Specification Driven Analog and Mixed-Signal Verification

Henry Chang, Ken Kundert
Designer’s Guide Consulting, Inc.
Failure is Not an Option

• With rise of complexity has come a dramatic increase in functional errors
  – Increasing dependency between analog and digital
  – Co-verification of interface is half-hearted
  – Bugs found on both sides of interface …

![Diagram showing interaction between Vendor A and Vendor B]
What Causes the Failure

- Complexity at many levels
- Analog performance analysis tools differ from ones needed for analog verification (AV)
  - One focuses on the signal path
  - Other focuses on what surrounds the signal path
Analog Verification

Finding all functional issues related to analog

- Bugs in RTL that talks to analog, e.g. calibration, control loops, power-up sequences, etc.
- Bugs in analog-digital interface
- Incorrect default settings for analog
- Bugs in embedded software that controls analog
- Functional bugs in analog- bad digital decoding, swapped or inverted signals, etc.

3/2/2022
Digital Design & Verification

• Specification is hard
  – Leads to separation between design and verification

• Verification about complexity and state
  – Leads to directed random, coverage, assertions in testbench

• Digital does not have an implementation problem (synthesis exists)
  – It has a specification interpretation and complex verification problem
Analog Design & Verification

- Generally, analog blocks have no state
- Analog does not have a spec interpretation problem
  - Function straightforward, constraints well defined
- The problem for analog blocks
  - Schematics are golden
  - Schematic is error prone to create, cannot be used in verification (too slow, too late)
- Analog has a design implementation and basic verification problem
  - Schematics often have *functional* bugs (translation error, not misunderstanding)
  - Need to create *validated* models for verification

Not issues in digital
Analog Verification Blind Spots

• For analog designers …
  – Do not understand need for models
  – And the need for those models to be fully verified
  – Visual inspection is not verification
  – Focus is on performance, not function

• For digital verification engineers …
  – Do not understand that AV have different challenges
    • Many DV techniques do not apply (at least initially)
  – And that models are created by hand and not part of the traditional analog design process
Chip Level Verification

• Chip-level verification flow can be any digital flow
  – Just need analog behavioral models
  – Without models cannot do true chip-level verification
  – Can be discrete-time only or mixed discrete/continuous time
What is Needed?

- **Validated models are required early!**
- Only 4 months for design; parallel tasks required
  - Cannot wait for schematics to start verification
- Digitally corrected analog requires close coupling when writing RTL
- Everyone benefits from early availability of models
  - System designers, product engineers, test engineers
  - Digital designers, digital verification engineers
  - Analog verification engineers
- AV should begin the day chip level specs are ready
How to Efficiently Write Models and Validate Them?

• Recognize AV is not a hard problem
  – High-level modeling languages and mixed-signal simulators exist
  – Functional analog models are straightforward
  – Self checking functional tests are straightforward
  – Automating self checking tests is straightforward
module flash_adc (out, in, clk, bias, pwrdn, vdd);  
  input in, clk, bias, pwrdn, vdd;  
  output [15:0] out;  
  electrical in, bias, vdd;  
  integer i, level;  
  reg pwrFault, biasFault;  
  reg [15:0] d;  
always @(posedge clk) begin  
  pwrFault = (V(vdd) > 1.9) || (V(vdd) < 1.7);  
  biasFault = (I(vdd,bias) > 16u) || (I(vdd,bias) < 14u);  
  level = 16*V(in);  // convert input to an integer  
  for (i=0; i<16; i=i+1)  
    d[i] = (i < level);  
end  
assign out = (pwrdn || pwrFault || biasFault) ? 16'bx : d;  
analog begin  
  V(vdd,bias) <+ pwrdn ? 0 : 0.5 + 20k*I(vdd,bias);  
  I(vdd) <+ pwrdn ? 1u : 500u;  
end  
endmodule
module testbench ();
reg [7:0] in;
electrical out, vdd;
integer i, failed;
real Vdd;
dac8 DUT(.out(out), .in(in), .vdd(vdd));
analog V(vdd) <+ transition(Vdd, 0, 50n);
initial begin
  Vdd = 2.5;
  for( i='h0; i <= ‘hFF; i = i+1) begin
    in = i;
    #(100n);
    failed = abs(256*V(out) – i) > 0.5;
    $display("%0s: out[%0d] = %0.3f, expected %0.3f", failed ? "FAIL" : "Pass", i, V(out), i/256.0);
  end
$finish;
endmodule
Issue With Approach

• Analog verification requires 20% of the design effort
  – 30% to 40% for a new team
  – Many models need to be written and verified
    • Majority of time spent on block level models
    • Need models before chip-level verification can begin
• Resources difficult to find
  – Analog designers cannot solve the problem
    • Not their job, overworked, ill equipped
  – Digital designers or verification engineers
    • Most not familiar with reading schematics
    • May not be familiar with analog
How to Address Efficiency?

