A VHDL-AMS Pre/De-emphasis buffer model using IBIS v3.2 data

IBIS Summit at DesignCon East 2004

Holiday Inn, Boxborough Woods, MA

April 5, 2004

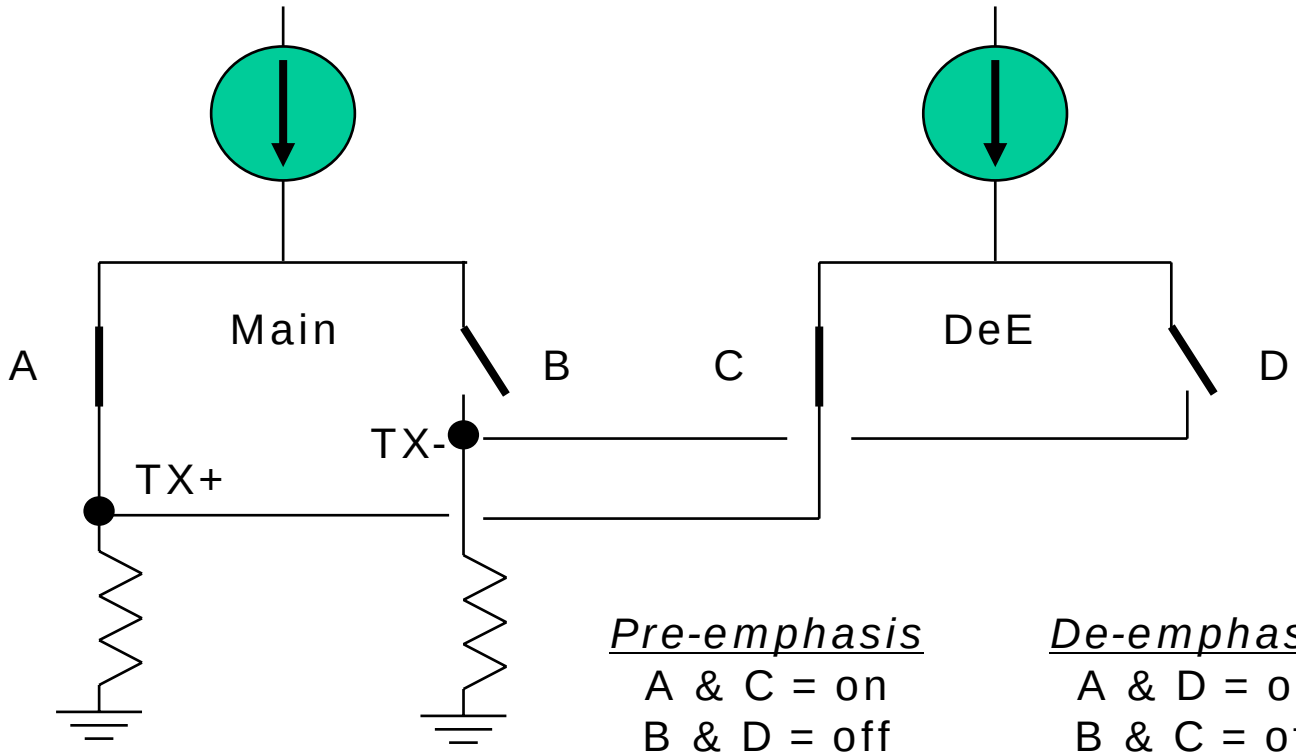
Arpad Muranyi
Signal Integrity Engineering
Intel Corporation
arpad.muranyi@intel.com

Outline

- Block diagram of a Pre/De-emphasis buffer
- Background / features
- Block diagram of VHDL-AMS model
- Work to be done
- Code segments
- Simulation schematics
- Waveform examples
- Summary

