Peak Distortion Analysis implemented in VHDL-AMS

IBIS Advanced Technology Modeling task group

September 5, 2006

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

Background

- **It all started with the discussions in the IBIS Macro Modeling subcommittee (now called Advanced Technology Modeling) on the Cadence API proposal which claims that the *-AMS languages are not sufficient for channel analysis**

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

- **The committee has a hard time to disprove this claim because so far no one has done it using the *-AMS language(s) and tools**

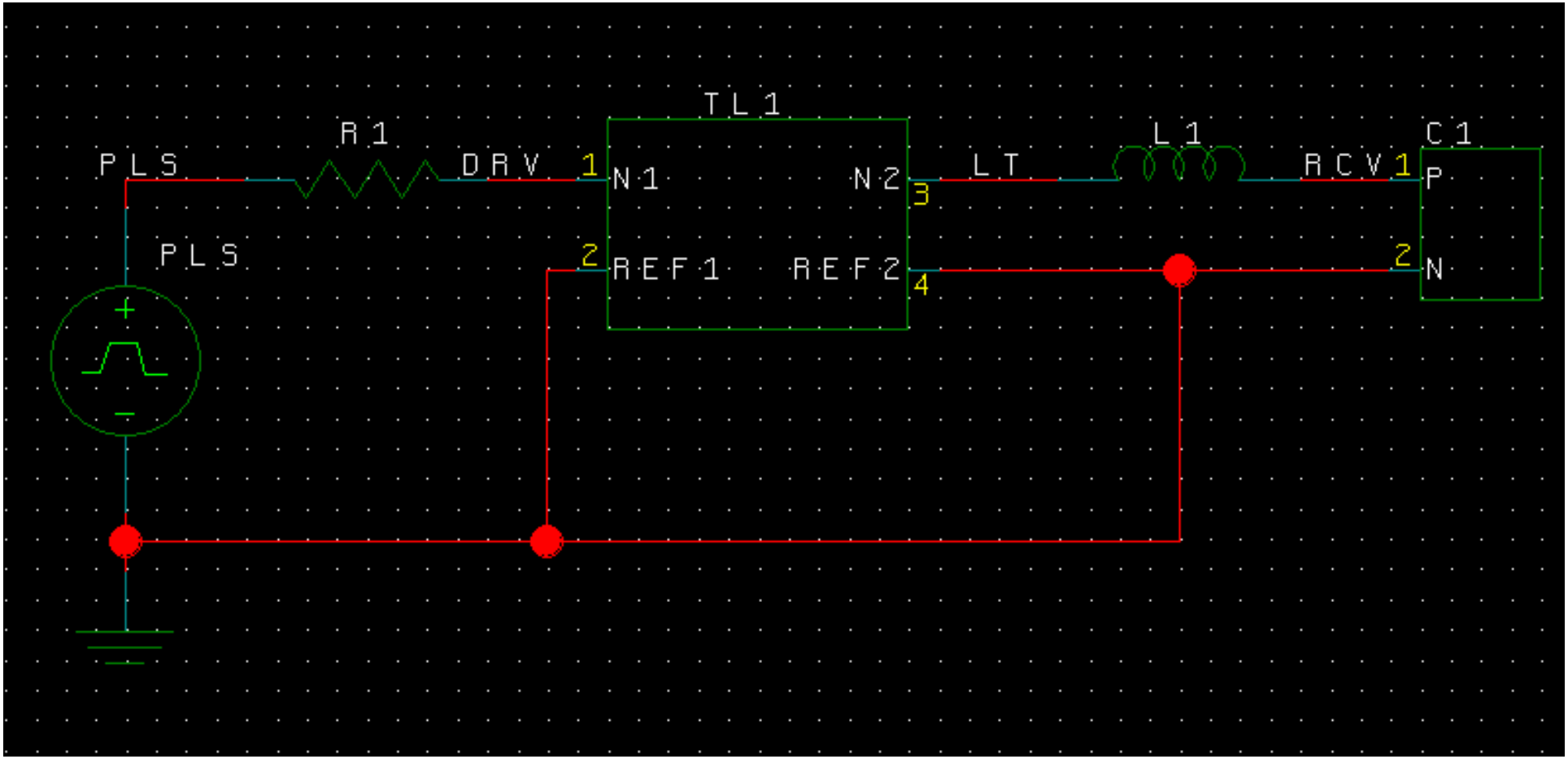
- **Well, let's give it a try and code up the algorithm found in Brian Casper's publicly available presentation in VHDL-AMS…**

