Details on True Differential Buffer Characterization Revisited

IBIS Summit at PCB Conference East

Westford, MA

October 21, 2003

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

Background (1)

- **The author of this presentation introduced a page on true differential buffers in his "Introduction to IBIS Models and IBIS Model Making" class materials in September 2000**
  - a proprietary LVDS buffer was used to make recommendations for generating IV and Vt curves to convert it to an IBIS model
  - this buffer did not have an on-die differential termination (pad-to-pad)
- **Hazem Hegazy gave two presentations on this subject in January and March 2001 at IBIS summits**
  - quoting the above class material, some shortcomings of the suggested technique were pointed out and explained by Hazem
    - models of buffers having differential (pad-to-pad) on-die termination made the way it was suggested in the class material are inaccurate under loading conditions that are different from the loading conditions used for making the model
  - new techniques were demonstrated giving better correlation with the original SPICE model, Proposal-III being the best and most accurate

Background (2)

- **The author of this presentation introduced a new, improved technique in the PCB East IBIS Summit October 2002**
  - the common and differential current components were obtained using a nested sweep and surface plots
  - suggestions were made that the data generated this way could be used with existing IBIS 3.2 keywords
  - no verification was done to prove the concept
- **This presentation closes the loop and shows an actual implementation of the above technique with waveform overlays**
  - this proves that the technique works and provides IBIS models which yield correct DC levels under all loading conditions
  - some higher order effects will be shown which need more work to describe the transient behavior of the buffers more accurately

Further study and work needed (from 2002)

- ✓ **This concept needs to be proven with examples**
  - this could be done in the near future
- ✓ **Need to develop a differential C_meter to measure capacitive coupling between pins**
  - [C Series] can be used to hold this value in the IBIS model
- ➢ **More experiments need to be done to find out how the differential current varies with respect to time during transitions from one state to another**
  - do we need Vt curves for these differential elements also?
- ➢ **Simulation tool vendors should implement the series element features in IBIS v3.2 ASAP!**
  - unfortunately not all tools support these v3.2 keywords yet

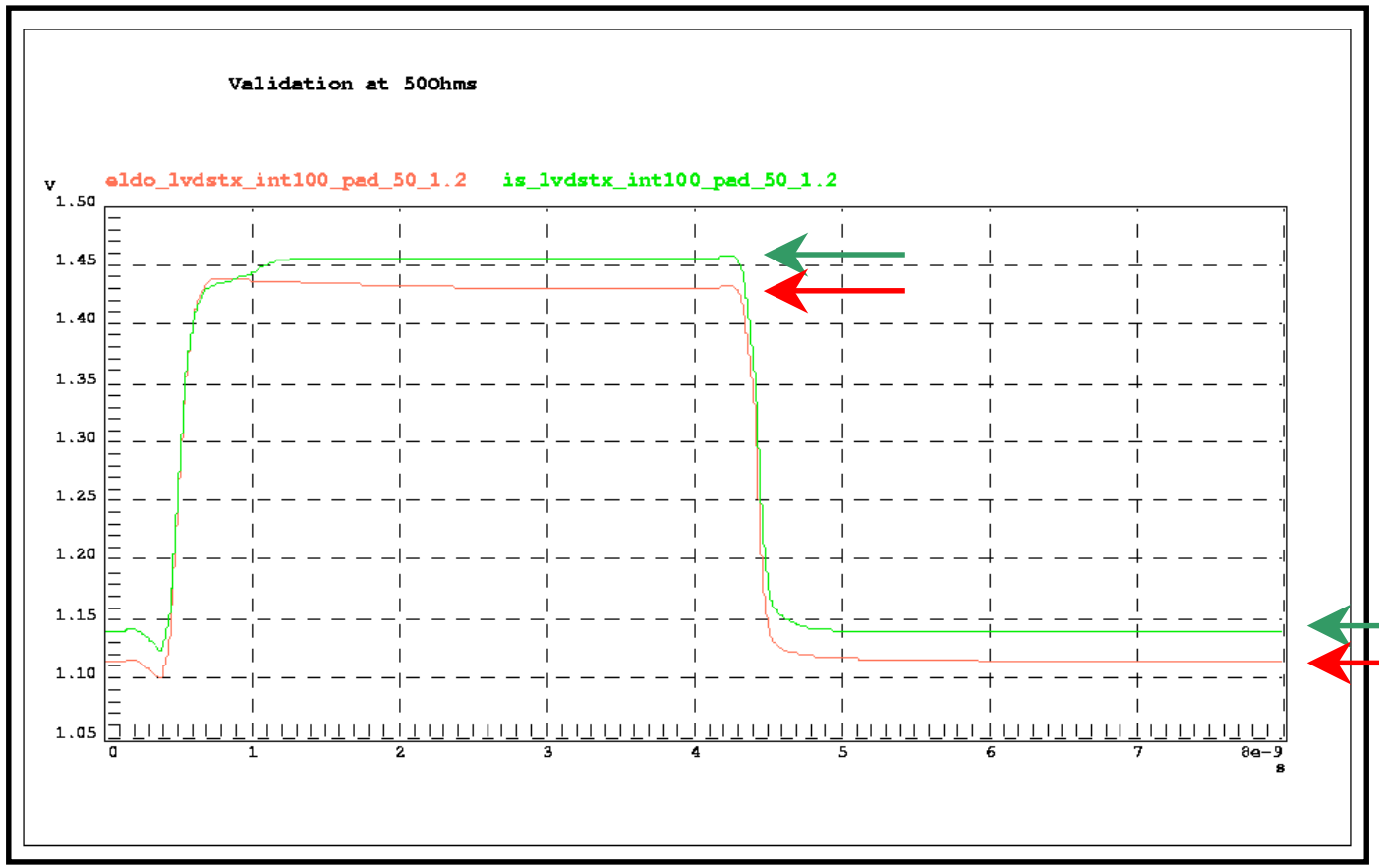
Sadly this is still true!!

Problem statement

- **Just like transmission lines, true differential buffers have common and differential mode impedance components**
  - Common mode current flows between the pad and the supply rails
  - Differential mode current flows between the two pads
- **Traditional IBIS modeling treats differential buffers as two independent [Model]s driven by a stimulus and its complement**
  - This approach does not describe the coupling effects between pads
  - It may work under certain circumstances, but it is inaccurate in general, causing DC shifts in the signal levels

DC error illustration

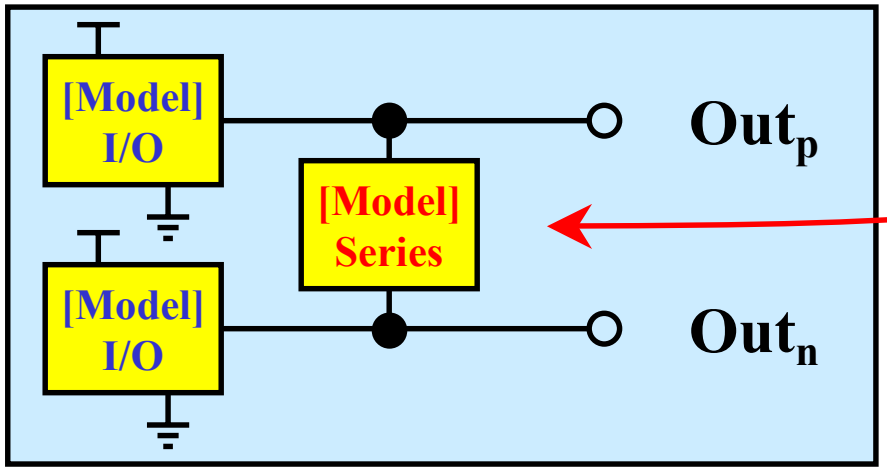
- **Two independent single ended [Model]s yield incorrect DC levels when simulated with loads that are different from V_fixture and R_fixture in the [Model]**



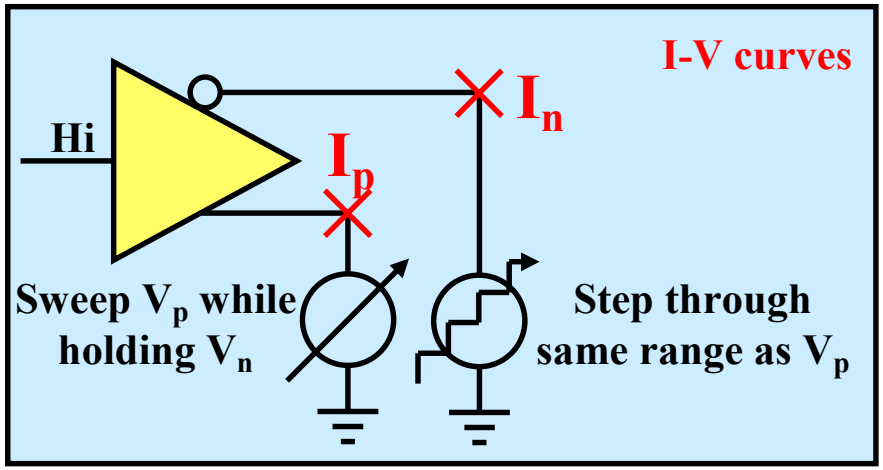
* H. Hegazy and M. Korany "LVDS Modeling", IBIS Summit, March 2001, p. 14.

