation with incorrect C_comp



Good correlation with correct C_comp

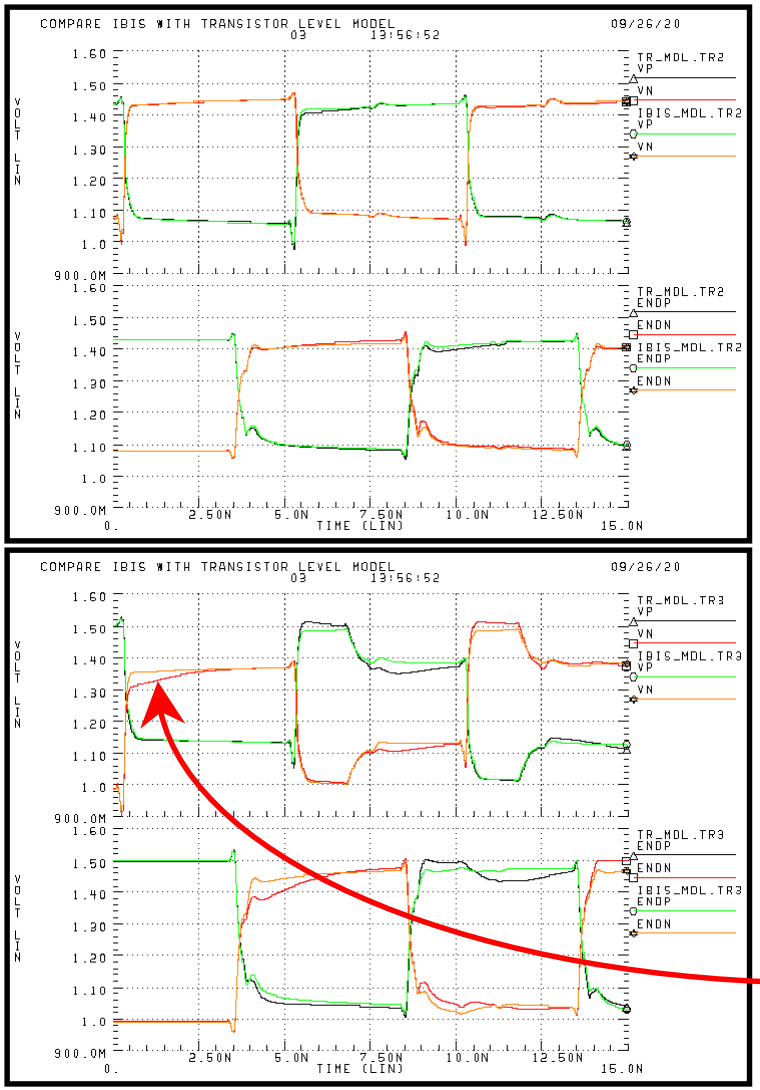


Asymmetry in buffer behavior



- **Every other transition is less accurate**
  - a different and alternating set of waveforms that are switched by the rising and falling edge stimulus may give better results
- **IBIS 3.2 cannot describe data dependent models**

