Extensions to Verilog-A to Support Compact Device Modeling

Laurent Lemaitre, Geoffrey Coram†, Colin McAndrew, Ken Kundert‡

Motorola, †Analog Devices, ‡Cadence Design Systems
Outline

• What is Compact Modeling?
  – The 95% attraction of Verilog-A
• Simulator/Compact Model Interaction
  – Full Duplex communication link
• The remaining 5%
  – What’s missing in Verilog-A for CM
  – Proposed language enhancements
• Wrap-up
Compact Models

A “compact model” is a model intended for use in an analog circuit simulator – “analytic” or “SPICE” model

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Modeling Basis</th>
<th>Accuracy</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCAD (physical)</td>
<td>Very detailed Physics</td>
<td>Microscopic</td>
<td>Minutes</td>
</tr>
<tr>
<td>Compact (analytic)</td>
<td>Physics Based</td>
<td>Mostly Reasonable</td>
<td>Milliseconds</td>
</tr>
<tr>
<td>Behavioral</td>
<td>Electrical Performance</td>
<td>Macroscopic</td>
<td>Immaterial</td>
</tr>
</tbody>
</table>
Compact Model Compilers

• Coding compact models in C is
  – time consuming
  – error-prone
  – a non-value added, 2\textsuperscript{nd} rate job
  – an impediment to adopting improved software practices
  – a barrier to re-use
  – a barrier to co-development
  – simulator specific, non-standard, ...
The Problem
The Problem
The Solution

Verilog-A+CGenerators

VBIC
ACM
USIM
HiCUM
BSIM
Mextram
SP
MM11
HiSIM

Eldo
Spectre
ADS
Smash
HSIM
APLAC
Nanosim
HSIM
HSPICE
Golden Gate
AMS
Why Verilog-A?

• 95% perfect for defining compact models
  – even though meant for behavioral models

• Advantages
  – widely used, standard language
  – do not have to code derivatives
  – “natural” and flexible for writing models
  – concise, easily readable code
  – <5% of the effort of C coding
Compact Model Concepts

- Model Parameters
  - $T_{OX}$, $N_{SUB}$, $D_W$, $V_{EF}$, ...

- Instance Parameters
  - $L$, $W$, Area, Perimeter, $N_{base}$, $m$, ...

- Model initialization
  - code independent of geometry and bias

- Instance initialization
  - code dependent on geometry, but not bias

- Evaluation

- Post-processing
Parameter Aliases

• It is sometimes convenient to be able to have aliases defined for parameters
• Primarily for user frustration avoidance
  – confusing “0” with “O,” “1” with “l”
  – common aliases: “dtemp,” “dta,” “trise”
• Propose “alias” for parameters

```verilog
parameter real vto=0.4;
parameter alias vt0=vto;
```
Parameter Aliases

- It is sometimes convenient to be able to have aliases defined for parameters.
- Primarily for user frustration avoidance:
  - confusing “0” with “O,” “1” with “l”
  - common aliases: “dtemp,” “dta,” “trise”
- Propose “alias” for parameters

```
parameter real vto=0.4;
parameter alias vt0=vto;
```
Optional Terminals

• Sometimes it is useful to be able to have optional terminals
  – to select 3 or 4 terminal BJT models
  – for local $\Delta T$ self-heating nodes
• “Ground,” “float,” or “ignore” are possible behaviors
• Verilog supports this
• Discussion of proposal on-going

```
instance (n1,n2,,n4,,n5); // ??
```
Description Field: --- Construct

• For many parameters, terminals, and variables, being able to embed a text description in Verilog-A is desirable
• Trailing string after triple-dash construct

parameter real w=1.0u --- "width";
Unit Field

- For many parameters and variables, being able to specify the unit is desirable (should be SI where possible)
- String following default value for parameters
- String following name for variables

```verbatim
parameter real w=1.0u "m" --- "width";
real pdiss "W" --- "power dissipation";
```
Unit and Description

• Make code more understandable
• Allow automated generation of model documentation
  – tabulated parameter lists, with defaults, units, and descriptions
• Allow automated generation of display information (e.g. .OP) with units and descriptions
String Variables

