The *-AMS experience

IBIS Summit, DAC 2007

June 5, 2007

Arpad Muranyi

Signal Integrity Engineering

Intel Corporation

arpad.muranyi@intel.com

Outline

- IBIS / *-AMS background
- Problems
- Current situation
- Strengths and weaknesses of *-AMS and other languages
- Looking for solutions
- Need to make a decision

*Other brands and names are the property of their respective owners

Background

- The VHDL-AMS and Verilog-AMS language extensions were first introduced in the IBIS v4.1 specification on January 30, 2004
- The expectation was that they will solve the biggest problem in IBIS: RIGIDITY due to assumed algorithms on how the data is interpreted or used by the simulation tools.
- Reminder: IBIS files contain data only, algorithms are defined and implemented by the tool vendors.
- New types of behaviors required new keywords and changes to the specification, a lengthy process.

There was hope

- **The VHDL-AMS and Verilog-AMS language extensions were expected to provide the means for a model maker to write their own algorithms.**
- **This should have eliminated the need for any new keywords since the *-AMS models can describe the device behavior as well as the data that is needed for it (optionally reading the data from external files, including IBIS files).**

IBIS + AMS = data + algorithm

Problems

- **Implementing the *-AMS languages in simulator tools is not easy ($$$).**

- **Small companies would have to start from scratch.**

- **Large companies usually do not offer the *-AMS languages in their lower cost SI tools.**
  - the more expensive IC design tools usually have it
  - depends on business model and product line

- **The languages are unfamiliar to most SI engineers.**

- **New IBIS keywords (in the old style) are still being requested.**
  - these are mostly for analog features, not SERDES related algorithmic modeling features

More problems

- **There is an ongoing push for filling the gaps of Berkeley-SPICE in IBIS.**

- **This is essentially calling for a standardized SPICE under the IBIS umbrella.**

- **The attempt to fill this need with the *-AMS Macro Model Library didn't seem to succeed.**

- **The Macro Model Library may be useful for Pre/De-emphasis buffers, but it is insufficient for SI work with high speed SERDES buffers and/or algorithmic modeling.**

- **SERDES experts *believe* that the *-AMS languages are insufficient for SERDES buffers and algorithmic modeling.**

*Other brands and names are the property of their respective owners

Current state of affairs

- *-AMS is not gaining too much momentum in the SI world.
- Most SERDES modeling experts use other languages (C, Matlab* by The MathWorks, Inc., etc…).
- Participants of the IBIS-ATM group are working on a non-AMS API proposal (BIRD) for IBIS to address the SERDES modeling needs.
- HDL languages are inadequate to describe hardware (?!?)
  - traditionally these languages were used on transistor level
  - IBIS seems to want to apply it to system or PCB level work
- General Purpose languages do not have built in hardware concepts for model/algorithm writers.
  - nodes, terminals
  - branch currents, voltages, etc…

*Other brands and names are the property of their respective owners

The Matlab* story

- Matlab* is extremely popular among a vide variety of disciplines: electronics, physics, chemistry, geology, biology, finance, stock market, you name it…

- Why? Intuitive language, doesn't bog down the user with unnecessary computer science requirements and hurdles.

- How is this done? They have a vast amount of toolboxes on top of a C-like language.

- The missing piece: Despite Simulink*, Matlab* has no direct SI and PCB-level simulation capabilities.

- The same is true for C, it didn't become popular until the various libraries came along (MFC, .NET, etc…), but these libraries are still not as easy to use as the Matlab* toolboxes, and you need to write your own simulator...

*Other brands and names are the property of their respective owners

Technology supposed to make our life easier

- **Keep in mind, most engineers and scientists *enjoying* Matlab* are not computer scientists!**

- **In other words, they do not want to have to worry about constructors, destructors, memory allocations, inheritance, declarations, marshaling, type casting, etc..., they want to solve their own engineering problems, which is hard enough in itself.**

- **Computers, programming languages, (and simulation tools) supposed to help to make our jobs easier, not harder.**

- **Example: Verilog-AMS does not have an "sgn" function.**
  Yes, it is easy to write an IF/ELSE statement to return a +1, 0, or a -1, but why put this burden on the user of the language? His/her mind was geared to solve a problem that needed "sgn" and not to figure out how to code the "sgn" function…

*Other brands and names are the property of their respective owners

Looking for solutions

- Using the Matlab* philosophy, we could develop easy to use “toolboxes”, function libraries built on the basic capabilities of the *-AMS languages.

- Are the *-AMS languages powerful enough to address all the system level hardware modeling needs?

- If not, can the API-s of the *-AMS languages be utilized to shoehorn user written functions into the languages?

- Do we need to approach the language workgroups to request the necessary improvements?
  (Are they going to be responsive, interested)?

- Or should we pick (an)other language(s) for modeling? If so, which one would work the best?

*Other brands and names are the property of their respective owners

GP programming languages and modeling

- **General purpose programming languages do not come with built in concepts of electronics.**
  - no wires, nets, nodes, terminals, branches
  - no node voltages, branch currents
  - no relationships between charge, current, voltage, etc…

  (These are the strength of the *-AMS languages)

- **These concepts could be established in a special “electronics library” written for simulation and modeling purposes.**

- **If such a library doesn’t exist, should someone create one for the EDA industry?**

- **If so, this should preferably be a standardized library that works across all simulators.**

*Other brands and names are the property of their respective owners

Decision to make

- **Simulation tools have a good tie with *-AMS, but the *-AMS languages and/or model development environments are lacking.**
  - EDA vendors could offer independent model development environments as separate products from the simulators
  - lower cost simulators could use (pre compiled) models developed under different model development tools

- **The C, Matlab*, etc… languages are much more sophisticated, have excellent development environments, but have poor ties with hardware simulation environments.**

- **We must either improve the *-AMS picture, or import the C, Matlab*, etc… capabilities into the simulation tools (with user friendly hardware description extensions).**

*Other brands and names are the property of their respective owners