Higher order effects need more study



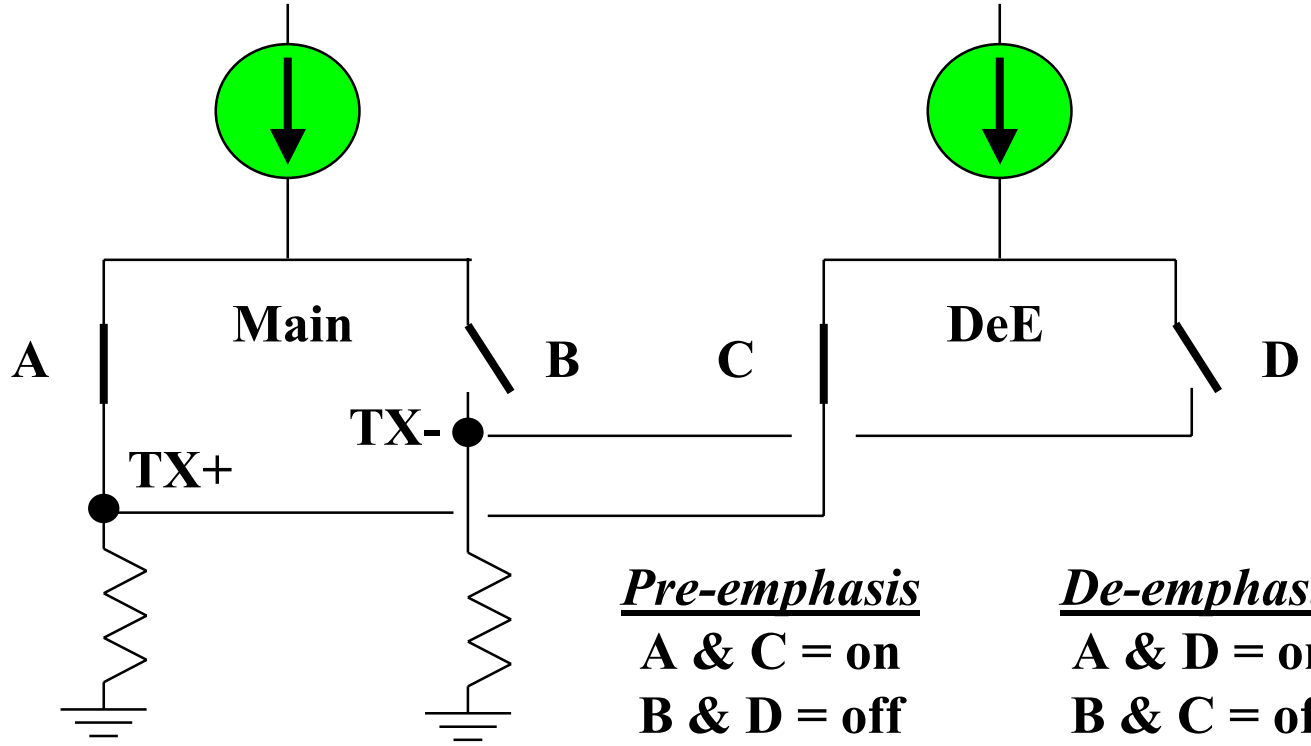
- **T-line with matched termination yields excellent results**
- **Higher order effects of the buffer's transient behavior becomes more visible when the T-line has badly mismatched termination**

Lab #4b - true differential buffer

- **Pretend that the io33v.lis file contains the common mode components of a differential buffer**
- **Pretend that the differential components can be described by a simple RC circuit**
  - add two or more pins to the pin list under the [Pin] keyword
  - fill out the pin number, signal and buffer name columns
  - add a new line under the keyword [Diff Pin] for these two new pins
  - add a [Series Pin Mapping] keyword using these two new pin names
  - create a new [Model] (.BUF or MR1 file) of type Series by hand and use the [R Series] and [C Series] keywords to describe the differential characteristics of the buffer
    - use 500 Ω and 500 fF for typical their values, you can use NA for min., and max.
  - fill in the values for the subparameters (use NAs if appropriate)
- **Generate a new IBIS file using this .PIN file with MI (or IBIS Center)**

Pre/De-emphasis buffer

- Pre-emphasis sums both sources through one “leg”
- De-emphasis “steals” current from non-driving leg
- Total current in system always the same