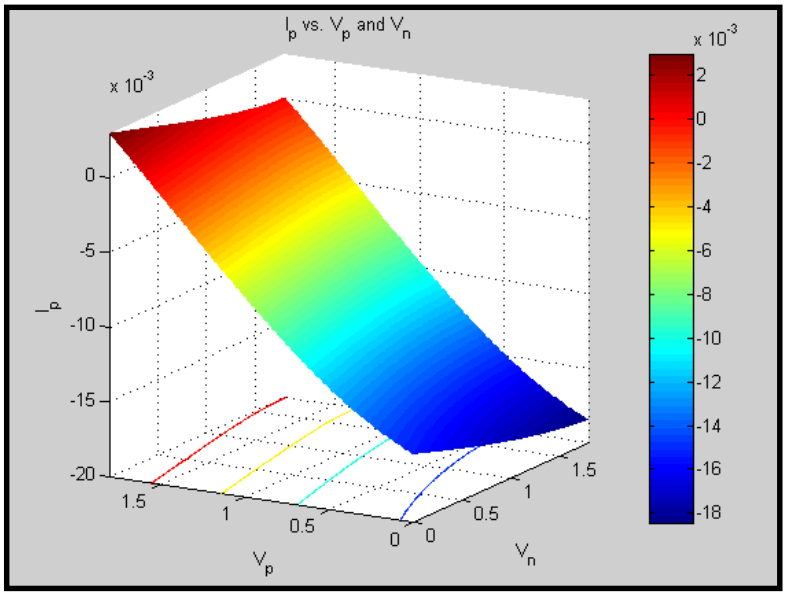
Improved IBIS modeling approach (using only v3.2 keywords)

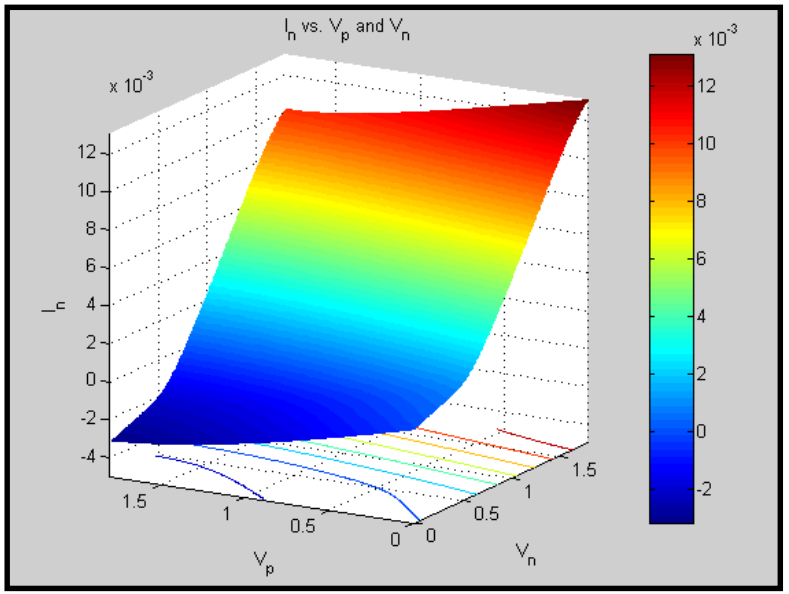
- **Common mode (pad-to-supply rail) characteristics can be described by the regular [Model] keywords as before**
- **Differential mode (pad-to-pad) characteristics can be described with an additional [Model] of type Series placed between the pads of the two normal single ended [Model]s**
  - Use the [Series Pin Mapping] keyword to map the Series [Model] to the appropriate pads
- **Inside the Series [Model] use any of the series keywords**
  - [R Series], [Series Current], [Series MOSFET], etc…



Illustrating measurement setup and results







HSPICE implementation of nested sweep for a 1.8 V differential buffer

```
LVDS differential buffer I-V curve generator
*******************************************************************************
.TRAN  1.0ms  180.0ms  SWEEP  Sw_n  LIN  181  0.0  1.8
.OPTIONS POST=1 PROBE INGOLD CO=132 NUMDGT=8 ACCURATE RMAX=0.2
*******************************************************************************
.param  Sw_n= 1.25
*******************************************************************************
.PRINT TRAN
+ V_p    = V(Sw_p)
+ Isw_p  = I(Vsw_p)
+ Isw_n  = I(Vsw_n)
*******************************************************************************
V_vcc  VCC  0   DC= 1.8
V_in   In   0   DC= 1.8
V_en   En   0   DC= 1.8         $ Enable is active high.  This is drive mode.
*******************************************************************************
Vsw_p  0  Sw_p  PWL
+   0.0ns        0.0
+ 180.0ms       -1.8
*
Vsw_n  0  Sw_n  DC='-1*Sw_n'
*------------------------------------------------------------------------------*
X1  In  Sw_p  Sw_n  Vcc  0  En  BUFFER
*******************************************************************************
.END
*******************************************************************************
```

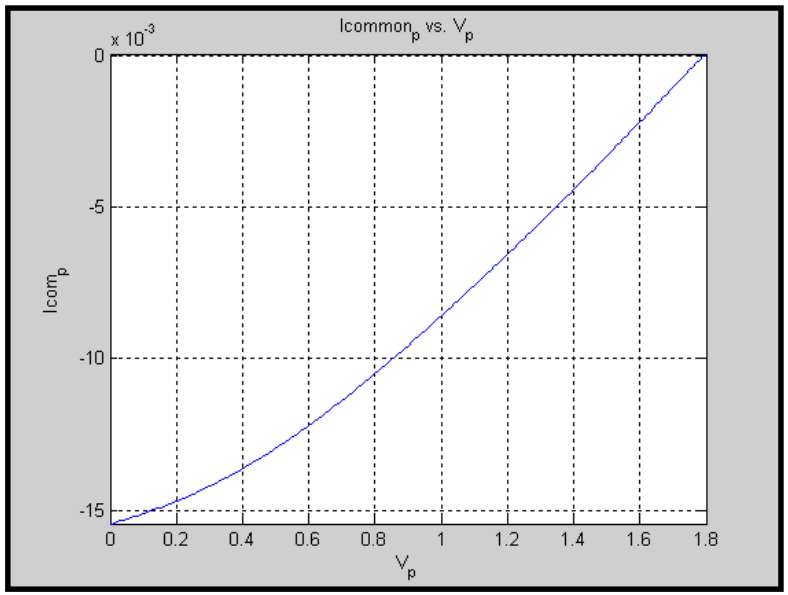
Separating the common mode current

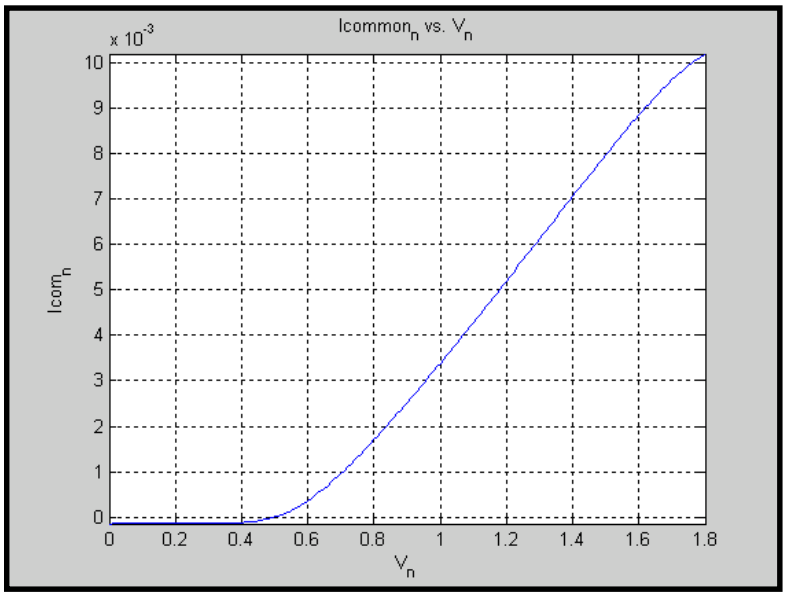
- **Assume that $I_{diff} = 0$ when $V_{out\_p} = V_{out\_n}$**
  - I.e. no energy is stored, and/or we sweep slowly enough, so that the stored energy is not visible even if it is there
- **$V_{out\_p} = V_{out\_n}$ is along the diagonal of the IV surface plot**
- **The current values along the diagonal therefore 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)

