Digital Friendly Behavioral Modeling Methodology, intended for Mixed Signal Functional Verification

Example Amplifier Model:

```
1 // Amplifier gain
2 real gain = 1000; //
3
4 // Output computation
5 always @(posedge input) begin
6   out = input * gain;
7 end
```

### Features

- **Modeling**
  - Slow Better Close to Speed*
- **Continuous Evaluation**
- **Verilog-Approach** Verilog-A Verilog-AMS

### Comparison with Other Methods:

- **Pros:**
  - Tight Integration with Digital/SystemVerilog
  - Improved feature set with IEEE 1800:2012 LRM

- **Cons:**
  - Increased manual effort
  - Mixed Signal Buses
  - Analog Stimuli concerns
  - Randomization, etc. do not work out-of-the-box

### Analog Stimulus Development

Analog Stimuli are quintessential for mixed signal testbenches. The stimuli need to be able to drive complex protocols, and have the ability to drive mixed signal randomized vectors. However, there is limited support for real randomization amidst vendor tools, and sometimes even requires special licenses. This was worked around by suitably constrained random integer division:

```
1 class xtda;
2   int a; // Universally supported construct
3   constraint ctt1 { a < 1230; }
4   constraint ctt2 { a < 10; }
5   endclass
6   ...
7 if (ca.randomize() == 1) begin
8   x = ca.a/123.0;
9   $display("REAL Value = %g \n", x);
10   end
```

### Analog Modeling – Contention & Unknown states

High Impedance and Unknown states – vitally important for AMS modeling. Example scenarios:

1. Mixed signal feedback loops. Improperly handled digital unknown states can shadow real bugs (Figure 4).
2. Mixed signal contention on chip pads/block pins

### Structural Concerns

Netlisting and Integration are two major concerns here. The following challenges were faced in this course:

- Need for SV 2012 netlist.
- Type coercion.
- Mixed Signal Buses.

The netlist was worked around using a custom script.

**Type Coercion** refers to type-propagation for leaf level nets all the way to the top level. This is necessary because the top level connections remain essentially “typeless” otherwise, causing compile issues. A corollary is conflict resolution, where a conflict between multiple nettypes needs to be sorted out. A pictorial representation is as in Figure 5. The coercion was carried out using SV 2012’s generic “Interconnect”, while conflict resolution was sorted out manually for the current flow.

### Results

The zero delay feedback usage workarounds suggested were benchmarked for simulation accuracy and speed for the case of non-inverting amplifier circuit. As observed, the ideal amplifier gadget actually slowed down the simulation, even for the case where it did come out of delta cycles (that is, where explicit delay is provided in the model). On the other hand, after modifications as discussed, the simulations were <10% faster.

### Analog Modeling – Zero Delay Feedbacks

Simple non-inverting amplifier configuration. Expected output voltage/transfer function:

```
Vout = Vin \cdot \frac{G}{1 + G \cdot z^{-1}}
```

However, this gets difficult to model:

1. Node ‘inn’ could have multiple drivers for complex configurations → Sort out by custom nettypes (SV 2012)
2. Zero Delay Feedback → Simulation gets hung
3. Small, Artifical Delay → Amplifier gets unstable (Figure 2)

Achieved via nettypes:

```
1 typedef struct {...
2 endclass
```

### Analog Modeling – High Impedance and Ground Refs

High Impedance and Ground references – Extremely common in AMS IPs. Traditional methods do not allow ‘strong’ or ‘weak’ real nets. Even if this was allowed, the limited array of options may not be sufficient. Nettypes help sort out this situation by allowing ‘drive impedence’ on nodes. Appropriate updates need to be done in resolution functions as well as mcmic this function as required.

```