IBIS 4.1 Macros for Simulator Independent Models

DAC2005 IBIS Summit
Anaheim, CA
June 14, 2005

Arpad Muranyi
Barry Katz, Mike LaBonte, Scott Mc Morrow,
Donald Telian, Todd Westerhoffer, Ken Willis
Background

• At the January 31, 2005 IBIS Open Forum Summit Donald Telian (Cadence) presented: “Modeling Complex IO with IBIS 4.1”
• The subject on the macro modeling idea was repeated at various meetings since then:
  • IBIS Summit (March 11, 2005)
  • Cadence webinar (March 23, 2005)
  • SPI Workshop (May 10-13, 2005)
• The proposal of the presentation generated vigorous discussions on the IBIS email reflectors and at various meetings

http://www.eda.org/ibis/summits/jan05/telian.pdf
http://www.eda.org/ibis/futures/ibis-futures-issues-mm.pdf
The real problem

- **IBIS is running out of steam for advanced buffer modeling**
  - took a long time to get [Driver Schedule] to work for pre/de-emphasis buffers
  - other types of advanced buffers may follow soon
  - HSPICE (and others) allow tricks around B-element
  - strict IBIS simulators can’t do anything about it
  - Cadence can do similar tricks in DML (K-SPICE)

- Many SI engineers don’t want to fiddle with tricks, so they prefer transistor level models
  - amplified by the common belief that transistor models are more accurate than behavioral models

- The *-AMS extensions in IBIS are too advanced
  - learning a new language is a deterrent factor to many
  - not too many board level SI simulators support it yet
  - few *-AMS models exist
Cadence’s SPICE macro model proposal

- Write the tricks that HSPICE and K-SPICE can do in Berkeley SPICE
  - reason: Berkeley SPICE is one of the approved language extensions of IBIS 4.1
  - problem: Berkeley SPICE is very limited with its E, F, G, H elements to be useful for anything
- Proposal: add the missing features of Berkeley SPICE to the IBIS specification
  - Berkeley SPICE stopped being developed in 1993
- This would allow model makers to write macro models for advanced buffer types that cannot be modeled by IBIS keywords
SPICE macro model pros and cons

- This would be like standardizing SPICE
  - wouldn’t be a bad idea after 20+ years...
  - whose syntax should it be?
  - would Synopsys allow IBIS to publish the HSPICE syntax?
  - Cadence may be willing to “donate” K-SPICE
  - K-SPICE has no transistor model capabilities
  - are all tool vendors going to want to implement K-SPICE?
- SPICE macro modeling provides useful solutions for the short term, but
- The proposal requires BIRDs for the IBIS spec
  - usually slow process
  - the faster we want to do it the fewer the features will be, and the more often BIRDs will need to be written
- Delays the already slow acceptance of *-AMS
Vendor neutral SPICE syntax?

- An idea surfaced during discussions to invent a vendor neutral SPICE syntax
  - to avoid favoritism with any vendor
  - to make the amount implementation effort about the same for each vendor
  - to avoid any copyright issues
- This will not solve any technical implementation issues
  - the “neutral” SPICE syntax may have certain features which exist in one tool, but not another, or
  - some features may resemble one tool’s syntax more than another tool’s syntax
- It would take a considerable amount of time to write such a SPICE syntax from scratch
  - remember the IBIS-X efforts?
A new idea

- Why not use the analog subset of the *-AMS languages for macro modeling?
- The macro model will have to be netlisted in Verilog-AMS or VHDL-AMS
  - these netlists are very much SPICE-like
- Hide the equations in a library of building blocks
  - develop a set of SPICE compatible *-A(MS) building blocks (E, F, G, H elements)
  - the user of these building blocks doesn’t have to know about the underlying equations or *-AMS syntax
  - SPICE tools could substitute their equivalent elements
- No changes are required in the IBIS spec
  - [External Circuit] or [External Model] can be used to instantiate these macro models from IBIS
  - this can be done as we speak
  - helps the adoption of *-AMS in general
VHDL-A(MS) or Verilog-A(MS)?

