Compatibility of Models Among Simulators Facilitated by Verilog-A Technique

Modelithics, inc.
Larry Dunleavy
Jiang Liu
Eric O’Dell

MOS-AK Workshop, Berkeley California, December 7, 2016
• Introduction to Modelithics and multi-simulator compatibility problem
• Model development flows using custom API’s vs. Verilog-a & limitations
• Application to customized non-linear models
• A GaN HEMT model example
• Closing comments
Modelithics addresses the industry need for high-accuracy RF/Microwave simulation models that:

- Are standardized across various vendors
- Offer enhanced scalability features
- Provide significant savings versus in-house modeling

Modelithics primary business offerings/expertise:

- Sell and support premium active and passive model libraries (over 12,000 COTS devices represented!)
- Perform custom measurement and modeling services for RF, microwave and mm-wave devices

Highly scalable and accurate models supporting multiple vendors on multiple platforms
• There are lots of RF and microwave CAE software platforms in the market that cater to specific interests of designers
  – Keysight ADS & Genesys, NI AWR DE, Ansys Designer/HFSS, Cadence Spectre, Pspice, Sonnet
  – Budget, familiarity, type of designs, PDK availability, etc.

• It is a tough task for model developers to develop different model versions for all supported simulators with consistent performance
  – Adapting the same model to each different EDA tool API’s is time and labor-intensive!
• A “write-once, run-everywhere” modeling approach would be ideal scenario for developers as they can focus more on the model performance less on API differences
• The Verilog-a modeling language has potential in this regard as more and more CAE vendors adopt a “Verilog-A path” in their products
Typical Development Procedure using Built-in APIs

Model Parameter Registration

Inform the simulator what parameters will be passed into the model (*simulator specific*)

Nodal Y Matrix Filling

Assemble the MNA (modified nodal analysis) matrix by stamping all the elements used in the model netlist to the matrix (*simulator specific*)

Math Model Implementation

Add the mathematic expression of the model to the code (*can be generalized to share across simulators*)

Model Compilation / Debugging

Compile the model code and test the model object to make sure it matches with expectation (*simulator specific*)

PDK packaging / Documentation

Prepare the PDK delivery (*simulator specific*)

This process has to be repeated for every simulator supported: time and labor intensive
Currently Modelithics highly scalable “Microwave Global™” passive RLC models (RLCs) based on the built-in APIs approach for multiple CAE platforms, including:

- Keysight’s ADS and Genesys
- NI AWR Design Environment (aka Microwave Office)
- Cadence’s Virtuoso
- Ansys’ HFSS
- Sonnet

We are able to divide the model codes into general and simulator-specific modules so we can focus on model performance instead of implementations

(But we have come a long way to get to this point! )
Verilog-a: Brief Introduction

• Verilog-a is a subset of Verilog-AMS (analog and mixer-signal) standard
  – High-level hardware description language for analog behavior; specifically it is designed to describe models for SPICE-class simulators;
  – It enables designers to characterize electrical systems, through the potential and flow (voltage and current) description;
  – Components are constructed using collections of nodes (voltages) and branches (currents)
Simulator Independence of Verilog-a

- Verilog-a models are theoretically independent from specific simulators:
  - Model developers only need to concentrate on constructing the constitutive relationships between the input and output using the verilog-a syntax, while verilog-a compilers process the interaction between the models and simulators
  - Similar to the “Write-once, run everywhere” cross-platform Java language
Typical Development Procedure using Verilog-a Approach

1. Model Parameter Registration
   - Inform the simulator what parameters will be passed into the model (verilog-a specific).

2. Math Model Implementation
   - Add the mathematic expression of the model to the code (verilog-a specific).
   - Assemble the MNA (modified nodal analysis) matrix by stamping all the elements used in the model netlist to the matrix (verilog-a specific).

3. Nodal Y Matrix Filling
   - Compile the model code and test the model object to make sure it matches with expectation (simulator specific).
   - Prepare the PDK delivery (simulator specific).

4. Model Compilation / Debugging Simulator 1

5. Model Compilation / Debugging Simulator n

6. PDK packaging / Documentation Simulator 1

7. PDK packaging / Documentation Simulator n
Limitations in Verilog-a Models

Translator-dependent:

- It is up to the vendors’ translator to get the verilog-a code correctly translated into built-in APIs style for simulation (errors can happen here).
- The simulated results may be different slightly due to: How the model is linearized; initialized; numeric truncation, etc.
- It is up to the translator to determine if all or part of the standard is supported.
- Some system calls may not be supported equally between translators.
Simulation speed:

- Due to the generalization of the Verilog-a language, the translated code may not be as succinct as using the built-in APIs directly
- This can lead to some loss in the simulation speed
MDLX’s Verilog-a Diode Model Library

• NLD (Non-linear Diodes) Library
  – Over 70 models
  – Available in multiple simulators
    • Keysight ADS
    • Keysight Genesys
    • NI/AWR Design Environment (AWRDE)
    • ANSYS HFSS, Cadence (in development)
  – Uses same Verilog-a cores for each simulator
    • 10 in house developed diode model cores
  – Parasitics not included in cores
    • Added as part of model porting to each simulator

Toshiba 1SV229 Varactor Diode Model Symbols from (left to right) AWR, Genesys and ADS
Modelithics Qorvo-GaN Non-linear Model Library

- ~60 models: both die-level and package devices
- Fully sponsored - free to Qorvo customers
- Available in multiple simulators
  - Keysight ADS
  - NI/AWR Design Environment (AWRDE)
- Uses same Verilog-a cores for each simulator
- Modelithics version of a modified Angelov model
  - Parasitics separated from nonlinear core
  - Enable intrinsic voltage/current sensing
  - Enhanced scalability and thermal modeling features.

TGF2819-FL 120W GaN Device Model
Case Study: VA-based GaN Transistor Model in Three CAEs

• A GaN model is illustrated here as a successful verilog-a model practice
• It consists of a modified Angelov nonlinear core implemented in verilog-a, surrounded by parasitic and package elements
• DCIV/SP/power sweep/loadpull contours will be demonstrated in following slides
  – As will be demonstrated, good agreements have been achieved through the usage of Verilog-a core model for both linear and nonlinear performances.
A simple DCIV Simulation Setup in three Simulators

- Cadence
- ADS
- AWR
IV Simulation Results

**ADS**

**Cadence**

**AWR**
S-parameter Results: Cadence vs. ADS

5GHz, 32Vds, -2.77Vgs, Ids 100mA,
S-parameter Results: AWR vs. ADS

5GHz, 32Vds, -2.77Vgs, Ids 100mA,
Power Sweep: AWR vs. ADS

5GHz, 32Vds, -2.77Vgs, Ids 100mA,
Power Sweep: Cadence vs. ADS

5GHz, 32Vds, -2.77Vgs, Ids 100mA,
Power Contours: 9GHz, 32Vds, -2.76Vgs, IdsQ of 100mA
PAE Contours: 9GHz, 32Vds, -2.76Vgs, IdsQ of 100mA
Conclusion

- The myriads of available CAE solutions in the market present a challenge for model providers to port and maintain model updates.
- Traditionally Model providers have to code separately for each set of built-in APIs.
- Verilog—a modeling approach offers a “write-once, run everywhere” possibility that steers developers’ focus on model quality instead of implementation.
- Not so fast! There remain differences among the vendors’ translators so that the code they develop can work properly and consistently across simulators and some custom treatment for each simulator is still required.