Block diagram of a 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

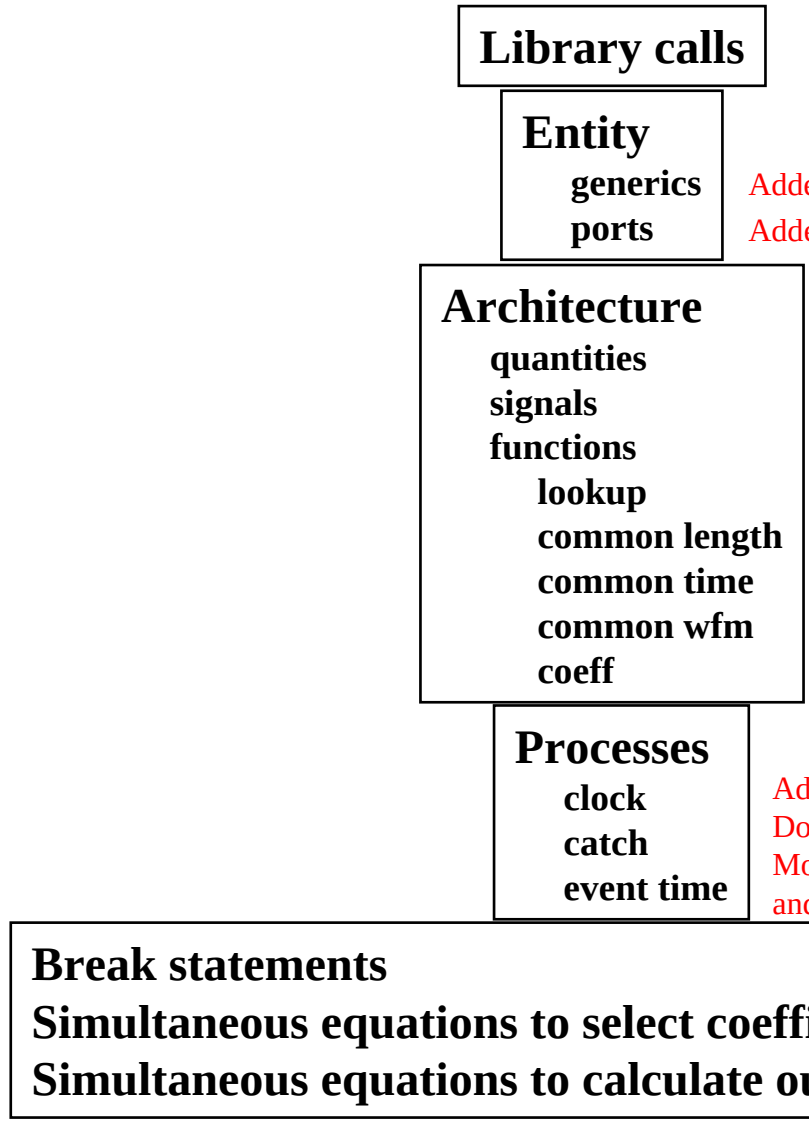
De-emphasis
A & D = on
B & C = off

Curtesey of Michael Mirmak

Background / features

- **The VHDL-AMS Pre/De-emphasis buffer model introduced in this presentation is based on the VHDL-AMS single-ended and differential buffer models introduced at the June 5 and June 23, 2003 and February 2, 2004 IBIS Summits**
  - http://www.eda.org/pub/ibis/summits/jun03b/muranyi1.pdf
  - http://www.eda.org/pub/ibis/summits/feb04a/muranyi2.pdf
- **This model also incorporates the modeling technique developed for differential buffers presented at the October 15, 2002 and October 21, 2003 IBIS Summits**
  - http://www.eda.org/pub/ibis/summits/oct02/muranyi.pdf
  - http://www.eda.org/pub/ibis/summits/oct03/muranyi.pdf
- **Main features:**
  - *1 digital input,* 1 digital enable, *1 digital clock input* (with edge selector parameter)
  - 1 digital output (dummy for receiver out)
  - 4 analog supplies, *2 analog I/O ports* (differential pair with selectable initial condition)
  - Uses normal IBIS data (I-V and V-t tables) for “common mode” component [Model]
  - Uses fitted coefficients (calculated from the I-V tables of the [Series MOSFET] model) for the “differential” component
  - Includes 4-way split C_comp plus C_diff
  - “I/O_open_source” model, i.e. there is no “pulldown” I-V table (for SATA buffers)
  - Uses the 1 equation / 1 unknown algorithm

Block diagram of the VHDL-AMS model



Added "Edge" and "Out_ini"
Added digital port for clock input

Doubled most quantities and signals for 2nd tap of driver,
Removed quantities and signals for pulldown

Reduced number of vectors by half since
this model handles 2 V-t tables only

Replaced 2EQ/2UK with 1EQ/1UK algorithm

Added clocking logic,
Doubled digital logic for 2nd tap of driver,
Modified previous logic to be more efficient,
and removed pulldown equations

Doubled all equations for 2nd tap of driver, modified equations to accommodate the more efficient logic, and removed pulldown equations

Work to be done

- **Modify C_comp compensation algorithm to account for mutual loading effects between taps**
  - See Michael Mirmak's presentation today
  - Have equations already, but need to implement and test them
- **Test and verify that the model is working correctly**
  - Generate waveform overlays with SPICE model simulations

VHDL-AMS implementation – changes (1)

```
entity IBIS_DIFF_OS_CLK_2TAP is

  generic (Edge        : integer := 2;      -- "0" = Falling edge triggered
                                            -- "1" = Rising edge triggered
                                            -- "2" = Triggers on both edges
           Out_ini     : std_logic := 'Z';  -- Initial condition for output

           C_comp      : real := 1.00e-12;  -- Default C_comp value and
           k_C_comp_pc : real := 0.25;      -- splitting coefficients
           k_C_comp_pu : real := 0.25;
           k_C_comp_pd : real := 0.25;
           k_C_comp_gc : real := 0.25;
           C_diff      : real := 50.0e-15;  -- Default C_diff value (50.0fF)
           ------------------------------------------------------------------
           -- [Pullup Reference] and [Pulldown Reference] values
           ------------------------------------------------------------------
           V_pu_ref : real := 1.8;
           V_pd_ref : real := 0.0;
           ------------------------------------------------------------------
           -- Coefficients of Idiff surface (from Matlab surface fitting)
           ------------------------------------------------------------------
           k0 : real := -6.503179353194756e-006;
           k1 : real :=  2.541816815085296e-003;
           k2 : real := -2.541334083148360e-003;
           k3 : real :=  2.809854297776799e-005;
           k4 : real := -4.580644144607367e-004;
           k5 : real :=  4.354430013260378e-004;

           R_diff : real := 700.0;  -- In case a linear resistor does the job
-------------------------------------------------------------------------------
-- Vectors of the IV curve tables
-------------------------------------------------------------------------------
```