- Several SPICE tools are starting to implement Verilog-A
  - HSPICE 2005.03, and others...
- Some tools have full *-AMS support
- Verilog-A is IBIS compatible, since it is a subset of Verilog-AMS
  - tools usually implement more than just pure Verilog-A
  - “AMS” models do not have to include mixed signal or digital constructs
- Verilog-A(MS) is more similar to SPICE
  - but the specification is quite loose
- VHDL-A(MS) is a more robust specification
  - a little less human readable, less SPICE-like
New proposal – part 1

- Develop a **standard** library in Verilog-A(MS)
  - a complete set of “modules” (building blocks) to be used for macro modeling
  - a little more widely supported than VHDL-A(MS)
- If needed, a corresponding and compatible library in VHDL-A(MS) could also be developed
  - there is no technical reason that this couldn’t be done
- Participation in library development welcome
  - tool vendors preferred with strong commitment
  - the effort must be coordinated to achieve consistent and coherent content
- A prototype library could be targeted for the next IBIS summit at DesignCon East, September 2005
New proposal – part 2

- Develop a set of macro models (templates) for commonly used and well known buffer types
  - impedance compensated buffers, true differential, pre/de-emphasis buffers, buffers with deterministic jitter insertion, DFE, FIR equalized receiver models, etc...

- This macro model library would not have to be standardized
  - contains only a netlist of building block instances

- This macro model library may be extended by anyone at any time
Macromodel Hierarchy

IBIS File

[External Circuit] or [External Model] call macro model templates

Macro model templates call building blocks from standard library

Building blocks are written using the analog only features of the *-AMS languages, and can be substituted with native SPICE elements in SPICE tools if necessary
A macro model example

- A differential pre/de-emphasis buffer
  - a circuit netlist serves as the macro model, instantiating
    - four Verilog-A or VHDL-AMS IBIS I/O buffer models,
    - an inverter,
    - two ideal delays, and
    - eight current sources to scale the Boost buffer’s current

Diagram borrowed from M. Mirmak
The HSPICE macro model

```
.SUBCKT PreDe_IO In_D En_D IO_P IO_N PC_ref PU_ref PD_ref GC_ref
+ TDelay  = 10.0e-9;
+ ScaleBoost = -0.5;
*
XPosM In_D IO_P PU_ref PC_ref PD_ref GC_ref En_D IO_buf
XNegM In_NM IO_N PU_ref PC_ref PD_ref GC_ref En_D IO_BUF
XPosB In_PB IO_P PU_refPB PC_refPB PD_refPB GC_refPB En_D IO_buf
XNegB In_NB IO_N PU_refNB PC_refNB PD_refNB GC_refNB En_D IO_buf
*
Einvl In_NM PD_ref VCVS PU_ref In_D 1
Edlyl In_NB PD_ref VCVS DELAY In_D PD_ref TD=TDelay
Edly2 In_PB PD_ref VCVS DELAY In_NM PD_ref TD=TDelay
*
VpcP PC_ref PC_refPB DC=0
VpuP PU_ref PU_refPB DC=0
VpdP PD_ref PD_refPB DC=0
VgcP GC_ref GC_refPB DC=0
*
VpcN PC_ref PC_refNB DC=0
VpuN PU_ref PU_refNB DC=0
VpdN PD_ref PD_refNB DC=0
VgcN GC_ref GC_refNB DC=0
*
FpcP PC_ref IO_P CCCS VpcP ScaleBoost
FpuP PU_ref IO_P CCCS VpuP ScaleBoost
FpdP PD_ref IO_P CCCS VpdP ScaleBoost
FgcP GC_ref IO_P CCCS VgcP ScaleBoost
*
FpcN PC_ref IO_N CCCS VpcN ScaleBoost
FpuN PU_ref IO_N CCCS VpuN ScaleBoost
FpdN PD_ref IO_N CCCS VpdN ScaleBoost
FgcN GC_ref IO_N CCCS VgcN ScaleBoost
*
.ENDS
```
module PreDe_IO (In_D, En_D, IO_P, IO_N, PC_ref, PU_ref, PD_ref, GC_ref);
    input      In_D, En_D;
    electrical In_D, En_D;
    inout     IO_P, IO_N, PC_ref, PU_ref, PD_ref, GC_ref, PC_refPB, PU_refPB, PD_refPB, GC_refPB;
    electrical IO_P, IO_N, PC_ref, PU_ref, PD_ref, GC_ref, PC_refPB, PU_refPB, PD_refPB, GC_refPB;
    parameter real TDelay = 10.0e-9;
    parameter real ScaleBoost = -0.5;