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

The simulation setup

- **For the sake of this initial experiment, a simple circuit was used to generate a reasonable pulse response**
- **The circuit of the "channel" consists of a Thevenin driver using a pulse voltage source and a resistor, an ideal T-line, an inductor and capacitor at the end**
  - details shown on the next page
- **For the sake of this experiment I didn't care whether the channel was realistic or not, as long as it gave a reasonable pulse response**
- **The capacitor model has two "architectures"**
  - a normal capacitor
  - a normal capacitor plus the PDA algorithm

Schematics of the simulated circuit

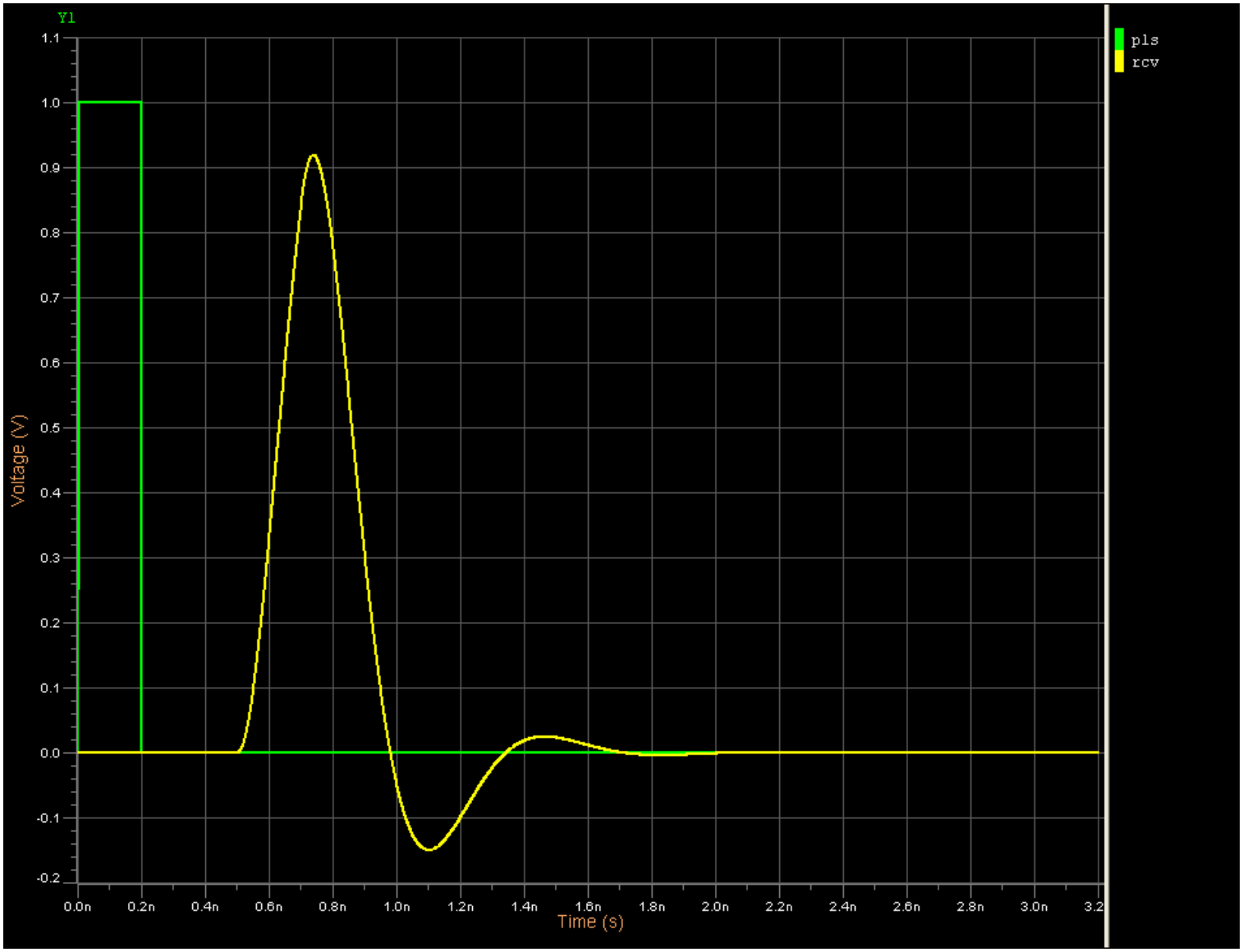


Pulse: 1 V, 200 ps wide, 1 ps edge
R1: 100 Ω
T-line: 100 Ω, 0.5 ns, lossless
L1: 10 nH
C1: 1 pF