VHDL-AMS implementation – changes (2)

```
-------------------------------------------------------------------------
-- V_fixture and R_fixture values
-------------------------------------------------------------------------
          Vfx_pu_on  : real := 1.0;
          Vfx_pu_off : real := 1.0;

          Rfx_pu_on  : real := 50.0;
          Rfx_pu_off : real := 50.0;

          ---------------------------------------------------------------
          Delta_t    : real := 1.0e-12);      -- This parameter
          -- determines what the maximum time delta will be between the
          -- points of the Vt curves and scaling coefficient curves
          -- after preprocessing the input data.

     -------------------------------------------------------------------------
     port (signal   In_D   : in   std_logic;
           signal   En_D   : in   std_logic;
           signal   Rcv_D  : out  std_logic;
           signal   Clk    : in   std_logic;

           terminal IO_p   :      electrical;
           terminal IO_n   :      electrical;
           terminal PC_ref :      electrical;
           terminal PU_ref :      electrical;
           terminal PD_ref :      electrical;
           terminal GC_ref :      electrical);

end entity IBIS_DIFF_OS_CLK_2TAP;
--=========================================================================
```

VHDL-AMS implementation – changes (3)

```
--=============================================================================
architecture DIFF_OS_CLK_2TAP_1EQ of IBIS_DIFF_OS_CLK_2TAP is
  ------------------------------------------------------------------------------
  -- Common mode components for IV curves
  ------------------------------------------------------------------------------
  quantity  Vpc_p_0    across  Ipc_p_0    through  PC_ref  to  IO_p;
  quantity  Vpu_p_0    across  Ipu_p_0    through  PU_ref  to  IO_p;
  quantity  Vgc_p_0    across  Igc_p_0    through  IO_p    to  GC_ref;

  quantity  Vpc_n_0    across  Ipc_n_0    through  PC_ref  to  IO_n;
  quantity  Vpu_n_0    across  Ipu_n_0    through  PU_ref  to  IO_n;
  quantity  Vgc_n_0    across  Igc_n_0    through  IO_n    to  GC_ref;

  quantity  Vpc_p_1    across  Ipc_p_1    through  PC_ref  to  IO_p;
  quantity  Vpu_p_1    across  Ipu_p_1    through  PU_ref  to  IO_p;
  quantity  Vgc_p_1    across  Igc_p_1    through  IO_p    to  GC_ref;

  quantity  Vpc_n_1    across  Ipc_n_1    through  PC_ref  to  IO_n;
  quantity  Vpu_n_1    across  Ipu_n_1    through  PU_ref  to  IO_n;
  quantity  Vgc_n_1    across  Igc_n_1    through  IO_n    to  GC_ref;
  ------------------------------------------------------------------------------
  -- Common mode components for C_comp
  ------------------------------------------------------------------------------
  quantity  Vc_pc_p  across  Ic_pc_p  through  PC_ref  to  IO_p;
  quantity  Vc_pu_p  across  Ic_pu_p  through  PU_ref  to  IO_p;
  quantity  Vc_pd_p  across  Ic_pd_p  through  IO_p    to  PD_ref;
  quantity  Vc_gc_p  across  Ic_gc_p  through  IO_p    to  GC_ref;

  quantity  Vc_pc_n  across  Ic_pc_n  through  PC_ref  to  IO_n;
  quantity  Vc_pu_n  across  Ic_pu_n  through  PU_ref  to  IO_n;
  quantity  Vc_pd_n  across  Ic_pd_n  through  IO_n    to  PD_ref;
  quantity  Vc_gc_n  across  Ic_gc_n  through  IO_n    to  GC_ref;
  ------------------------------------------------------------------------------
  -- Differential IV surface and C_comp
  ------------------------------------------------------------------------------
  quantity  V_pn     across  I_pn     through  IO_p    to  IO_n;
  quantity  Vc_diff  across  Ic_diff  through  IO_p    to  IO_n;
  ------------------------------------------------------------------------------
```