IO_buffer PosM (In_D, En_D, Rcv_PM, IO_P, PC_ref, PU_ref, PD_ref, GC_ref);
IO_buffer NegM (In_NM, En_D, Rcv_NM, IO_N, PC_ref, PU_ref, PD_ref, GC_ref);
IO_buffer PosB (In_PB, En_D, Rcv_PB, IO_P, PC_refPB, PU_refPB, PD_refPB, GC_refPB);
IO_buffer NegB (In_NB, En_D, Rcv_NB, IO_N, PC_refNB, PU_refNB, PD_refNB, GC_refNB);

Inverter Inv1 (In_D, In_NM, PU_ref, PD_ref);
Delay #(.DelayTime(TDelay))  Dly1 (In_D, In_NB, PD_ref);
Delay #(.DelayTime(TDelay))  Dly2 (In_NM, In_PB, PD_ref);

Isource #(M(ScaleBoost))  IpcP (PC_ref, IO_P, PC_ref, PC_refPB);
Isource #(M(ScaleBoost))  IpuP (PU_ref, IO_P, PU_ref, PU_refPB);
Isource #(M(ScaleBoost))  IpdP (PD_ref, IO_P, PD_ref, PD_refPB);
Isource #(M(ScaleBoost))  IgpP (GC_ref, IO_P, GC_ref, GC_refPB);
Isource #(M(ScaleBoost))  IpcN (PC_ref, IO_N, PC_ref, PC_refNB);
Isource #(M(ScaleBoost))  IpuN (PU_ref, IO_N, PU_ref, PU_refNB);
Isource #(M(ScaleBoost))  IpdN (PD_ref, IO_N, PD_ref, PD_refNB);
Isource #(M(ScaleBoost))  IgpN (GC_ref, IO_N, GC_ref, GC_refNB);
endmodule
The Verilog-A library (1)

```verilog
module Inverter (In, Out, Pref, Gref);
    input    In, Pref, Gref;
    electrical In, Pref, Gref;
    output   Out;
    electrical Out;

    analog begin
        V(Out, Gref) <+ V(Pref, In);
    end
endmodule
```

```verilog
module Delay (In, Out, Gref);
    input    In, Gref;
    electrical In, Gref;
    output   Out;
    electrical Out;

    parameter real DelayTime = 0.0 from [0:inf];

    analog begin
        V(Out, Gref) <+ absdelay(V(In, Gref), DelayTime);
    end
endmodule
```
module Isource (OutP, OutN, SenseP, SenseN);
  input    SenseP, SenseN;
  electrical SenseP, SenseN;
  output   OutP, OutN;
  electrical OutP, OutN;

  parameter real M = 1.0;

  analog begin
    V(SenseP, SenseN) <+ 0.0;
    I(OutP, OutN)    <+ M * I(SenseP, SenseN);
  end
endmodule