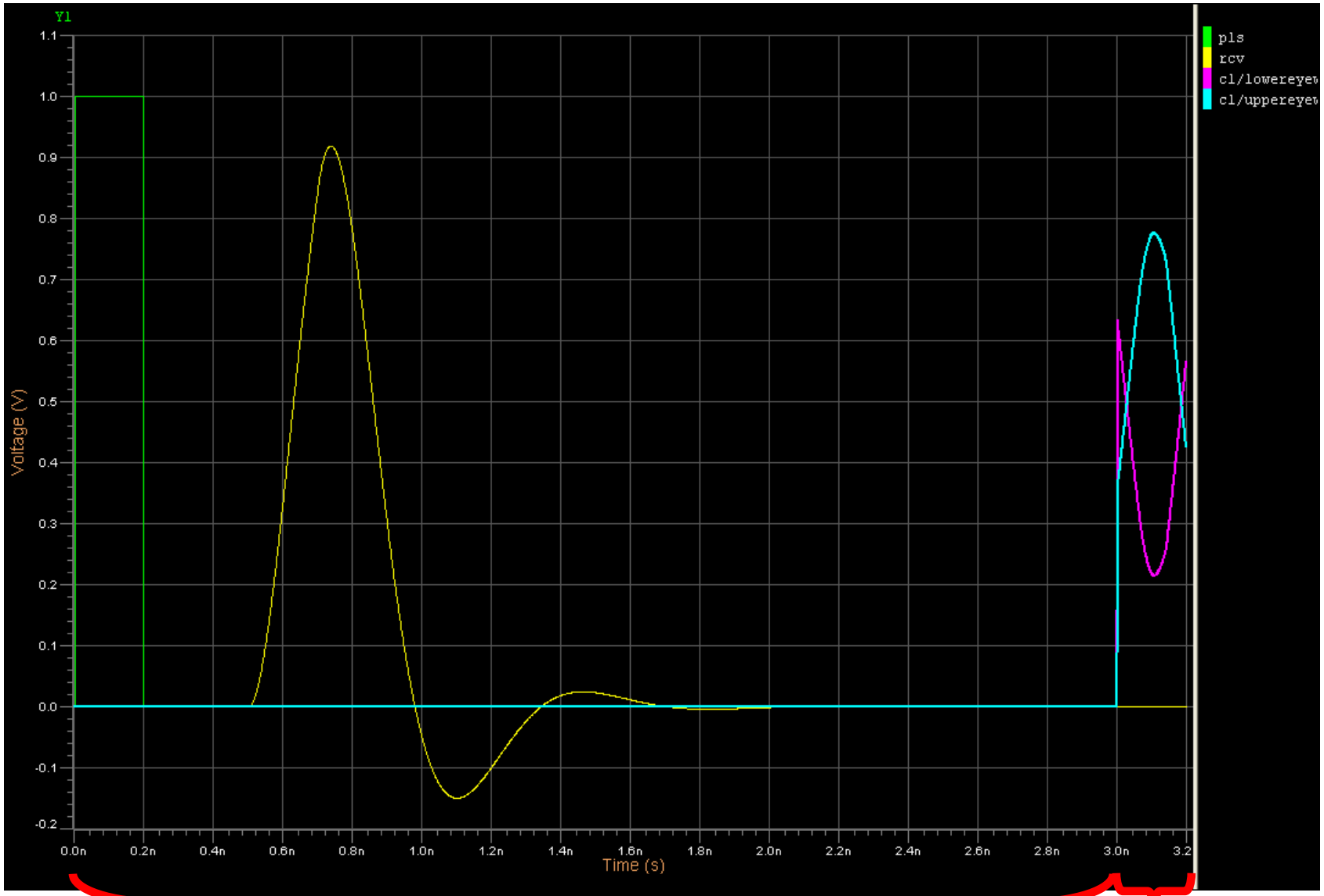
Overview of the operation of the model

- **A normal time domain simulation is started**
  - this example uses the first 3 ns of the simulation to generate the pulse response
  - during this portion of the simulation the points of the pulse response waveform are stored in a “real_vector” (3000 points in this example)
  - a 1 ps fixed time step is used to make the waveform processing simpler
- **At the 3 ns time point of the simulation a process is activated**
  - this is executed in the digital solver of the tool (which makes it fast)
  - the process considers the peak of the pulse response the “cursor” point and finds the index of that point in the vector
  - then the process executes a function twice to generate an upper and lower eye contour using the methodology described in Brian Casper’s presentation
- **The time domain simulation is continued for another 200 ps**
  - this is done to plot the content of the upper and lower eye contour vectors