VHDL-AMS implementation – changes (4)

```
-------------------------------------------------------------------------
 -- Various signals and quantities (for internal calculations)
 -------------------------------------------------------------------------
 signal    Data_D             : std_logic := Out_ini;
 signal    Data_InvDel_D      : std_logic := Out_ini;

 signal    pu_p_on_0          : std_logic := '0';
 signal    pu_p_off_0         : std_logic := '0';
 signal    pu_n_on_0          : std_logic := '0';
 signal    pu_n_off_0         : std_logic := '0';

 signal    pu_p_on_1          : std_logic := '0';
 signal    pu_p_off_1         : std_logic := '0';
 signal    pu_n_on_1          : std_logic := '0';
 signal    pu_n_off_1         : std_logic := '0';

 signal    Tpu_p_on_event_0   : real := 0.0;
 signal    Tpu_p_off_event_0  : real := 0.0;
 signal    Tpu_n_on_event_0   : real := 0.0;
 signal    Tpu_n_off_event_0  : real := 0.0;

 signal    Tpu_p_on_event_1   : real := 0.0;
 signal    Tpu_p_off_event_1  : real := 0.0;
 signal    Tpu_n_on_event_1   : real := 0.0;
 signal    Tpu_n_off_event_1  : real := 0.0;

 quantity  k_pu_p_0           : real := 0.0;
 quantity  k_pu_n_0           : real := 0.0;

 quantity  k_pu_p_1           : real := 0.0;
 quantity  k_pu_n_1           : real := 0.0;
--=========================================================================
```

VHDL-AMS implementation – changes (5)

```
--============================================================================
  function Lookup (Extrapolation : in string := "IV";
                   X             : in real;
                   Ydata         : in real_vector;
                   Xdata         : in real_vector) return real is
  ----------------------------------------------------------------------------
  ...
--============================================================================
  function Find_common_length (Max_dt : real := 1.0e-12;
                               Twfm_1 : in real_vector;
                               Twfm_2 : in real_vector) return integer is
  ----------------------------------------------------------------------------
  ...
--============================================================================
  function Common_time (Max_dt : real := 1.0e-12;
                        Twfm_1 : in real_vector;
                        Twfm_2 : in real_vector) return real_vector is
  ----------------------------------------------------------------------------
  ...
--============================================================================
  function Common_wfm (New_t : in real_vector;
                       Vwfm  : in real_vector;
                       Twfm  : in real_vector) return real_vector is
  ----------------------------------------------------------------------------
  ...
--============================================================================
  function Coeff (Edge : in string;
                  Vwfm : in real_vector;
                  Twfm : in real_vector;
                  Rfx  : in real;
                  Vfx  : in real;
                  Iiv  : in real_vector;
                  Viv  : in real_vector;
                  Vref : in real) return real_vector is
  ----------------------------------------------------------------------------
  ...
```

VHDL-AMS implementation – changes (6)

```
--=============================================================================
  Clock: process (Clk, En_D) is
  begin
  -----------------------------------------------------------------------------
    if (Clk = '1') and (Clk'LAST_VALUE = '0') then         -- Rising edge
      if (Edge = 1) or (Edge = 2) then
        Data_InvDel_D <= not Data_D;   -- One clock delayed inverted Data_D
        Data_D <= In_D;                -- Clocked input
      end if;
    elsif (Clk = '0') and (Clk'LAST_VALUE = '1') then      -- Falling edge
      if (Edge = 0) or (Edge = 2) then
        Data_InvDel_D <= not Data_D;   -- One clock delayed inverted Data_D
        Data_D <= In_D;                -- Clocked input
      end if;
    end if;
  -----------------------------------------------------------------------------
  end process Clock;
--=============================================================================
  Catch_0: process (Data_D, En_D) is
  begin
    Rcv_D <= Data_D;                                    -- Dummy receiver logic
  -----------------------------------------------------------------------------
    if (En_D = '1') and (Data_D = '1') then             -- Find logic state
      pu_p_on_0  <= '1';
      pu_n_off_0 <= '1';
      pu_n_on_0  <= '0';
      pu_p_off_0 <= '0';
    elsif (En_D = '1') and (Data_D = '0') then
      pu_p_on_0  <= '0';
      pu_n_off_0 <= '0';
      pu_n_on_0  <= '1';
      pu_p_off_0 <= '1';
    else
      pu_p_on_0  <= '0';
      pu_n_off_0 <= '1';
      pu_n_on_0  <= '0';
      pu_p_off_0 <= '1';
    end if;
  -----------------------------------------------------------------------------
  end process Catch_0;
--=============================================================================
```