module IO_buffer (In_D, En_D, Rcv_D, IO, PC_ref, PU_ref, PD_ref, GC_ref);
...
...
...
endmodule
The VHDL-AMS macro model (1)

```
-- genhdl\predeemphasis/predeemphasis.vhd
-- Generated by SystemVision netlister 1.0 build 2005.25.1_SV
-- File created Mon Jun 06 11:26:00 2005

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.electrical_systems.all;
LIBRARY edulib;
USE work.all;

entity PREDEEMPHASIS is
end entity PREDEEMPHASIS;

architecture arch_PREDEEMPHASIS of PREDEEMPHASIS is
  signal RCV_D_PM: STD_LOGIC;
  signal IN_BN: STD_LOGIC;
  terminal GCREF_PB: ELECTRICAL;
  terminal VCC: ELECTRICAL;
  terminal PUREF_NB: ELECTRICAL;
  signal RCV_D_NM: STD_LOGIC;
  signal IN_BP: STD_LOGIC;
  terminal GCREF_NB: ELECTRICAL;
  terminal PDREF_PB: ELECTRICAL;
  terminal PCREF_PB: ELECTRICAL;
  signal RCV_D_PB: STD_LOGIC;
  terminal PDREF_NB: ELECTRICAL;
  terminal PCREF_NB: ELECTRICAL;
  signal IN_MN: STD_LOGIC;
  signal RCV_D_NB: STD_LOGIC;
  signal IN_MP: STD_LOGIC;
  terminal OUT_N: ELECTRICAL;
  terminal OUT_P: ELECTRICAL;
  terminal PUREF_PB: ELECTRICAL;
  signal EN_D: STD_LOGIC;

begin
```
The VHDL-AMS macro model (2)

MAIN_P : entity WORK.IBIS_IO(IO_2EQ)
generic map ( C_COMP => 1.2E-12 )
port map ( IN_D => IN_MP,
            EN_D => EN_D,
            RCV_D => RCV_D_PM,
            IO => OUT_P,
            PC_REF => VCC,
            PU_REF => VCC,
            PD_REF => ELECTRICAL_REF,
            GC_REF => ELECTRICAL_REF );

MAIN_N : entity WORK.IBIS_IO(IO_2EQ)
generic map ( C_COMP => 1.2E-12 )
port map ( IN_D => IN_MIN,
            EN_D => EN_D,
            RCV_D => RCV_D_NM,
            IO => OUT_N,
            PC_REF => VCC,
            PU_REF => VCC,
            PD_REF => ELECTRICAL_REF,
            GC_REF => ELECTRICAL_REF );

BOOST_P : entity WORK.IBIS_IO(IO_2EQ)
generic map ( C_COMP => 1.2E-12 )
port map ( IN_D => IN_BP,
           EN_D => EN_D,
           RCV_D => RCV_D_PB,
           IO => OUT_P,
           PC_REF => PCREF_PB,
           PU_REF => PUREF_PB,
           PD_REF => PDREF_PB,
           GC_REF => GCREF_PB );

BOOST_N : entity WORK.IBIS_IO(IO_2EQ)
generic map ( C_COMP => 1.2E-12 )
port map ( IN_D => IN_BN,
           EN_D => EN_D,
           RCV_D => RCV_D_NB,
           IO => OUT_N,
           PC_REF => PCREF_NB,
           PU_REF => PUREF_NB,
           PD_REF => PDREF_NB,
           GC_REF => GCREF_NB );
The VHDL-AMS macro model (3)

IPCP : entity WORK.CCCS(IDEAL)
generic map ( GAIN => -0.5 )
port map ( OUT_P => VCC,
OUT_N => OUT_P,
IN_P => VCC,
IN_N => PCREF_PB );

IPUP : entity WORK.CCCS(IDEAL)
generic map ( GAIN => -0.5 )
port map ( OUT_P => VCC,
OUT_N => OUT_P,
IN_P => VCC,
IN_N => PUREF_PB );

IPDP : entity WORK.CCCS(IDEAL)
generic map ( GAIN => -0.5 )
port map ( OUT_P => ELECTRICAL_REF,
OUT_N => OUT_P,
IN_P => ELECTRICAL_REF,
IN_N => PEREF_PB );

