The IBIS-X macro language .....

some applications

Al Davis
Idaho State University

Outline

- The component as a macro
- Inherit and Extend
- Test Data, Test Load
- Differential devices

The component as a macro

- Fixed topology
- "Foreach" arrays

The component as a macro

[Pin]

```
foreach line in [Pin]
  capacitor C$0$ ($0$ 0) C =
                        $C_pin$ || [Package]C_pkg
  if ($2$ != "NC")
    resistor R$0$ ($0$ $0$.int) R =
                        $R_pin$ || [Package]R_pkg
    inductor L$0$ ($0$.int $1$) L =
                        $L_pin$ || [Package]L_pkg
    if ($2$ != "GND" && $2$ != "POWER")
      subckt X$0$ ($1$ 0 control$0$ enable$0$
       npdr$0$ npur$0$ ngcr$0$ $npcr$0$)
            [Model]$2$
    end if
  end if
end foreach
```

The component as a macro

[Pin Mapping]

```
foreach line in [Pin_Mapping]
  define npdr_$0$ = $1$
  define npur_$0$ = $2$
  define ngcr_$0$ = $3$ || $1$
  define npcr_$0$ = $4$ || $2$
end foreach

 ]
```

The component as a macro

[Diff Pin]

```
foreach line in [Diff_Pin]
  define diff_$0$ = $1$
  define diff_$1$ = $0$
  define vdiff_$0$ =
          (($2$ != "NA") || .2) / 2
  define vdiff_$1$ = -vdiff_$0$
  define delay_$0$ =
          (($3$ != "NA") || 0) / 2
  define delay_$1$ = -delay_$0$
end foreach

 ]
```

The component as a macro

[Series Pin Mapping]

```
foreach line in [Series_Pin_Mapping]
  subckt X$0$$1$ (signal$0$ signal$1$
      control$0$ enable$0$ npdr_$0$
      npur_$0$ ngcr_$0 $npcr_$0$)
      [Model]$2$
end foreach


]
```

The component as a macro

Wrapping it up

```
[Define Component] IBIS.3.2
foreach line in [Pin_Mapping]
...
foreach line in [Diff_Pin]
...
foreach line in [Pin]
...
[End Define Component]
```

The component as a macro

I didn't show you

- Package model
- Some details
- Coupling

So what???

- Complete flexibility in pin-to-pad
- User extendable

Inherit and extend

- Start with a standard macro
- Inherit it
- Then add your part

- Example: Differential buffers

Inherit and extend

Differential buffers

```
[Define Component] extended_diff
inherit [Component]IBIS.3.2
foreach line in [Diff_Model]
 subckt X$0$$1$ ($0$ $1$ control$0$
    enable$0$ npdr$0$ npur$0$
    ngcr$0$ npcr$0$) [Model]$2$
end foreach
[End Define Component]
```

Inherit and extend

Differential buffers

```
[Component] RGF5463
Component_type extended_diff
|| other stuff like it always was!
[Diff Model]
2 3 diff_in
4 5 diff_out
[End Define Component]
```

Inherit and extend

Differential buffers

```
[Define Model] Input_diff (ports)
inherit [Model]Input
resistor Rpcx (dpin power_clamp_ref)
    I = [POWER_Clamp](-V)
resistor Rgc (dpin gnd_clamp_ref)
    I = [GND_Clamp](V)
[End Define Model]
```

Inherit and extend

Differential buffers

```
[Model] diff_in
Model_type Input_diff
|| just like the regular Input!
.....
[End Define Model]
```

Test data, Test load

- There is nothing special about "[Model]"!
- Take advantage of that to make something new.

Test data, Test load

```
[Define Test Load] single_ended (control en
          near far gnd)
node pulldown_ref pullup_ref gnd_clamp_ref
power_clamp_ref
node 2, 3, 4, 5, 6, 7
capacitor C1_near (near gnd) C = C1_near
resistor  Rs_near (near 2) R = Rs_near
inductor  Ls_near (2 3) L = Ls_near
capacitor C2_near (3 gnd) C = C2_near
| ... part left out
subckt X_far (far gnd control en pulldown_ref
    pullup_ref gnd_clamp_ref power_clamp_ref)
       Receiver_model
[End Define Test Load]
```

Test data, Test load

```
[Test Load] load-1     ]
Test_Load_type single_ended
C1_near      = 1p
Rs_near      = 10
Ls_near      = 1n
C2_near      = 1p
Rp1_near     = 100
Rp2_near     = 100
Td           = 1ns
Zo           = 50
```

Test data, Test load

```
[Define Test Data] (trigger dut_near dut_far
                    golden)
node enable_pin pulldown_ref pullup_ref
gnd_clamp_ref power_clamp_ref
subckt x_dut (dut_near 0 trigger enable_pin
    pulldown_ref pullup_ref gnd_clamp_ref
    power_clamp_ref) Driver_model
subckt x_load (dut_near dut_far 0) Test_load
trigger  TR   (Logic(control) == 1)
trigger  TF   (Logic(control) == 0)
vsource (golden 0) V =
      wave([Rising_Golden_Waveform](T-TR),
           [Falling_Golden_Waveform](T-TF))
[End Define Test Load]
```

Test data, Test load

```
[Test Data] test2    ]
Test_Data_type single_ended

Driver_model Buffer1
Test_load Load1

[Rising Golden Waveform]
0.0s  25.2100mV  15.2200mV  43.5700mV
0.2ns  2.3325mV  -8.5090mV  23.4150mV
0.4ns  0.1484V   15.9375mV  0.3944V
0.6ns  0.7799V    0.2673V   1.3400V
0.8ns  1.2960V    0.6042V   1.9490V
1.0ns  1.6603V    0.9256V   2.4233V
```