- Verilog-A includes “real” and “integer” data types, propose adding “string”
- Useful for polarity specification
- Implicitly requires overloading of operators to handle string data types

```verbatim
parameter string type=“n” --- “polarity”;
if (type = “n”)
    Vgs = V(g,s);
else
    Vgs = V(s,g);
```
Model and Instance Parameters

• Verilog-A parameters are at present implicitly of type “instance”
• Compact models need to differentiate between “instance” and “model” parameters
  – for efficiency (“model” and “instance” initialization)
  – for practical use, “model” parameters in model files provided by modeling groups, “instance” parameters in netlists generated from design kits
• Propose addition of keywords to “parameter” declaration

```
instance parameter real w =1u "m" --- "width";
model parameter real tcl=1m "/C" --- "linear TC";
```
$\text{param\_given()}$

- In compact models, it is useful to be able to determine if a parameter has been specified
  - e.g. where multiple ways of specifying some related parameters is possible ($\gamma$, $N_{\text{sub}}$)
  - using defaults like -1.0 or 9.99e99 is inelegant and can be ambiguous

- Propose addition of $\text{param\_given()}$ function

```verbatim
if ($\text{param\_given(rsub)} && \text{param\_given(xj)})
    rs=(rsub/xj)*(l/w)/8
else
    rs=rshsub*(l/w)/8
```
Although conceptually it appears desirable to separate simulator algorithms from models, this is not always possible in practice:
- gmin for improved convergence
- homotopy may require specific action in models

Propose addition of the function
```
$simparam("name",[1|0,default])
```

```
gmin=$simparam("gmin",1); // die if !defined
gmin=$simparam("gmin",0,1.0e-12);
```
“Display” Parameters

• It is desirable to be able to define OP (operating point display) parameters
  • Proposal is to define this as “top level” variables
    – those declared outside “analog” block
    – simple and effective approach
  • Should define units and description for each “OP” variable

real pdiss “W” --- “power dissipation”;
analog begin : module
Derivatives

• BCRs implicitly have derivatives generated if necessary
• Would like explicit derivatives for OP information display
• Propose $ddx()$ operator to allow this

real gm "S" --- "transconductance";
analog begin : module
    gm = $ddx(Ids,V(g));
• Although it appears at first that it could be difficult to handle this automatically, it is in most cases quite simple
  – divide controlling $I(n1,n2)$ by $m$
  – multiply output $I(n1,n2)$ by $m$
  – multiply noise currents ($A^2/Hz$) by $m$
  – divide noise voltages ($V^2/Hz$) by $m$
• History: it’s easy to mess up $m$ code
• Mismatch is different, requires scaling by $1/sqrt(m)$ (instance parameters)
• How to handle this is being discussed
Conclusions

- There is a groundswell of support for the high-level-language+compilers approach for compact model definition and implementation
- Verilog-A is emerging as the language of choice for this task
- Some fine tuning (5%) of Verilog-A is required to optimize it for compact modeling
- Proposals for Verilog-A language extensions to better support compact modeling are being finalized at present
Conclusions

• Have looked at VBIC, Mextram, BSIM3v3, BSIM4, MOSCAP, R3, SP models
  – DC, AC, noise
  – scalability
  – statistical modeling (including mismatch)
  – multiplicity
  – …

• Proposals take us from 95% to 99.9% coverage for compact modeling needs
Acknowledgements

• Ilya Yusim (Cadence)
• Jeroen Paasschens (Philips)
• Sri Chandrasekaran (Motorola)
• David Zweidinger (TI)
• Marek Mierzwinski, Boris Troyanovsky, Patrick O’Halloran (Tiburon)
• Peter Liebmann, Han Koh (Xpedion)
• John Moore, Rick Poore (Agilent)
• Weidong Liu, Carl Gu (Synopsys)
• Mustafa Sungur (Nassda)
• Wolfgang Roethig (NEC)
• Gary Fedder (CMU)
• Kan Bakalar (Mentor Graphics)