IGCP : entity WORK.CCCS(IDEAL)
generic map ( GAIN => -0.5 )
port map ( OUT_P => ELECTRICAL_REF,
OUT_N => OUT_P,
IN_P => ELECTRICAL_REF,
IN_N => GCREF_PB );
The VHDL-AMS macro model (4)

IPCN : entity WORK.CCCS(IDEAL)
generic map ( GAIN => -0.5 )
port map ( OUT_P => VCC,
          OUT_N => OUT_N,
          IN_P => VCC,
          IN_N => PCREF_NB );

IPUN : entity WORK.CCCS(IDEAL)
generic map ( GAIN => -0.5 )
port map ( OUT_P => VCC,
          OUT_N => OUT_N,
          IN_P => VCC,
          IN_N => PUREF_NB );

IPDN : entity WORK.CCCS(IDEAL)
generic map ( GAIN => -0.5 )
port map ( OUT_P => ELECTRICAL_REF,
          OUT_N => OUT_N,
          IN_P => ELECTRICAL_REF,
          IN_N => PDREF_NB );

IGCN : entity WORK.CCCS(IDEAL)
generic map ( GAIN => -0.5 )
port map ( OUT_P => ELECTRICAL_REF,
          OUT_N => OUT_N,
          IN_P => ELECTRICAL_REF,
          IN_N => GCREF_NB );
R1 : entity EDULIB.RESISTOR(Ideal)
generic map ( RES => 100.0 )
port map ( P1 => OUT_N,
P2 => OUT_P );

V1 : entity EDULIB.V_CONSTANT(Ideal)
generic map ( LEVEL => 5.0 )
port map ( POS => VCC,
           NEG => ELECTRICAL_REF );

LEVELSET1 : entity EDULIB.LEVELSET
port map ( LEVEL => EN_D );

DIG_PULSE1 : entity EDULIB.DIG_PULSE(Ideal)
generic map ( INITIAL_DELAY => 0.1NS,
             PERIOD => 50NS )
port map ( OUT_STATE => IN_MP );

INVERTER1 : entity EDULIB.INVERTER
port map ( INPUT => IN_MP,
          OUTPUT => IN_MN );

DELAY_1 : entity WORK.DIG_DELAY(Ideal)
generic map ( DELAY => 10.0E-9 SEC,
             IC_OUT => '1' )
port map ( INPUT => IN_MN,
          OUTPUT => IN_BP );

DELAY_2 : entity WORK.DIG_DELAY(Ideal)
generic map ( DELAY => 10.0E-9 SEC,
             IC_OUT => '0' )
port map ( INPUT => IN_MP,
          OUTPUT => IN_BN );
end architecture arch_PREDEEMPHASIS;
Waveform overlay

![Waveform overlay graph]

- HSPICE_tr_pls
- HSPICE_tr_p
- HSPICE_tr_n
- HSPICE_IBIS_p
- HSPICE_IBIS_n
- Verilog_A_p
- Verilog_A_n
- VHDL_AMS_p
- VHDL_AMS_n
Summary

- Macro modeling with "AMS is possible now without any changes to the IBIS specification
  - a standard, analog only "AMS library could be developed so that SPICE tools can substitute with their own elements if they don’t support the "AMS language directly
- Verilog-A(MS) netlists are very SPICE-like
  - very easy to write macro model netlists by hand
- VHDL-A(MS) netlists are also similar to SPICE
  - somewhat more cumbersome to write by hand
- A standard building block library will reduce the model developer’s learning curve and speed adoption of "AMS
- Simulation results match well
Call to action

- Soliciting for participation
  - need tool vendor’s participation to specify and define the building blocks for the standard library
  - participation from SPICE tool vendors is needed to make sure that these building blocks can be mapped to their elements

- Need to find a mechanism for the distribution and revision control of the standard library
  - the standard library needs strict control over its content to ensure that it can be supported widely

- Need to find a distribution channel for the macro model templates
  - could be done through the IBIS web site
  - no need for an official standards body, like ANSI, etc...