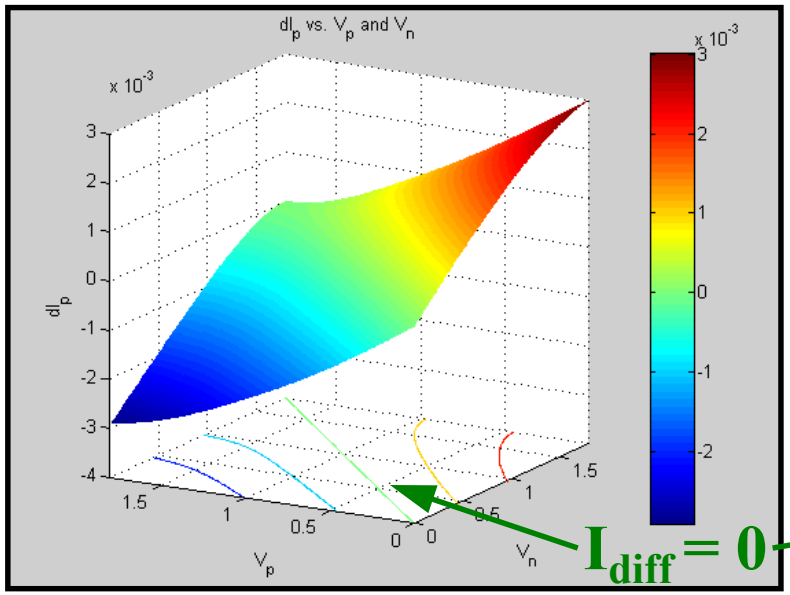
Separating the differential mode current

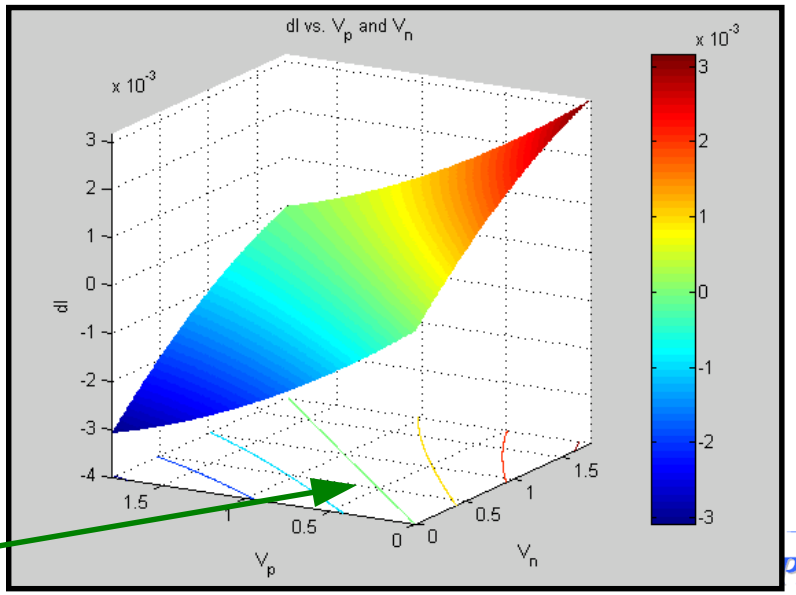
- **The off-diagonal current values represent the sum of the common and differential mode currents**
- **To obtain the differential mode currents alone, "normalize" the surface so that its diagonal values become zero**
  - Subtract the common mode component from the surface and use it for the Series [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

Illustrating the decomposition of common and differential currents









$I_{diff} = 0$

Perl code for extracting data from .LIS file

```
#!/usr/…/bin/perl
#*****************************************************************
if ($#ARGV != 0) {
   print "\nThe number of command line arguments is: ",$#ARGV+1,"\n";
   print "Usage: IVmatrix.pl 'Filename' (without the extension) \n\n";
   goto(Label_END);
   }
$LISfile = "$ARGV[0].lis";                                    # Change extension of file name to "lis"
$Mfile   = "$ARGV[0].m";                                      # Change extension of file name to "lis"
#*****************************************************************
$Bias_count  = 0;
$Row_count   = 0;
$Found_bias  = 0;
$Found_table = 0;
$Line        = 0;
open (INPUT,  "< $LISfile") || die "Can't open $LISfile: $!";
while (<INPUT>) {
   $Line = $Line+1;
   if (($Found_bias == 0) && ($_ =~ /(parameter sw_n)+(\s*=\s*)+(\S*)/i)) { # Find "parameter bias"
      $V_n[$Bias_count] = $3;
      $Found_bias = 1;
      $Bias_count = $Bias_count + 1;  }
   elsif (($Found_bias == 1) && ($Found_table == 0) && ($_ =~ /(time)+(\s*v_p)+(\s*isw_p)+(\s*isw_n)+/i)) {
      $Found_table = 1; }                                     # Find the beginning of the table
   elsif (($Found_table == 1) && ($_ =~ /\s*(\S+)\s*(\S+)\s*(\S+)\s*(\S+)/i)) {
      $V_p[$Row_count] = $2;                                  # Extract numbers from table
      $Isw_p[$Row_count][$Bias_count-1] = $3;
      $Isw_n[$Row_count][$Bias_count-1] = $4;
      $Row_count = $Row_count + 1;  }
   elsif (($Found_table == 1) && ($_ =~ /^(y)/i)) {
      $Found_bias  = 0;
      $Found_table = 0;
      $Rows        = $Row_count;
      $Row_count   = 0;  }
}
close (INPUT);
#*****************************************************************
```

Perl code for extracting data from .LIS file

```
#*********************************************************************
open (OUTPUT1, "> $Mfile")                          || die "Can't open file: $!";
(print OUTPUT1 "echo off;\n")                       || die "Can't write to file: $!";   # Turn Matlab's Echo off

(print OUTPUT1 "V_p=[\n")                           || die "Can't write to file: $!";   # Start Matlab matrix
for ($i=0; $i < $Rows; $i++) {                                                          # Loop $Rows times
   (print OUTPUT1 "$V_p[$i]")                       || die "Can't write to file: $!";   # Print a row of numbers to file
   if ($i < $Rows-1) {(print OUTPUT1 "\n")          || die "Can't write to file: $!";}  # Start a new row in file
   else              {(print OUTPUT1 "];\n")        || die "Can't write to file: $!";}  # Finish last row with a "]"
}
(print OUTPUT1 "V_n=[\n")                           || die "Can't write to file: $!";   # Start Matlab matrix
for ($i=0; $i < $Bias_count; $i++) {                                                    # Loop $Bias_count times
   (print OUTPUT1 "$V_n[$i]")                       || die "Can't write to file: $!";   # Print a row of numbers to file
   if ($i < $Bias_count-1) {(print OUTPUT1 "\n")    || die "Can't write to file: $!";}  # Start a new row in file
   else                    {(print OUTPUT1 "];\n") || die "Can't write to file: $!";}  # Finish last row with a "]"
}
for ($i=0; $i < $Bias_count; $i++) {                                                    # Loop $Bias_count times
   (print OUTPUT1 "I_p(:,$i+1)=[\n")                || die "Can't write to file: $!";   # Start Matlab matrix
   for ($j=0; $j < $Rows; $j++) {                                                       # Loop $Rows times
      (print OUTPUT1 "$Isw_p[$j][$i]")              || die "Can't write to file: $!";   # Print a row of numbers to file
   if ($j < $Rows-1) {(print OUTPUT1 ";\n")         || die "Can't write to file: $!";}  # Start a new row in file
   else              {(print OUTPUT1 "];\n")        || die "Can't write to file: $!";}  # Finish last row in file
   }
}
for ($i=0; $i < $Bias_count; $i++) {                                                    # Loop $Bias_count times
   (print OUTPUT1 "I_n(:,$i+1)=[\n")                || die "Can't write to file: $!";   # Start Matlab matrix
   for ($j=0; $j < $Rows; $j++) {                                                       # Loop $Rows times
      (print OUTPUT1 "$Isw_n[$j][$i]")              || die "Can't write to file: $!";   # Print a row of numbers to file
   if ($j < $Rows-1) {(print OUTPUT1 ";\n")         || die "Can't write to file: $!";}  # Start a new row in file
   else              {(print OUTPUT1 "];\n")        || die "Can't write to file: $!";}  # Finish last row in file
   }
}
close (OUTPUT1);
#*********************************************************************
Label_END:
#*********************************************************************
```

HSPICE toolbox for Matlab

- **The Matlab HSPICE toolbox from MIT may make the PERL script on the previous pages obsolete**

  `http://www-mtl.mit.edu/research/perrottgroup/tools.htm`

Matlab code for plotting the raw data

```
%************************************************************************
% The data file contains four variables:
%   V_p   -   Inner loop of the HSPICE sweep
%   I_p   -   Rows correspond to inner loop (V_p)
%             Columns correspond to outer loop (V_n)
%   V_n   -   Outer loop of the HSPICE sweep
%   I_n   -   Rows correspond to inner loop (V_p)
%             Columns correspond to outer loop (V_n)
%************************************************************************
% Load and plot the raw data
%************************************************************************
IVmatrix

figure(1);
subplot(1,1,1);
surfc(V_n,V_p,I_p);
title('I_p vs. V_p and V_n');
xlabel('V_n');
ylabel('V_p');
zlabel('I_p');
axis tight;
shading interp
view(-60,15);
colorbar;

figure(2);
subplot(1,1,1);
surfc(V_n,V_p,I_n);
title('I_n vs. V_p and V_n');
xlabel('V_n');
ylabel('V_p');
zlabel('I_n');
axis tight;
shading interp
view(-60,15);
colorbar;
%************************************************************************
```