VHDL-AMS implementation – changes (7)

```
--=============================================================================
  Catch_1: process (Data_InvDel_D, En_D) is
  -----------------------------------------------------------------------------
  begin
  -----------------------------------------------------------------------------
    if (En_D = '1') and (Data_InvDel_D = '1') then    -- Find logic state
      pu_p_on_1  <= '1';
      pu_n_off_1 <= '1';
      pu_n_on_1  <= '0';
      pu_p_off_1 <= '0';
    elsif (En_D = '1') and (Data_InvDel_D = '0') then
      pu_p_on_1  <= '0';
      pu_n_off_1 <= '0';
      pu_n_on_1  <= '1';
      pu_p_off_1 <= '1';
    else
      pu_p_on_1  <= '0';
      pu_n_off_1 <= '1';
      pu_n_on_1  <= '0';
      pu_p_off_1 <= '1';
    end if;
  -----------------------------------------------------------------------------
  end process Catch_1;
--=============================================================================
 pu_p_on_event_time_0: process (pu_p_on_0) is    -- Update event time if changed
  -----------------------------------------------------------------------------
  begin
      Tpu_p_on_event_0 <= now;
  -----------------------------------------------------------------------------
  end process pu_p_on_event_time_0;
--=============================================================================
  pu_p_off_event_time_0: process (pu_p_off_0) is  -- Update event time if changed
  -----------------------------------------------------------------------------
  begin
      Tpu_p_off_event_0 <= now;
  -----------------------------------------------------------------------------
  end process pu_p_off_event_time_0;
--=============================================================================
  ...
```

VHDL-AMS implementation – changes (8)

```
--=============================================================================
  break on pu_p_on_0;
  break on pu_p_off_0;
  break on pu_n_on_0;
  break on pu_n_off_0;

  break on pu_p_on_1;
  break on pu_p_off_1;
  break on pu_n_on_1;
  break on pu_n_off_1;
--=============================================================================
 -- This section contains the simultaneous analog equations to find the
  -- appropriate scaling coefficients according to the state the buffer.
  ------------------------------------------------------------------------------
  if (Tpu_p_on_event_0 = 0.0 and Tpu_p_off_event_0 = 0.0 and
      Tpu_n_on_event_0 = 0.0 and Tpu_n_off_event_0 = 0.0) use
                                                    -- Initialization
    if (pu_p_on_0 = '1') use                        -- Start with the end of the
      k_pu_p_0 == K_pu_on(K_pu_on'right);           -- Vt curves for those which
    elsif (pu_p_off_0 = '1') use                    -- are fully on initially
      k_pu_p_0 == K_pu_off(K_pu_off'right);
    else
      k_pu_p_0 == 0.0;
    end use;

    if (pu_n_on_0 = '1') use
      k_pu_n_0 == K_pu_on(K_pu_on'right);
    elsif (pu_n_off_0 = '1') use
      k_pu_n_0 == K_pu_off(K_pu_off'right);
    else
      k_pu_n_0 == 0.0;
    end use;
```