Pulse response of the circuit



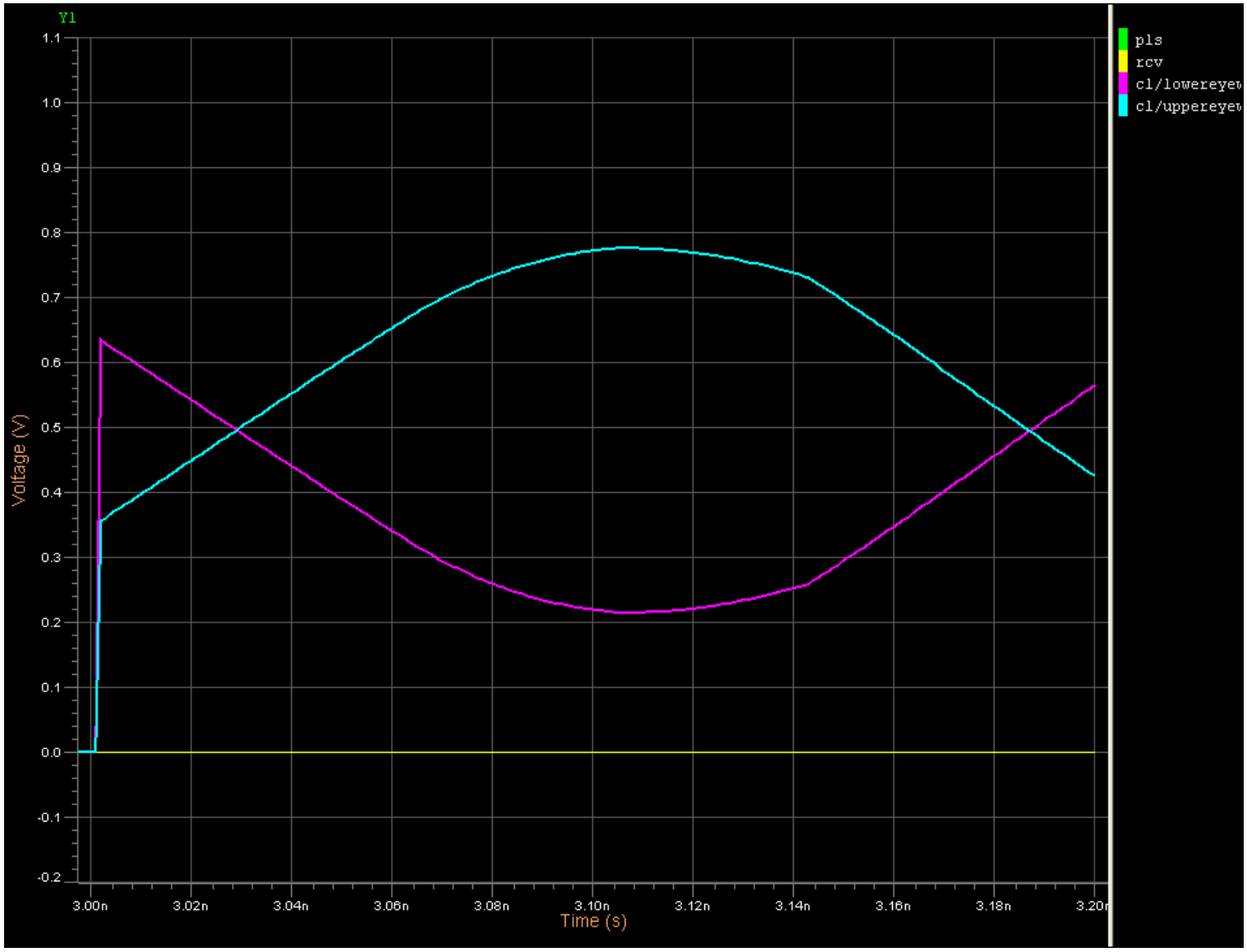
Adding the results of the PDA post processing step



Pulse response TD simulation

PDA result "worst eye"

Zooming in on the worst eye portion of the plot



Summary, benchmarks and future work

- **This experiment implemented the basic equations of PDA only**
  - pg. 45 – 64 in Brian Casper's presentation (cited on pg. 2)
  - no cross talk,
  - no jitter,
  - no statistical ISI or BER analysis included
- **Benchmarks with a 3000 point pulse response**
  - approximately 230 ms CPU time without PDA algorithm
  - approximately 240 ms CPU time with PDA algorithm
- **This experiment illustrates that VHDL-AMS can be used for "signal processing" algorithms**
  - still need to show that the more computationally intensive statistical analysis can also be implemented
  - implement same in Verilog-A(MS)

BACKUP

VHDL-AMS code

Code - processes (digital equations)

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

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

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

Code - analog equations

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

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

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

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

Code - function FindCursorIndex

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

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

Code - (essentials of) function EyeContour

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

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

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