Matlab code for calculating and plotting I_{common}

```
%************************************************************************
% Plot common mode currents (=diagonal of raw data)
%************************************************************************

figure(3);
subplot(1,1,1);
plot(V_p,diag(I_p));
title('Icommon_p vs. V_p');
xlabel('V_p');
ylabel('Icom_p');
axis tight;
grid

figure(4);
subplot(1,1,1);
plot(V_n,diag(I_n));
title('Icommon_n vs. V_n');
xlabel('V_n');
ylabel('Icom_n');
axis tight;
grid

%************************************************************************
```

Matlab code for calculating and plotting I_{diff}

```
%************************************************************************
% Generate "normalizing" matrices with rows containing diagonal values
%************************************************************************
norm_p = zeros(size(I_p));
norm_n = zeros(size(I_n));

diag_p = diag(I_p);
diag_n = diag(I_n);

for i = 1:length(diag_p)
   norm_p(i,:) = diag_p(i);       % Rows correspond to V_p
end                               % Columns correspond to V_n
for i = 1:length(diag_n)
   norm_n(:,i) = diag_n(i);       % Rows correspond to V_p
end                               % Columns correspond to V_n
%************************************************************************
% Calculate and plot differential current
%************************************************************************
dI_p = norm_p - I_p;
dI_n = I_n - norm_n;

figure(5);
subplot(1,1,1);
surfc(V_n,V_p,dI_p);
title('dI_p vs. V_p and V_n');
xlabel('V_n');
ylabel('V_p');
zlabel('dI_p');
axis tight;
shading interp
view(-60,15);
colorbar;
```

```
figure(6);
subplot(1,1,1);
surfc(V_n,V_p,dI_n);
title('dI_n vs. V_p and V_n');
xlabel('V_n');
ylabel('V_p');
zlabel('dI_n');
axis tight;
shading interp
view(-60,15);
colorbar;
```

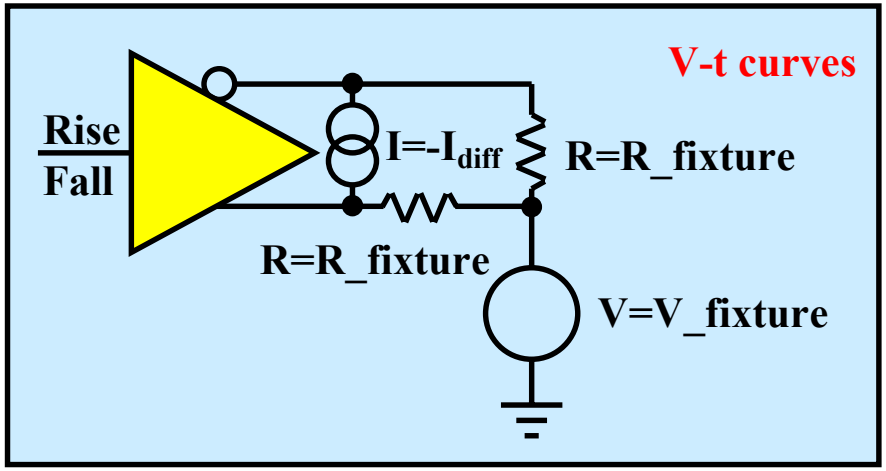
Matlab code for surface fitting and plotting I_{diff}

```
%************************************************************************
% Generate xp, xn and y matrices and calculate coefficients to
% dI_p = a0 + a1*xp + a2*xn + a3*xp*xn + a4*(xp)^2 + a5*(xn)^2
%************************************************************************
[rows,cols] = size(dI_p);
for c = 1:cols
   for r = 1:rows
      xp((c-1)*rows+r) = V_n(r);
      xn((c-1)*rows+r) = V_n(c);
      y((c-1)*rows+r) = dI_p(r,c);
   end
end

coeff0 = [0, 1e-3, 1e-3, 1e-3, 1e-6, 1e-6];             % Initial guess
Inputs=[xp; xn];

[coeff,resnorm,residual] = lsqcurvefit(@S_MOSFET_f,coeff0',Inputs,y);
coeff
resnorm
max(residual)

for c = 1:length(V_n)
  for r = 1:length(V_p)
    dI(r,c)=coeff(1)+coeff(2)*V_p(r)+coeff(3)*V_n(c)+coeff(4)*(V_p(r).*V_n(c))+coeff(5)*V_p(r).^2+coeff(6)*V_n(c).^2;
  end
end

figure(7);
subplot(1,1,1);
surfc(V_n,V_p,dI);
title('dI vs. V_p and V_n');
xlabel('V_n');
ylabel('V_p');
zlabel('dI');
axis tight;
shading interp
view(-60,15);
colorbar;
%************************************************************************
```

Matlab code for saving data for IBIS

```
%************************************************************************
% Write [Series MOSFET] keyword data to files (Differential)
%************************************************************************

for count = 1:19
   fid=fopen(strcat('Data_',num2str(count),'.txt'),'w');
   fprintf(fid,'%10.4f      %10.8e\r\n',sortrows([V_n(end)-V_n((count-1)*10+1:end),diag(dI_n,(count-1)*10)],1)');
   fclose(fid);
end

%************************************************************************
% Write regular IV curve data to files (Common)
%************************************************************************

fid=fopen(strcat('Data_Ip.txt'),'w');
fprintf(fid,'%10.4f      %10.8e\r\n',sortrows([V_p(end)-V_p,diag(I_p)],1)');
fclose(fid);

fid=fopen(strcat('Data_In.txt'),'w');
fprintf(fid,'%10.4f      %10.8e\r\n',sortrows([V_n,diag(I_n)],1)');
fclose(fid);

%************************************************************************
```

Generating waveform tables (Vt curves)

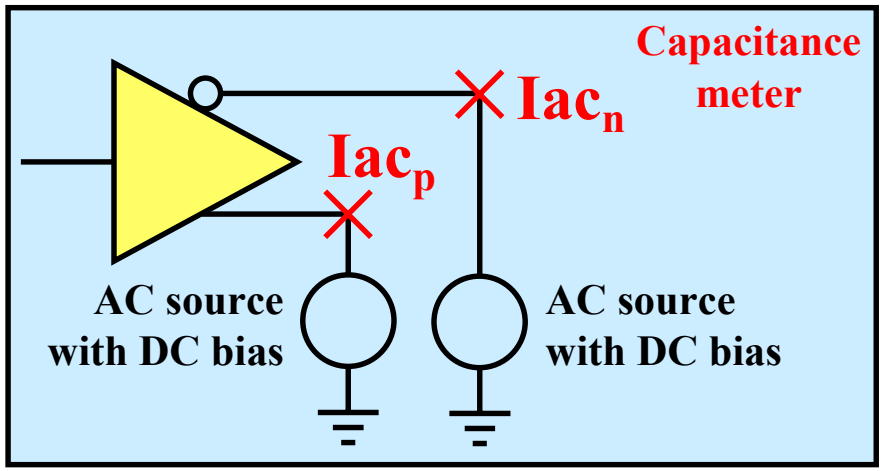


- **Run time domain simulations (.TRAN) using the above circuit**
  - R_fixture should be close to Z_{common} of the T-line the buffer will drive
  - V_fixture should be close to the DC high and low levels of the signal swing the buffer will achieve
  - Use current source to cancel differential current
- **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**

HSPICE implementation of a waveform generator

```
LVDS differential buffer Vt curve generator
*******************************************************************************
.TRAN  10.0ps  5.0ns
.OPTIONS POST=1 PROBE INGOLD CO=132 NUMDGT=8 ACCURATE RMAX=0.2
*******************************************************************************
.PRINT TRAN
+ V_p    = V(Out_p)
+ V_n    = V(Out_n)
*******************************************************************************
V_vcc    VCC 0   DC= 1.8
V_vtt    Vtt 0   DC= 1.0
*******************************************************************************
V_in   In   0    PULSE  (1.8  0.0  0.0  10.00ps 10.00ps 5ns 10ns)
V_en   En   0    DC= 1.8         $ Enable is active high.  This is drive mode.
*******************************************************************************
X1  In  Out_p  Out_n  Vcc  0  En  BUFFER
R1p     Out_p  Vtt    R= 50
R1n     Out_n  Vtt    R= 50
*-------------------------------------------------------------------------------
.param
+ k0 =  6.503179353194756e-006   $ Coefficients from Matlab surface fit
+ k1 = -2.541816815085296e-003   $ representing the Idiff surface
+ k2 =  2.541334083148360e-003
+ k3 = -2.809854297776799e-005
+ k4 =  4.580644144607367e-004
+ k5 = -4.354430013260378e-004
Gdn  Out_p  Out_n  Cur='k0 + k1*V(Out_p) + k2*V(Out_n) + k3*(V(Out_p)*V(Out_n)) + \\
+                      k4*Pow(V(Out_p),2) + k5*Pow(V(Out_n),2)'
*******************************************************************************
.alter
V_vtt    Vtt 0   DC= 1.2542
.alter
V_vtt    Vtt 0   DC= 1.5
*******************************************************************************
.END
```

