Practical Considerations for Real Valued Modeling of High Performance Analog Systems

Dushyant Juneja, Analog Devices, Inc.
Siddharth Prabhu, Analog Devices, Inc.
Syam Veluri, Analog Devices, Inc.
Agenda

• Introduction
  – Behavioral Modeling
  – RVMs – Introduction and flavor comparisons

• Modeling aspects in dealing with RVMs

• Integration aspects
  – Netlisting & Validation

• What’s missing, exactly?
  – With the standard
  – With the tools

• Conclusion
Real Valued Modeling

- **Behavioral modeling technique** to simulate *analog functionality* within a *digital simulation*
  - Required for robust Universal Verification Methodology (UVM), Metric Driven Verification (MDV)
  - Event-driven evaluation → faster top-level AMS functional verification environment
- SystemVerilog offers new constructs - user-defined nettypes, (UDTs) & user-defined resolution functions (UDRs).
- RVM augments other detailed models, may not completely replace them!
Real Valued Modeling

- Speed over accuracy and modeling details captured
- Abstraction level is much higher closer to sub-system level blocks

// Amplifier gain
real gain = 1000 ;

// Output computation
always @( inp, inm, vdd, vss ) begin
    tmp_in = inp - inm ;
    tmp_out = gain * tmp_in ;

    // Hard limiting the output
    if (tmp_out > vdd) : tmp_out = vdd ;
    else if (tmp_out < vss) : tmp_out = vss ;

    #1 out = tmp_out ;
end
## Comparison with Behavioral Modeling Standards

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Verilog-lookalike for SPICE</td>
<td>Superset of Verilog-A and Verilog-D</td>
<td>Verilog-AMS RVM subset</td>
<td>Most recent approach for RVM</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Continuous</td>
<td>Flexible</td>
<td>Discrete time</td>
<td>Discrete time</td>
</tr>
<tr>
<td>Digital Integration</td>
<td>Via Cosims</td>
<td>True AMS, limited UVM/SV</td>
<td>True AMS, limited UVM/SV</td>
<td>Excellent integration</td>
</tr>
<tr>
<td>Modeling Features</td>
<td>True analog modeling</td>
<td>True analog + mixed signal interaction</td>
<td>Abstract signal chain only</td>
<td>Abstract signal chain, but more permissive</td>
</tr>
<tr>
<td>Speed*</td>
<td>Slow</td>
<td>Better</td>
<td>Close to digital</td>
<td>Fast, close to digital</td>
</tr>
</tbody>
</table>

*Speed is a subjective matter – the comparison shown is approximate and highly dependent on level of abstraction.*
# RVM Feature Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>SV 2009 Real</th>
<th>Verilog-AMS wreal</th>
<th>SV 2012 Real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real values</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Real Nets, Unidirectional ports</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Bidirectional real ports</td>
<td>Not supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>User defined nettypes and Resolution</td>
<td>Not supported</td>
<td>Restricted</td>
<td>Supported</td>
</tr>
<tr>
<td>Real Randomization &amp; Constraints</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Limited support</td>
</tr>
<tr>
<td>Coercion</td>
<td>Not supported</td>
<td>Supported, but patchy</td>
<td>Using generic ‘interconnect’ type</td>
</tr>
</tbody>
</table>
SV 2012 RVM – UDTs/UDRs

- RVMs – promising approach for mixed signal verification
- SV 2012 the ‘latest & greatest’ of the suite
  - Extends RVM methods via ‘nettypes’ (my_type in code)
  - Custom resolution
  - Single kernel simulation
- Several new concerns!!!
Analog Event Explosions

Simple non-inverting amplifier configuration.

Expected:

\[ V_{out} = V_{in} \]

Actual:
Unstable oscillations!!!
Analog Event Explosions

Possible Issues:

- **Zero Delay Feedback** on finite delay amp → corrupt output!

- Unconditionally unstable configuration

\[
H_{\text{expected \ (opamp)}} = G, \ G \gg 1
\]

\[
H_{\text{actual \ (opamp)}} = \frac{G}{1 + G \cdot z^{-1}}
\]
Analog Event Explosions

Solution:
Can be compensated by Amplifier pole (band limited amplifier), if:

$$\delta < \frac{\tau}{G}$$

$$\delta = \text{simulator time step}, \ \tau = \text{Amplifier time constant}$$

Application – Simulation timestep needs to be carefully chosen!
Analog Event Explosions

Other Concerns:
1. Complex Configurations
   1. Multiple drivers on node such as ‘inn’ → Sorted out by custom resolution functions
   2. Impedance based feedbacks → Need for custom nettypes
2. Artificial error → Needs to be tailored using timestep control
3. Too many features, tight timestep → Slow simulations!
Analog Contention & Unknown States

• Multiple driver resolution and Unknown states – vitally important for AMS modeling
  – Improperly handled digital unknown states can shadow real bugs!
  – Mixed signal contention on muxed IOs
  – Contention due to testbench drivers

• Example – Feedback loop
  – X-Lock corrupts the loop!
Analog Contention & Unknown States