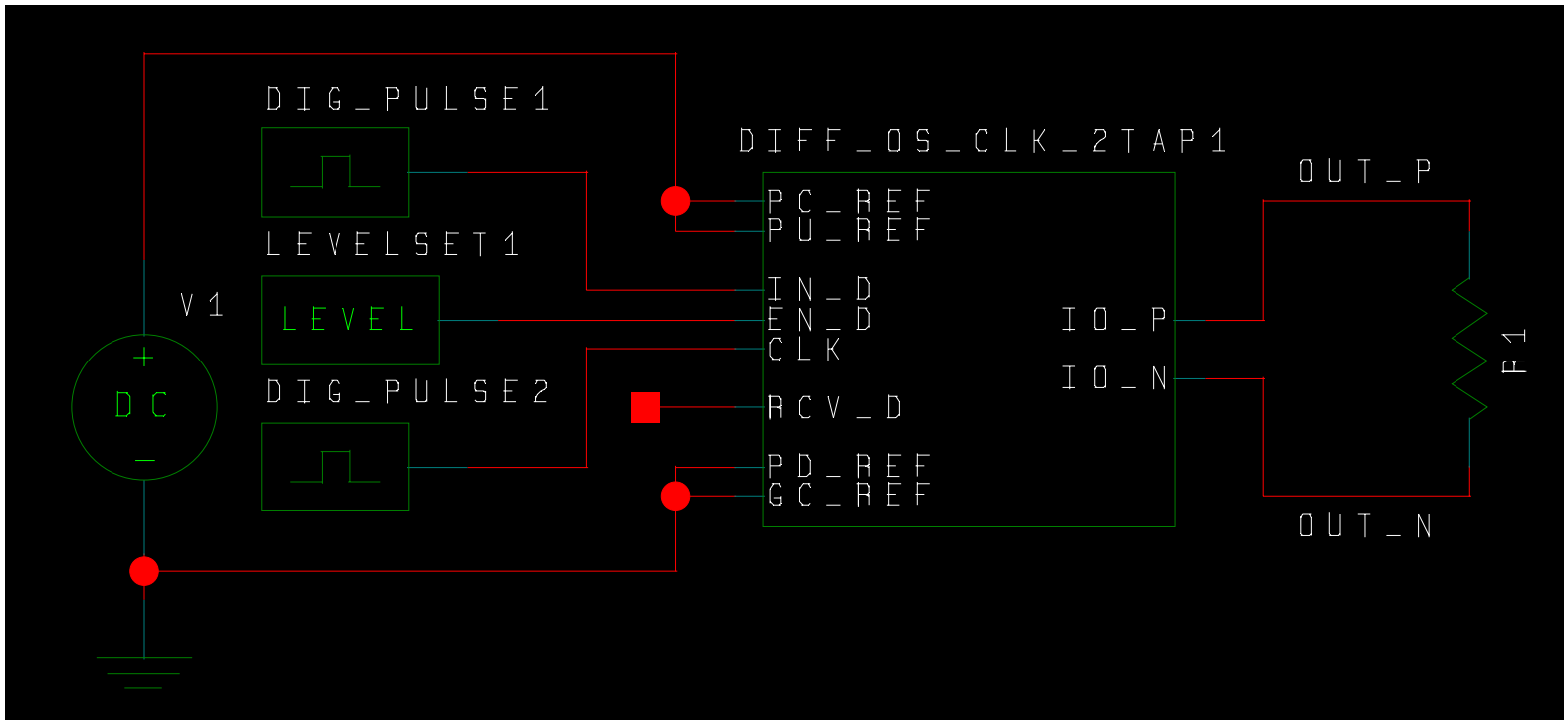
VHDL-AMS implementation – changes (9)

```
  else                                -- Look up coefficients in normal operation

    if (pu_p_on_0  = '1') use
      k_pu_p_0 == Lookup("Vt", now - Tpu_p_on_event_0, K_pu_on,  T_common);
    elsif (pu_p_off_0 = '1') use
      k_pu_p_0 == Lookup("Vt", now - Tpu_p_off_event_0, K_pu_off, T_common);
    else
      k_pu_p_0 == K_pu_on(K_pu_on'left);
    end use;

    if (pu_n_on_0  = '1') use
      k_pu_n_0 == Lookup("Vt", now - Tpu_n_on_event_0, K_pu_on,  T_common);
    elsif (pu_n_off_0 = '1') use
      k_pu_n_0 == Lookup("Vt", now - Tpu_n_off_event_0, K_pu_off, T_common);
    else
      k_pu_n_0 == K_pu_on(K_pu_on'left);
    end use;

  end use;
--=============================================================================
  -- This section contains the simultaneous analog equations to find the
  -- appropriate scaling coefficients according to the state the buffer.
  ------------------------------------------------------------------------------
  if (Tpu_p_on_event_1 = 0.0 and Tpu_p_off_event_1 = 0.0 and
      Tpu_n_on_event_1 = 0.0 and Tpu_n_off_event_1 = 0.0) use
                                                    -- Initialization
    if (pu_p_on_1 = '1') use                        -- Start with the end of the
      k_pu_p_1 == K_pu_on(K_pu_on'right);           -- Vt curves for those which
    elsif (pu_p_off_1 = '1') use                    -- are fully on initially
      k_pu_p_1 == K_pu_off(K_pu_off'right);
    else
      k_pu_p_1 == 0.0;
    end use;

    if (pu_n_on_1 = '1') use
      k_pu_n_1 == K_pu_on(K_pu_on'right);
    elsif (pu_n_off_1 = '1') use
      k_pu_n_1 == K_pu_off(K_pu_off'right);
    else
      k_pu_n_1 == 0.0;
    end use;
  ...
```