Measuring the common and differential C_comp



- **Run frequency domain simulations (.AC) 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: $C = Im(I) / (2\pi f * Amplitude)$**
  - 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**

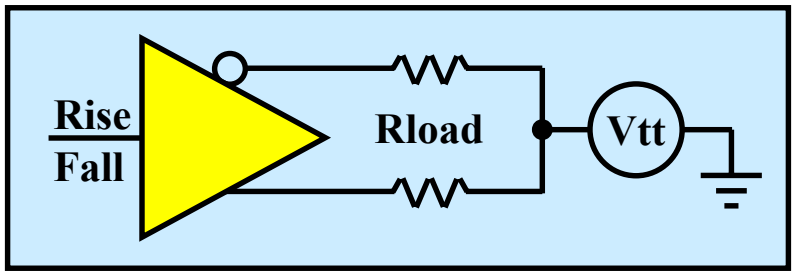
HSPICE implementation of differential C_meter

```
Differential Capacitance vs. frequency and DC offset
**********************************************************************
.AC  DEC  10  1.0  1.0e+9  SWEEP  Bias  LIN  7  1.0  1.6
.OPTIONS POST=1 PROBE INGOLD CO=132 NUMDGT=8 ACCURATE RMAX=0.2
**********************************************************************
.param Vcc  = 1.8
.param Vin  = 0.0
.param Ven  = 1.8          $ Enable is active high.  This is drive mode.
.param Bias = 1.2542       $ 'Vcc/2'
*---------------------------------------------------------------------
.param Ampl   = 1.0e-3
.param Ampl_p = Ampl
.param Ampl_n = 0
.param pi     = 3.14159265358979323846264338327 95
**********************************************************************
.probe AC
+ Cp    = PAR('sgn(Ampl_p) * (abs(Ii(Vac_p))-abs(Ii(Vac_n)))/(2*pi*HERTZ*Ampl)')
+ Cn    = PAR('sgn(Ampl_n) * (abs(Ii(Vac_n))-abs(Ii(Vac_p)))/(2*pi*HERTZ*Ampl)')
+ Cdiff = PAR('(sgn(Ampl_p)*abs(Ii(Vac_n)) + sgn(Ampl_n)*abs(Ii(Vac_p)))/(2*pi*HERTZ*Ampl)')
**********************************************************************
Vac_p  Out_p  0      AC= Ampl_p  DC= Bias
Vac_n  Out_n  0      AC= Ampl_n  DC= Bias
Vdc      Vdc  0      DC= Vcc
Vvcc     Vdc  Vcc    DC= 0
Vvss     Vss  0      DC= 0
**********************************************************************
V_in     In   0      DC= Vin
V_en     En   0      DC= Ven
**********************************************************************
X1  In  Out_p  Out_n  Vcc  Vss  En  BUFFER
**********************************************************************
.alter
.param Ampl_p= 0
.param Ampl_n= Ampl
**********************************************************************
.END
```

Comments on C_comp in IBIS

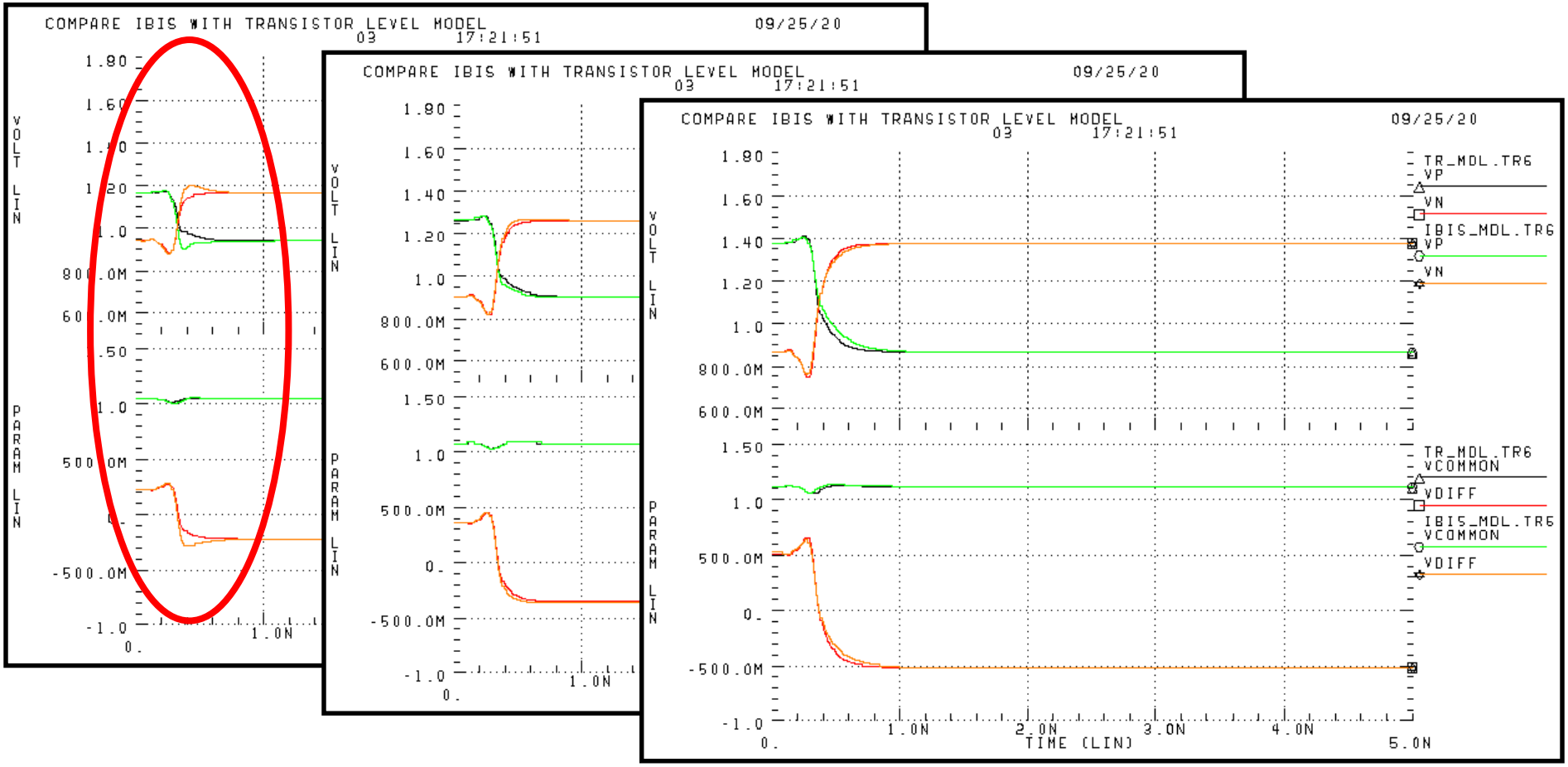
- The capacitance measurement setup on the previous slide provides frequency and voltage dependent capacitance
- Since C_comp in IBIS (through 4.0) can use a “single” value only we need to make some guesses for picking the best value from the available capacitance measurement results
- Put C_{common} into the C_comp parameter of [Model] that represents the common mode components of the buffer
- Use C_{diff} with the [C Series] keyword in the [Model] that represents the differential mode components of the buffer
- Using future versions of IBIS it will be possible to write models which make use of all of this data