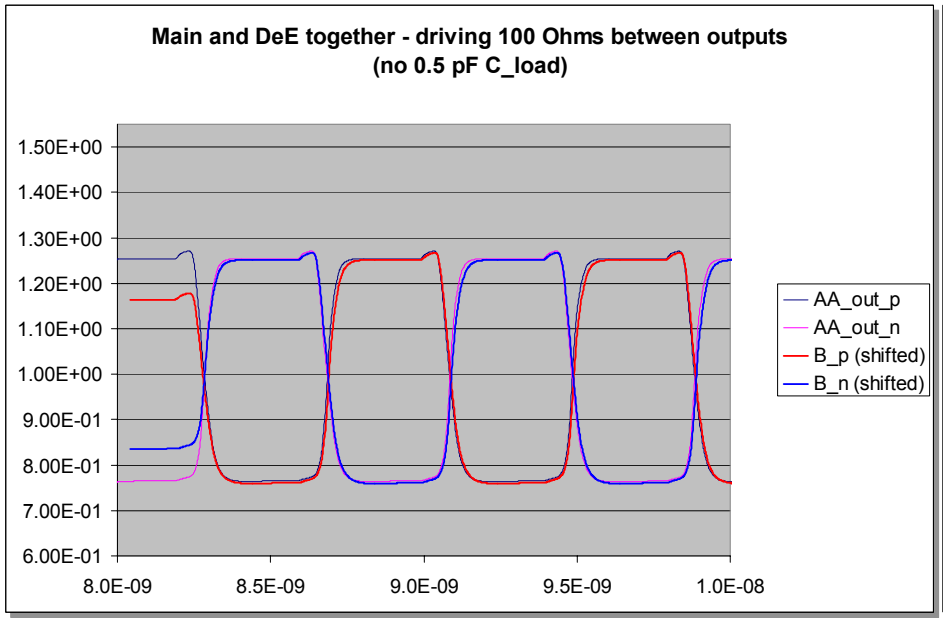
Pre-emphasis
A & C = on
B & D = off

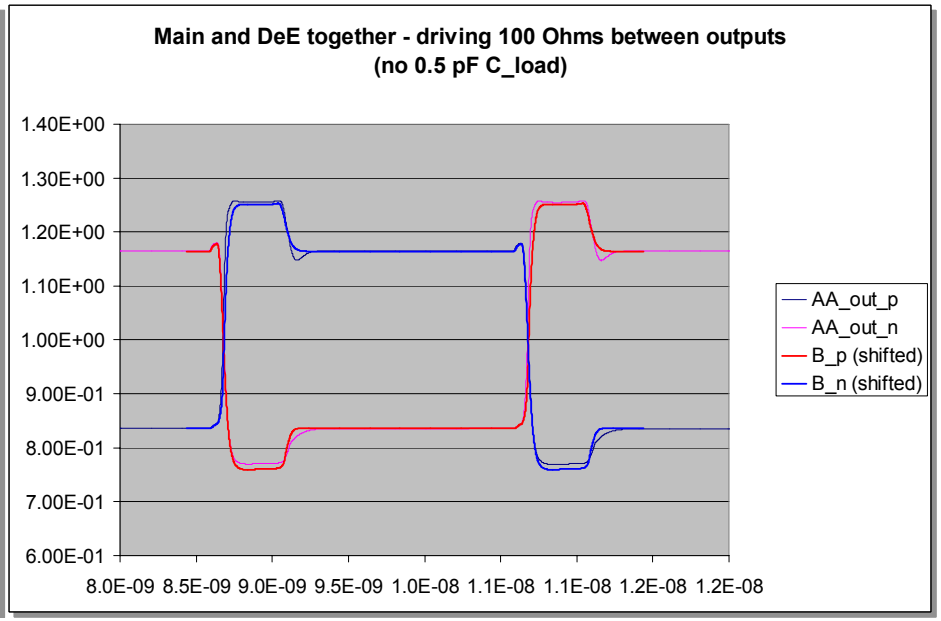
De-emphasis
A & D = on
B & C = off

Pre/De-emphasis buffers (PCI-Express)

- **Separate the two halves of the buffer in SPICE**
  - make a “Main” buffer from the SPICE netlist
  - make a “De-Emphasis” buffer from the SPICE netlist
- **Make a common mode models for each of these separately**
  - sweep IV curves with two sources driving the same voltage into both pins
- **If necessary make IV table for the [Series Current] keyword, or calculate the value of [R Series]**
  - if you use these keywords edit the IBIS file manually
- **Add [Diff Pin] keyword in IBIS file**
  - this is another manual edit in the IBIS file
- **Write comments to user of IBIS file to connect the “Main” and “DeE” buffers and drive them with a one bit delayed and inverted signal**

Pre/De-emphasis buffer correlation





Continuous strong bits correlate well

Alternating strong and weak bits don't correlate well due to data dependent modulation of current source (Miller effect)

Lab #5 - programmable buffer

- **Create a new IBIS model with a programmable buffer**
  - make a copy of one of the SPICE I/O or output models (.INC file) with a different file name (for example: IO_2x.inc)
  - edit its netlist to make a different strength buffer from it
    - (double the M=x multiplier of the output transistors from 12 and 6 to 24 and 12)
  - run HSPICE to generate a new .LIS file and convert it to a .BUF (or .MR1) file with MI (or IBIS Center) as before
  - add another pin to the pin list under the [Pin] keyword
  - put “progbuf1” (or anything you like) for its buffer name
  - add the keyword [Model Selector] after the [Pin] section
  - give it the same name you used in the pin list above (progbuf1)
  - type a list of model names (.BUF or .MR1 file names) under the [Model Selector] keyword with comments on their right
    - (for example: IO33v and IO33v2x)
  - create a new IBIS file using MI (or IBIS Center) using this .PIN file