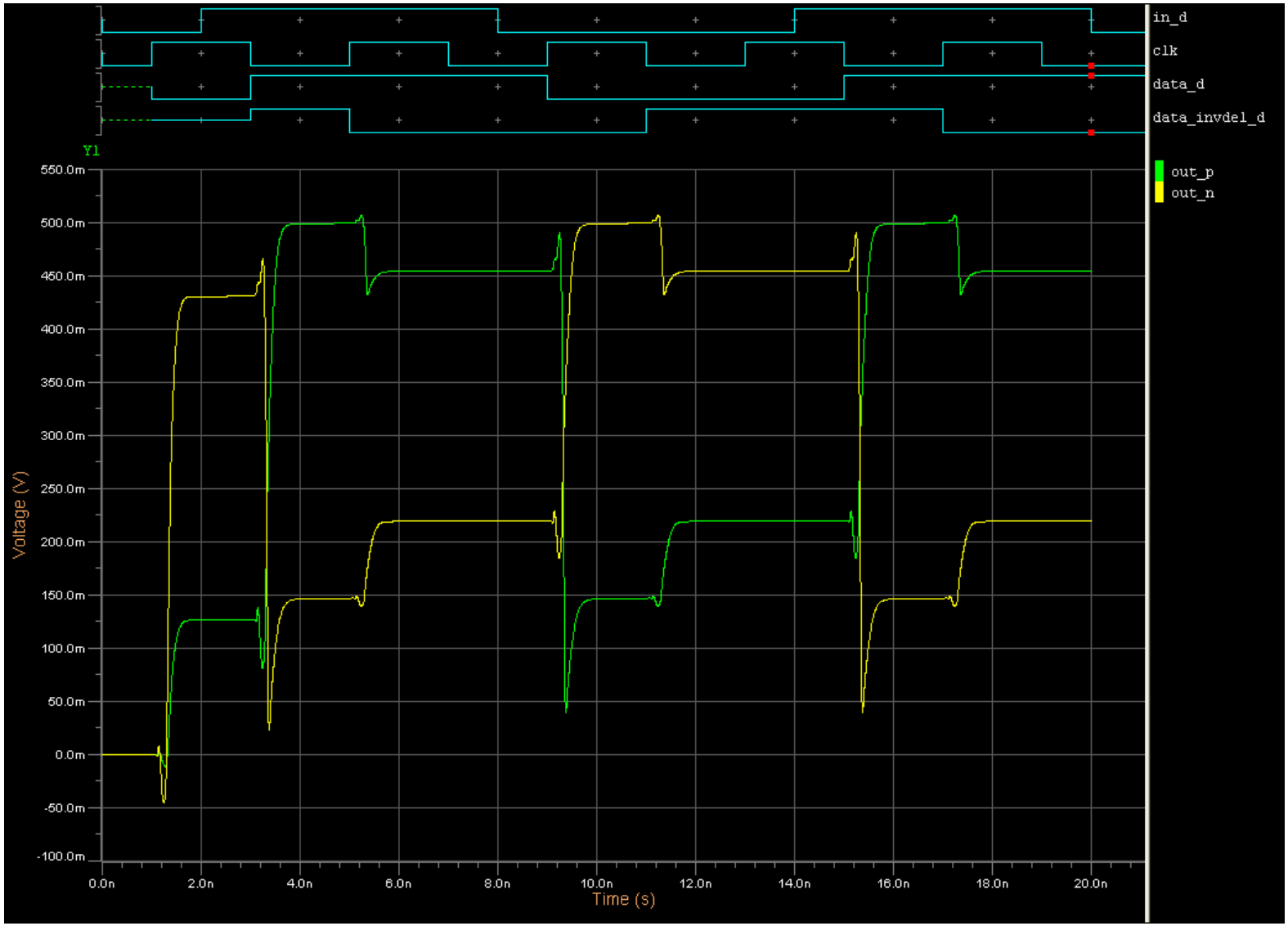
VHDL-AMS implementation – changes (10)

```
  ------------------------------------------------------------------------
  -- Common mode components for IV curves
  ------------------------------------------------------------------------
  Ipc_p_0   == -1.0            * Lookup("IV", Vpc_p_0, I_pc, V_pc);
  Ipu_p_0   == -1.0 * k_pu_p_0 * Lookup("IV", Vpu_p_0, I_pu, V_pu);
  Igc_p_0   ==                   Lookup("IV", Vgc_p_0, I_gc, V_gc);

  Ipc_n_0   == -1.0            * Lookup("IV", Vpc_n_0, I_pc, V_pc);
  Ipu_n_0   == -1.0 * k_pu_n_0 * Lookup("IV", Vpu_n_0, I_pu, V_pu);
  Igc_n_0   ==                   Lookup("IV", Vgc_n_0, I_gc, V_gc);

  Ipc_p_1   == -0.2            * Lookup("IV", Vpc_p_1, I_pc, V_pc);
  Ipu_p_1   == -0.2 * k_pu_p_1 * Lookup("IV", Vpu_p_1, I_pu, V_pu);
  Igc_p_1   ==  0.2            * Lookup("IV", Vgc_p_1, I_gc, V_gc);

  Ipc_n_1   == -0.2            * Lookup("IV", Vpc_n_1, I_pc, V_pc);
  Ipu_n_1   == -0.2 * k_pu_n_1 * Lookup("IV", Vpu_n_1, I_pu, V_pu);
  Igc_n_1   ==  0.2            * Lookup("IV", Vgc_n_1, I_gc, V_gc);
  ------------------------------------------------------------------------
  -- Common mode components for C_comp
  ------------------------------------------------------------------------
  Ic_pc_p == k_C_comp_pc * C_comp * Vc_pc_p'dot;
  Ic_pu_p == k_C_comp_pu * C_comp * Vc_pu_p'dot;
  Ic_pd_p == k_C_comp_pd * C_comp * Vc_pd_p'dot;
  Ic_gc_p == k_C_comp_gc * C_comp * Vc_gc_p'dot;

  Ic_pc_n == k_C_comp_pc * C_comp * Vc_pc_n'dot;
  Ic_pu_n == k_C_comp_pu * C_comp * Vc_pu_n'dot;
  Ic_pd_n == k_C_comp_pd * C_comp * Vc_pd_n'dot;
  Ic_gc_n == k_C_comp_gc * C_comp * Vc_gc_n'dot;
  ------------------------------------------------------------------------
  -- Differential IV surface and C_comp
  ------------------------------------------------------------------------
  I_pn    == k0 + k1*Vpd_p_0 + k2*Vpd_n_0 + k3*Vpd_p_0*Vpd_n_0 + k4*(Vpd_p_0**2) + k5*(Vpd_n_0**2);
--  I_pn    == V_pn / R_diff;       -- In case a linear resistor does the job
--  I_pn == 0.0;                    -- In case we don't want differential currents

  Ic_diff == C_diff * Vc_diff'dot;
--========================================================================
end architecture DIFF_OS_CLK_2TAP_1EQ;
--========================================================================
```