Comparing the simulation waveforms of the transistor and IBIS models

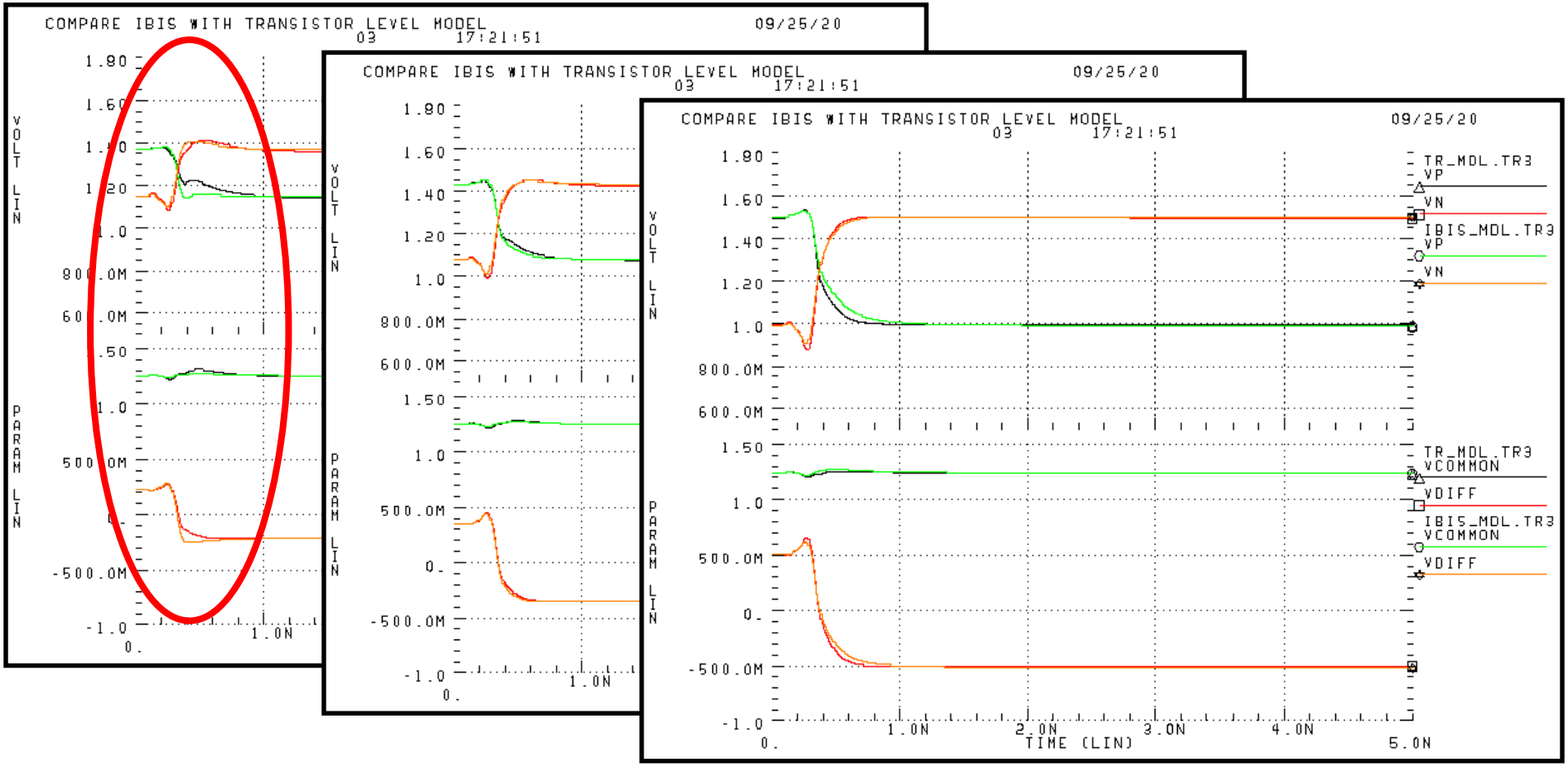


- **The waveforms on the following pages show the differential buffer driving a resistive load**
  - The resistor was split into two elements so that the node in the middle could be tied to a bias voltage (Vtt)
  - Three different resistance values were tested
    - **2 x 25 Ω, 2 x 50 Ω, 2 x 100 Ω**
  - Three different termination conditions (Vtt) were tested
    - **1.0 V source, No source (natural $V_{common} \cong 1.25$ V), 1.5 V source**
- **The plots on the top half show the actual waveforms**
- **The plots on the bottom half show V_{common} and V_{diff}**
  - $V_{common} = (V_p + V_n) / 2$
  - $V_{diff} = V_p - V_n$

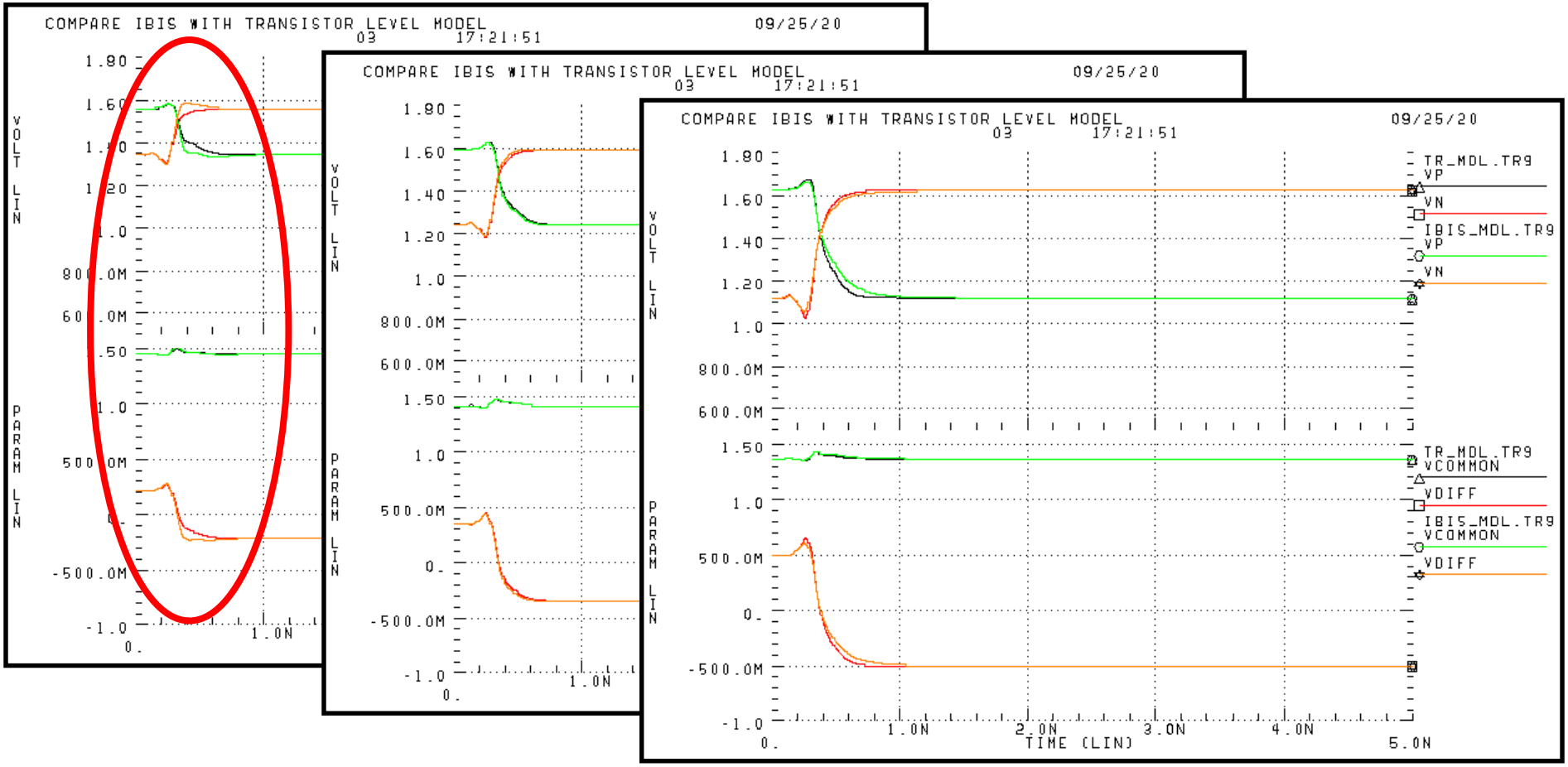
Waveforms with incorrect capacitance values (Vtt = 1.0 V)