• Sorted out via nettypes:

```c
typedef struct{
  real V;    // Voltage value
  real I;    // Current flow value
  logic isX; // flag for detecting unknown voltages
} my_net;

// Resolution function to highlight unknowns
function my_net my_res_fnc(input VI_with_X node[]);
  logic result_is_X;
  int i;
  ...
  for ( i=0 ; i < node.size() ; i++ ) begin
    if ( node[i].isX === 1'b1 )
      result_is_X = 1'b1;
  ...
  // Other stuff for dealing with V and I
```
High Impedance and Ground References

• Extremely common in AMS IPs.
• Need a way to determine ‘strength’ of a ‘node’/driver, and resolve accordingly
  – Better still, resolve based on ‘drive impedance’
• Nettypes come to the rescue!
  – Can be extended to handle impedances
  – Scalable approach
High Impedance and Ground References

```c
1 // New user defined type
typedef struct {
2   real voltage; // The actual signal
3   real Zr; // Real component of impedance/strength
} my_udt;

7 function my_udt my_res ( input drivers [] ) ;
8   ...
9   for ( i=0 ; i < drivers.size() ; i ++ )
10      if ( Zr = 0 ) // Ground/Strong Reference case
11         final_v = drivers[i].voltage;
12      else if ( Zr > Zr_max ) : next ; // Driver can be ignored!
13     ...
```
Integration Concerns

• Netlisting - Stitching/Connecting RVMs as per schematics (Structural concern)
  – Deriving top level SV netlist from analog repo
  – Mixed signal buses
  – Inherited connections

• Integration with top level digital sims
  – Coercion
  – Nettype compatibility

• Analog stimuli generation
Mixed Signal buses – various concerns

- No SPICE-like bitwise connections
  - Difficult to figure out connectivity at a glance

- AMS Buses typically have unpacked assignments
  - Disturbs majority SV netlisting streaming tools (output packed arrays by default)
  - Kills scope for bus concatenation
  - Bus reversal, etc. not possible either

- Need to be worked around by flattening and removing bus references
module mod_name(
    inout my_net ibus [3:0] );
...
endmodule

mod_name inst_name ( .ibus(obus) );

mod_name inst_name ( .ibus(`{p1,p2,p3,p4}) ) ;

mod_name inst_name ( .ibus[0](p1), ibus[1](p2), . . . );
RVM Netlisting – Inherited connections

Inherited connections – kind of global pins, inherited from higher levels in schematics

• Avoids cluttering the design symbols with repeated pins
  – Analog designers use it to make pretty schematics

• Actually just another invisible connection

• Majority of vendor tools fail to netlist this properly, need to be worked around with custom scripts
RVM Netlisting – Key requirement?

Coercion, or Type propagation

nodeA: no explicit definition

nodeA defined as ADInet1 (custom nettype)

module ana_top (nodeA, nodeB, ...);

module ana_mid_1 (nodeA_m, nodeC_m, ...);

module ana_mid_2 (nodeB_m, nodeD_m, ...);

module ana_child_1 (nodeA_c, nodeB_c, ...);
inout ADInet1 nodeA_c;
inout ADInet2 nodeB_c;
...

nodeB

nodeC

nodeC multiple drivers!

Conflict Resolution / Multiple Drivers

nodeC

module ana_child_2 (nodeB_c, nodeC_c, ...);
output ADInet1 nodeB_c;

module ana_child_3 (nodeC_c, nodeE_c, ...);
output ADInet2 nodeC_c;

Interface / Connect Modules: ADInet1 to ADInet2
RVM Netlist - Coercion and Conflict resolution

• NOT an issue for just digital simulations or single nettype simulations
  – `default_nettype` fails to handle multiple nettype scenario
• Need a forced resolution on nets of different discipline
  – Multi-driver, mixed discipline nettypes
  – Real – logic and arrayed ports
  – doing conflict resolution too
• Generic ‘interconnect’ to the rescue
  – Manual edits may be needed for tool generated netlists
  – What coerced to what? Still a mystery… !
RVM Simulation and Model Validation – connect rules

**SPICE Stimuli**
- Transistor Schematic
- Abstracted Model with UDT/UDR (.sv)

**SV Stimuli**
- Verification Engineer Testbench
  - Interested in functionality
  - Verilog on top flow

**Design Engineer Testbench**
- Interested in performance
  - SPICE on top flow

**Transistor Schematic**
- Abstracted Model with UDT/UDR (.sv)
RVM Integration – Other Concerns

- Not friendly with Low Power Intents like CPF/UPF…
  - Coercion breaks!
  - Power aware connect modules?
- SV Interfaces not too friendly with RVMs/nettypes
- Schematic capture tools – new checks needed (like SV keyword checks)
- Real randomized/Constraint Randomized stimuli!

...
Next Steps & Conclusion

• Discussed practical issues and work-around in Real Valued Modeling of High Performance Analog

• Limitations faced from IEEE 1800-2012 (SV-LRM)
  – Critical improvements required

• In spite of all these issues, we still believe SystemVerilog 2012 along with MDV/UVM flow will greatly enhance productivity.

• Will standards committee and EDA tools step up for resolving the challenges?!
Next Steps & Conclusion

- Develop a parameterized library of Real Valued Models
- Automated Insertion of Connect Modules
  - Looking forward to SV-AMS
  - Validation flows
- Top-Level SystemVerilog digital netlisting
- EDA tools, Standards committees and Verification teams have scope for mutual collaboration
Thank You