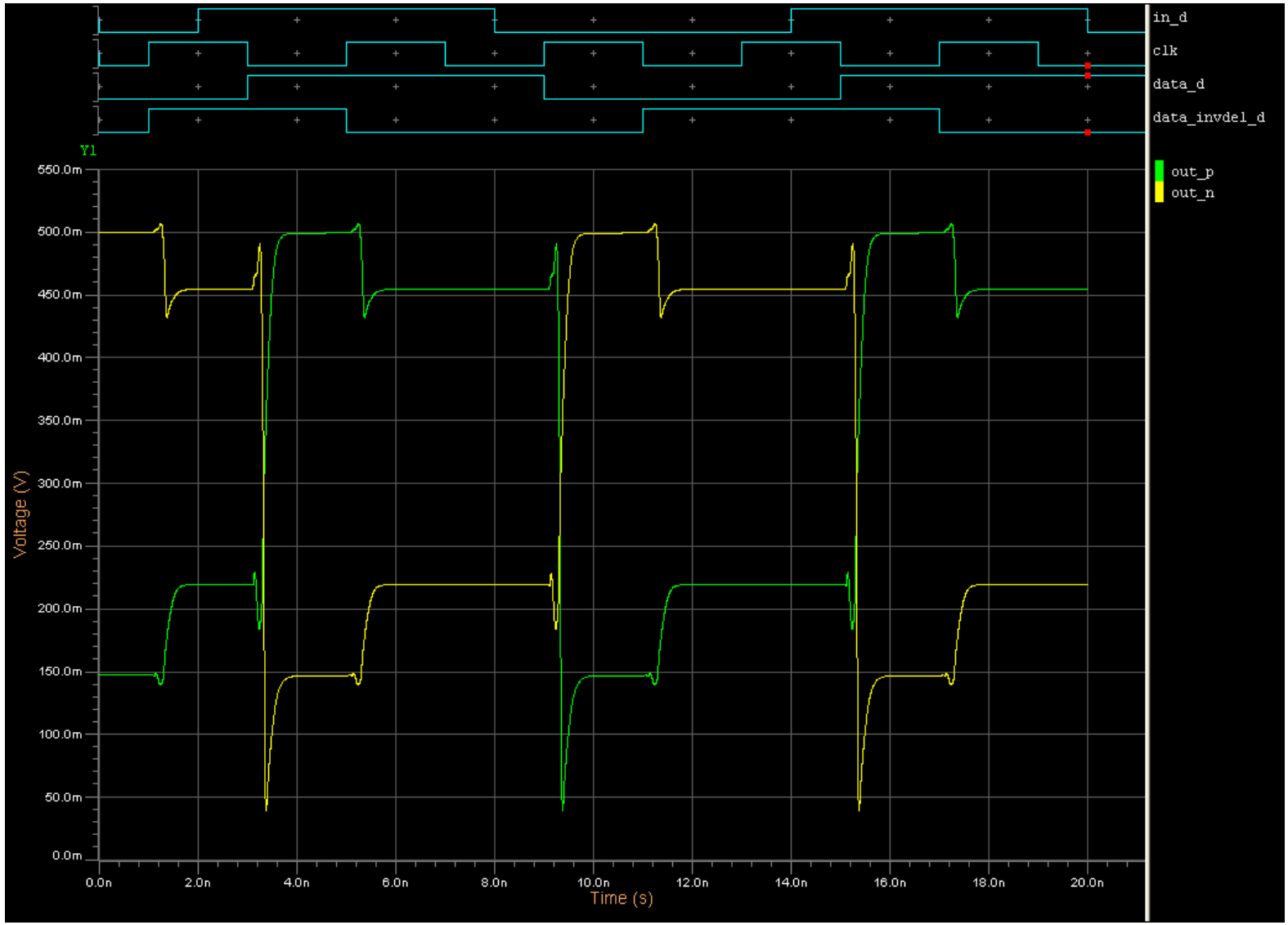
Simulation schematics



Waveform example - Out_ini = ‘Z’



Waveform example - Out_ini = ‘0’



Summary

- **A Pre/De-emphasis VHDL-AMS buffer model has been shown as a modification of the previously introduced “true differential” I/O model**
  - http://www.eda.org/pub/ibis/summits/apr04/IBIS_PreDe.vhd
  - Feel free to download and use the file any way you want
- **The modified model is a complete Pre/De-emphasis model**
  - It has only one digital input, a clock and a differential pair of analog I/O port
  - This model can be used with [External Model] in IBIS v4.1 (not tested yet)
- **The C_comp compensation algorithm needs more work**
  - The effects of the two buffer blocks loading each other must be accounted for
- **The model needs to be correlated and tested with a SPICE level model**