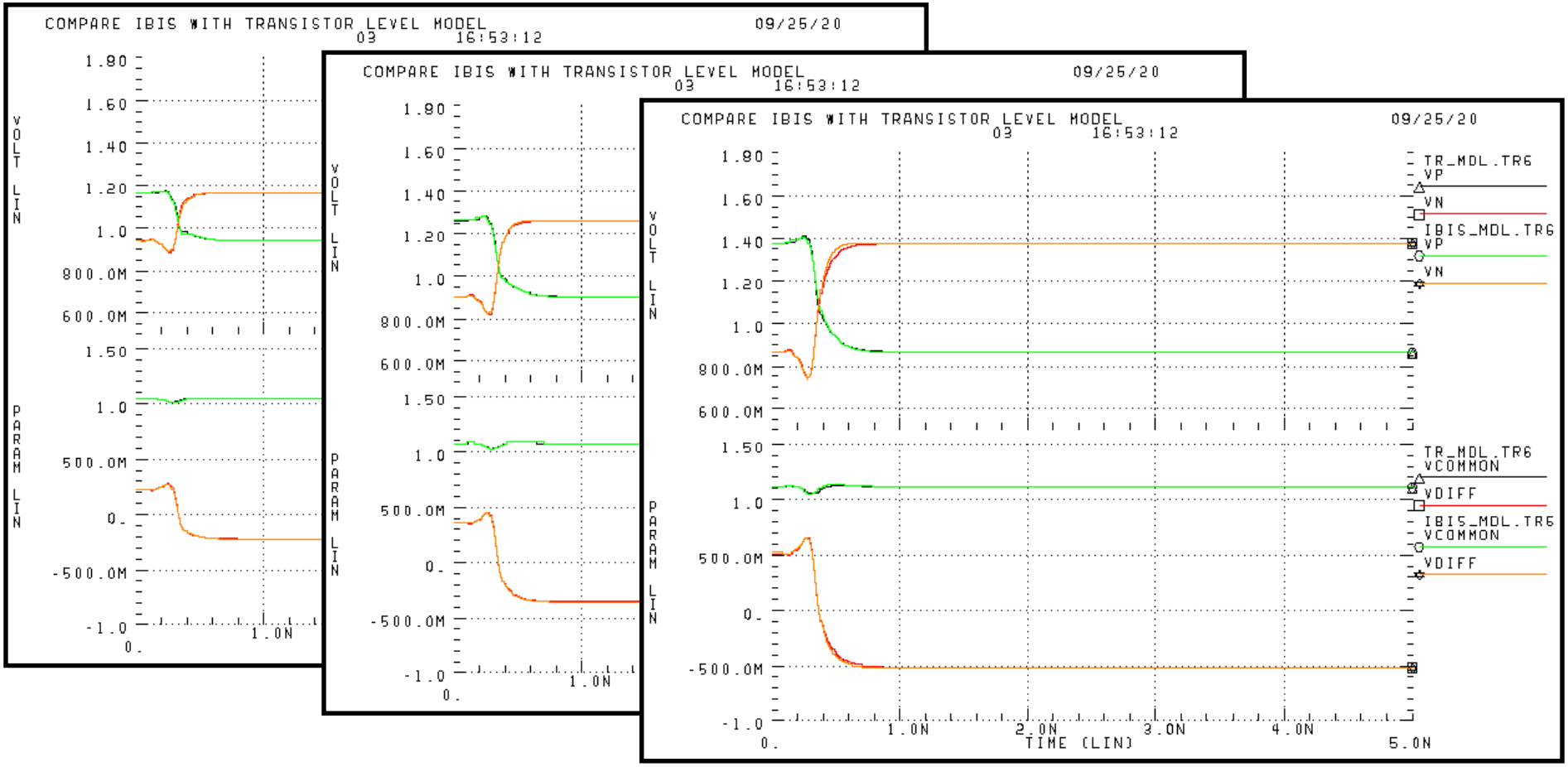
Waveforms with incorrect capacitance values (Vtt ≅ 1.25 V)



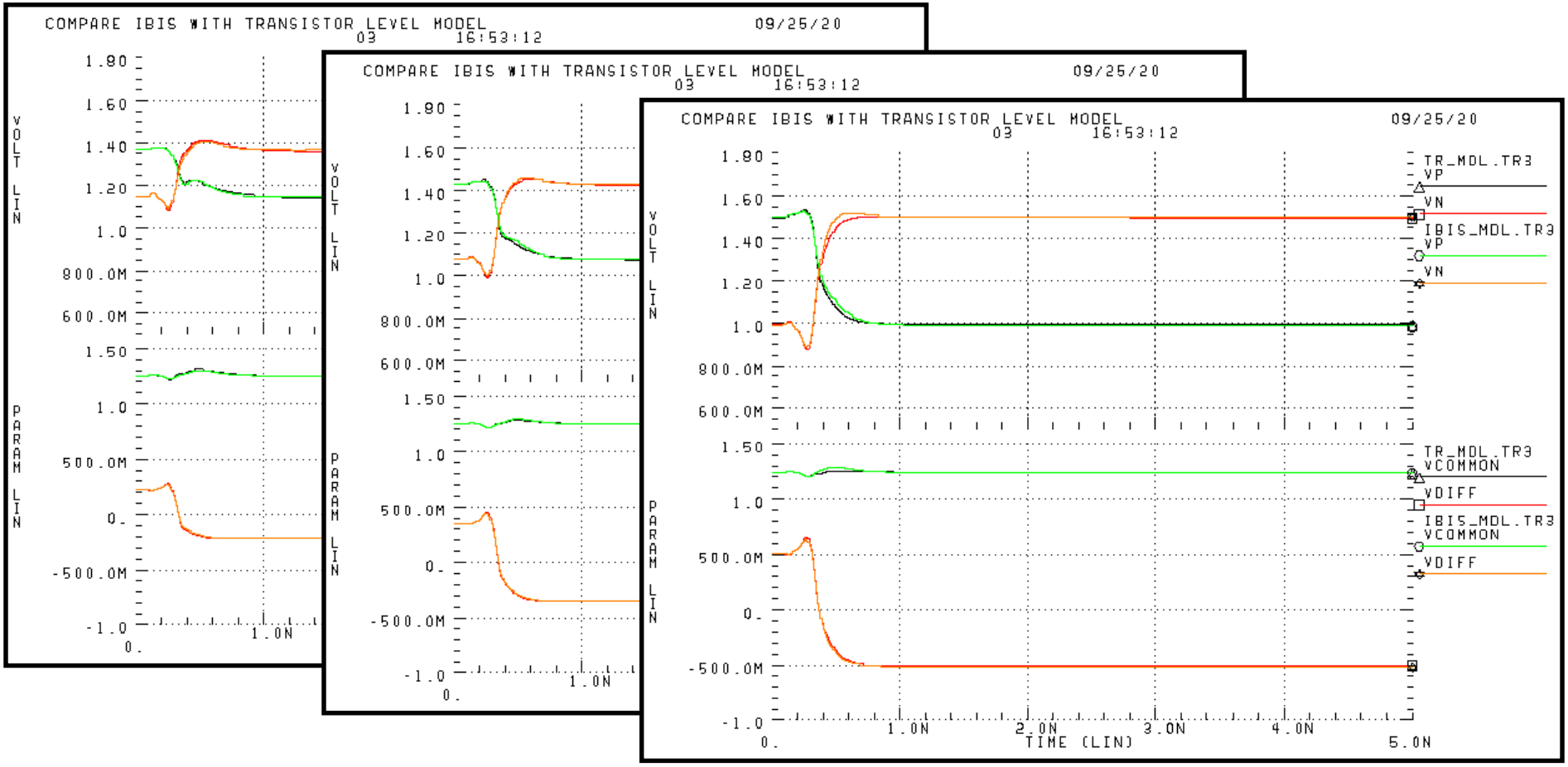
Waveforms with incorrect capacitance values (Vtt = 1.5 V)