• Improve process of developing and verifying models
• Take advantage of:
  – Ease in writing analog block models
  – Ease in specifying typical analog blocks
• Let analog designer focus on performance as always
• Focus on function first in verification
• Add automation
Start with Specifications

• Analog functional specs are generally pretty simple
  – Complexity naturally flees to digital
  – Easily created by analog designers

• Contains all information necessary to
  – Build the model
  – Verify the circuit
  – Verify the model

• Write formalized specification for analog blocks, one that is unambiguous and parsable
  – Design “language” of spec to be human readable
  – The result is an executable specification
Executable Specs

- Formalized Specification
- Testbench
- Verification Script

Using a generator...

Model of the analog block

Validates model and schematic are equivalent

Used to automate regression testing
## Example: PGA

<table>
<thead>
<tr>
<th>Name</th>
<th>Dir</th>
<th>Type</th>
<th>Description</th>
<th>Range</th>
<th>Behavior</th>
<th>Nominal</th>
<th>Instruments</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>output</td>
<td>voltage</td>
<td>PGA output</td>
<td></td>
<td>$V = en \times Gain \times V(in) \times \text{Fault}$</td>
<td></td>
<td>with tol=10m</td>
</tr>
<tr>
<td>in</td>
<td>input</td>
<td>voltage</td>
<td>PGA input</td>
<td></td>
<td>1.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gain</td>
<td>input</td>
<td>signed</td>
<td>Gain (in half dB steps)</td>
<td>[5:0]</td>
<td>$\text{Gain}=\text{dB}(\text{gain}/2)$</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>en</td>
<td>input</td>
<td>digital</td>
<td>Enable</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bias</td>
<td>input</td>
<td>ibias</td>
<td>Bias</td>
<td>5uA to 15uA</td>
<td>10u</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vdd</td>
<td>input</td>
<td>supply</td>
<td>Vdd</td>
<td>2.25V to 2.75V</td>
<td>$I = 100uA \times en$</td>
<td>2.5</td>
<td>flow: with tol=1m</td>
</tr>
<tr>
<td>gnd</td>
<td>input</td>
<td>ground</td>
<td>Ground</td>
<td>-10mV to 10mV</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: PGA

- Spec is sufficient to …
  - Generate pin accurate functional model
    - Model refuses to operate unless biased properly
  - Generate functional testbench
    - Observes output voltage, supply current
    - Exercises gain, enable
    - Suitable for circuit versus model testing
    - Full functional coverage
Why Executable Specs

- Human readable specs facilitate communication
  - Easy to read, understand, and update (reduces error)
  - Eliminates ambiguities (ex: direction of current flow)
  - Any design team can read (unlike schematics)
  - Incredibly concise compared to models – easy to enter
- Specs will always exist and be up to date
- Functionally verifies specification == schematics

![Diagram showing relationships between Specification, Model, Testbench, and Schematics]
Executable Spec Benefits

- Can use to generate functional models and testbenches
  - Creates models early in design process
  - Allows digital designers to create analog-dependent RTL
  - Gives time for analog and digital verification teams to do a good job
- Models are validated against schematics once schematics are ready
- Verified models available to …
  - Digital design & verification engineers
  - Chip level verification engineers
  - System designers, test engineers, product engineers, end customer
  - Assist in analog design
Models in Minutes

• We developed tool that translates specifications into model and testbench, PGA Example:
  – Model: 196 lines of code
  – Testbench: 663 lines of code

• Remove tedium
• Better written
• Avoid tool issues
• More uniform
• One person can write many more blocks
• Additional tables cover other key capabilities
Conclusion

• With spec-driven analog verification
  – Early, validated, and efficient models of analog blocks
  – Fully verify transistor level analog design
  – Enable co-design and co-verification of analog and digital
  – Enable chip level verification
• Analog designer or analog verification engineers can create the models quickly
• Designs can be verified with fewer AV engineers
• Allows AV engineers to work together better to verify design
• Practical, efficient, systematic, scalable, repeatable
Acknowledgements

• Flow based on many verification efforts on many types of chips including power management, audio, data communications, RF, etc.
• We would like to thank our colleagues at the many different companies with whom we have worked.