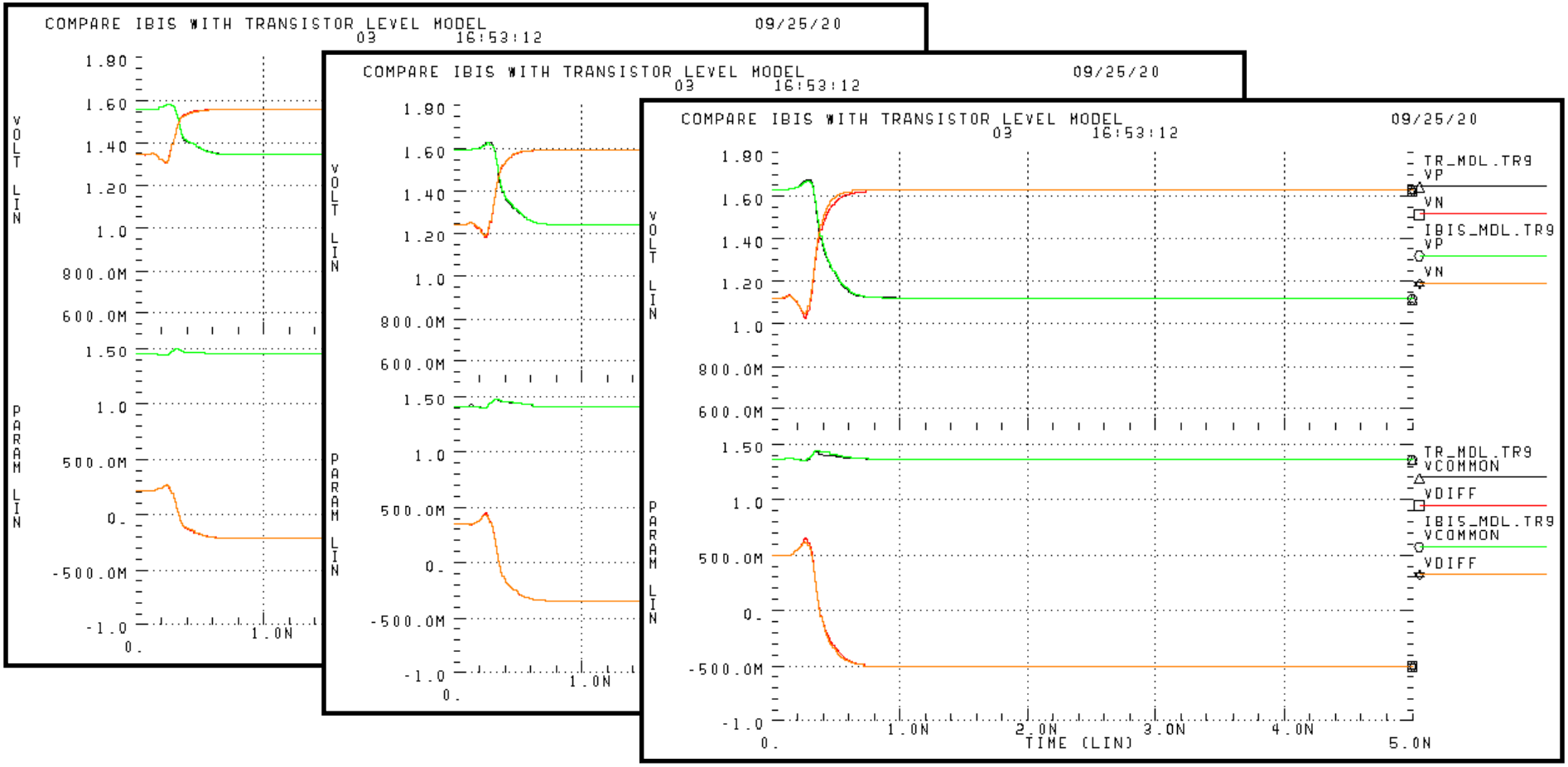
Waveforms with Vtt = 1.0 V



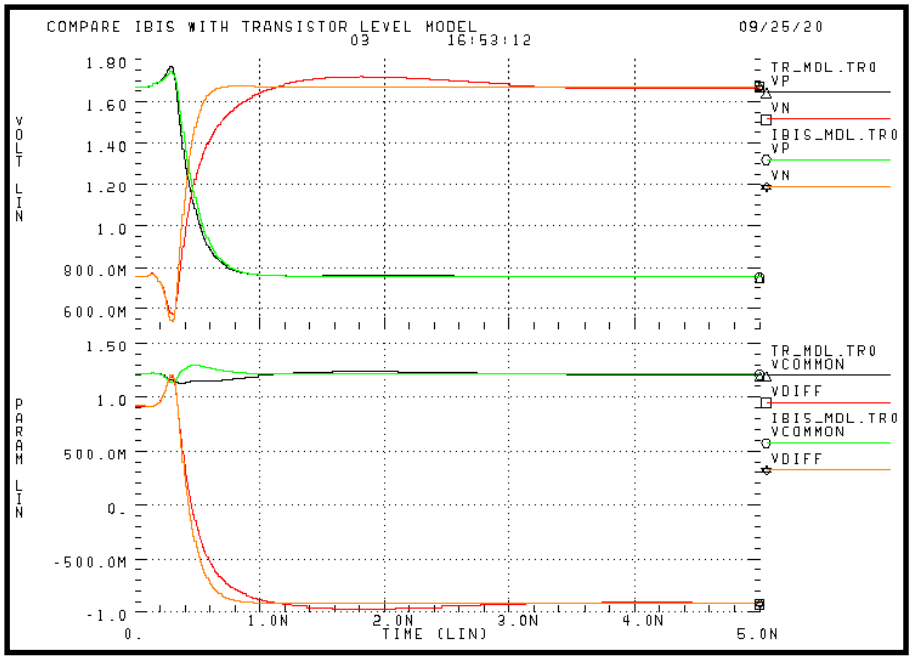
Waveforms without Vtt source ($V_{common} \cong 1.25$ V)



Waveforms with Vtt = 1.5 V

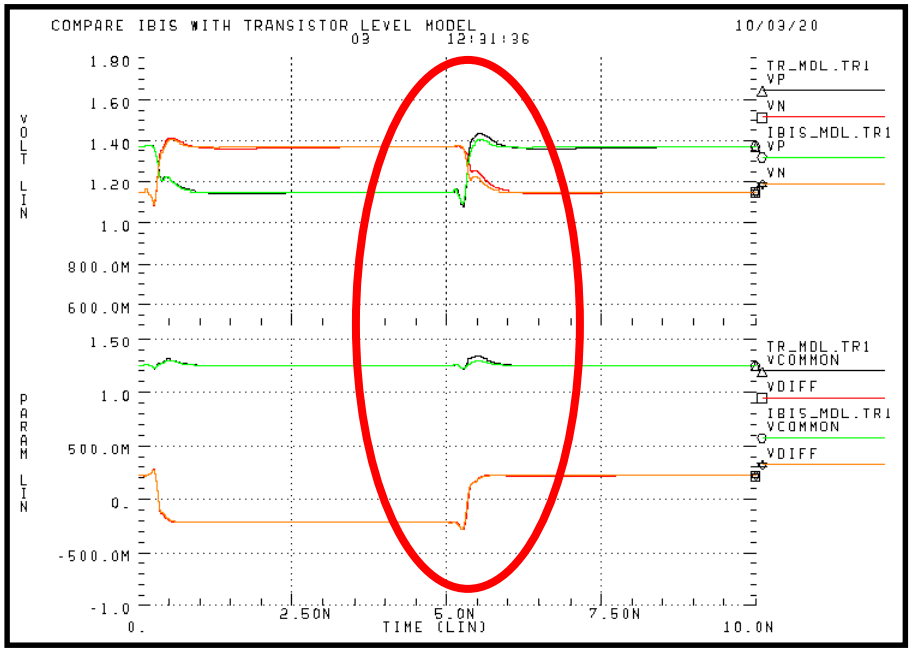


Further study (2003):
- Waveform comparison without load



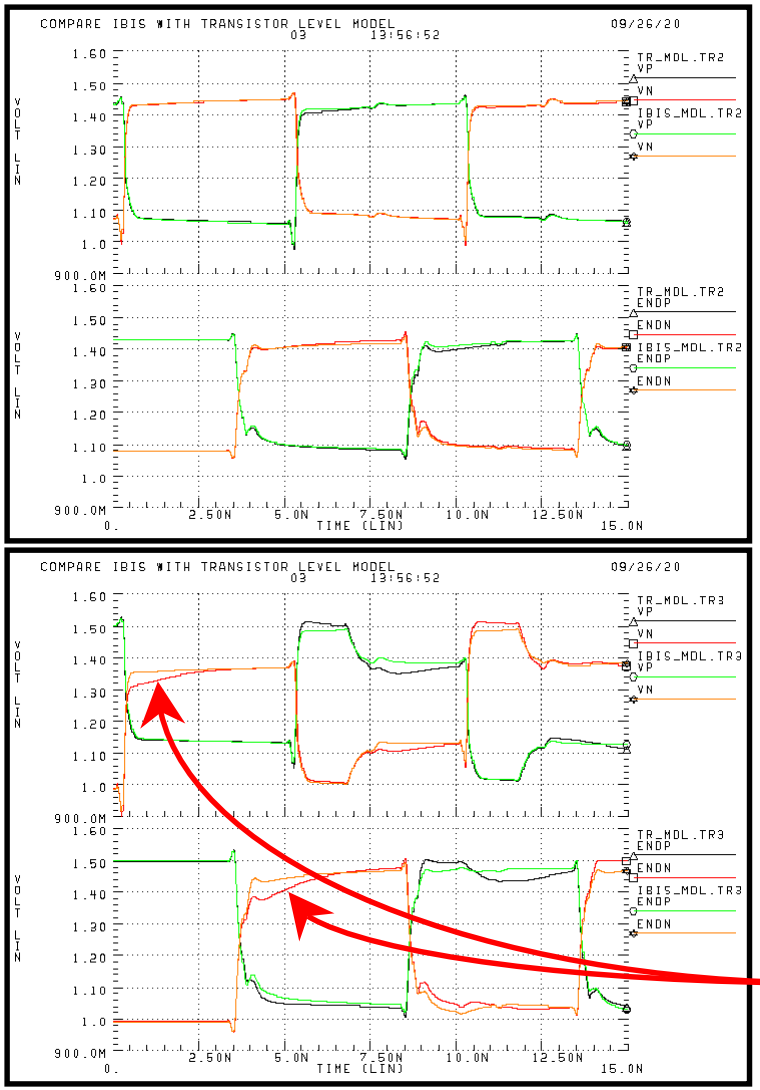
- **The no-load waveforms show some internal effects which need to be analyzed and understood better**
- **It may not be possible to model these effects with IBIS 3.2**

Further study (2003):
- Asymmetry in transistor model



- **Every other transition correlates less accurately**
- **Different set of waveforms may be needed for rising and falling edge stimulus in IBIS model**
- **IBIS 3.2 cannot handle data dependent models**

Further study (2003):
- Waveform comparison with T-line



- **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**

Conclusion

- **A more accurate modeling technique was shown using conventional IBIS 3.2 keywords**
  - This technique describes the static (DC) currents of a differential buffer completely and accurately
  - As a result the DC levels of the signals are correct under all loading conditions
- **A technique was shown how to measure the common and differential mode capacitance of the buffer**
  - As a result the waveforms show good correlation under “well behaved” conditions
- **Some discrepancies were also shown, which may be related to the voltage and frequency dependent die capacitance and/or higher order effects in the transient behavior of the buffer**
  - These are much more difficult to characterize and may need the new language extensions (*-AMS) of the upcoming IBIS 4.1 specification