

Changing the Paradigm for Compact Model Integration in Circuit Simulators Using Verilog-A

M. Mierzwinski, P. O'Halloran, B. Troyanovsky, R.
Dutton*

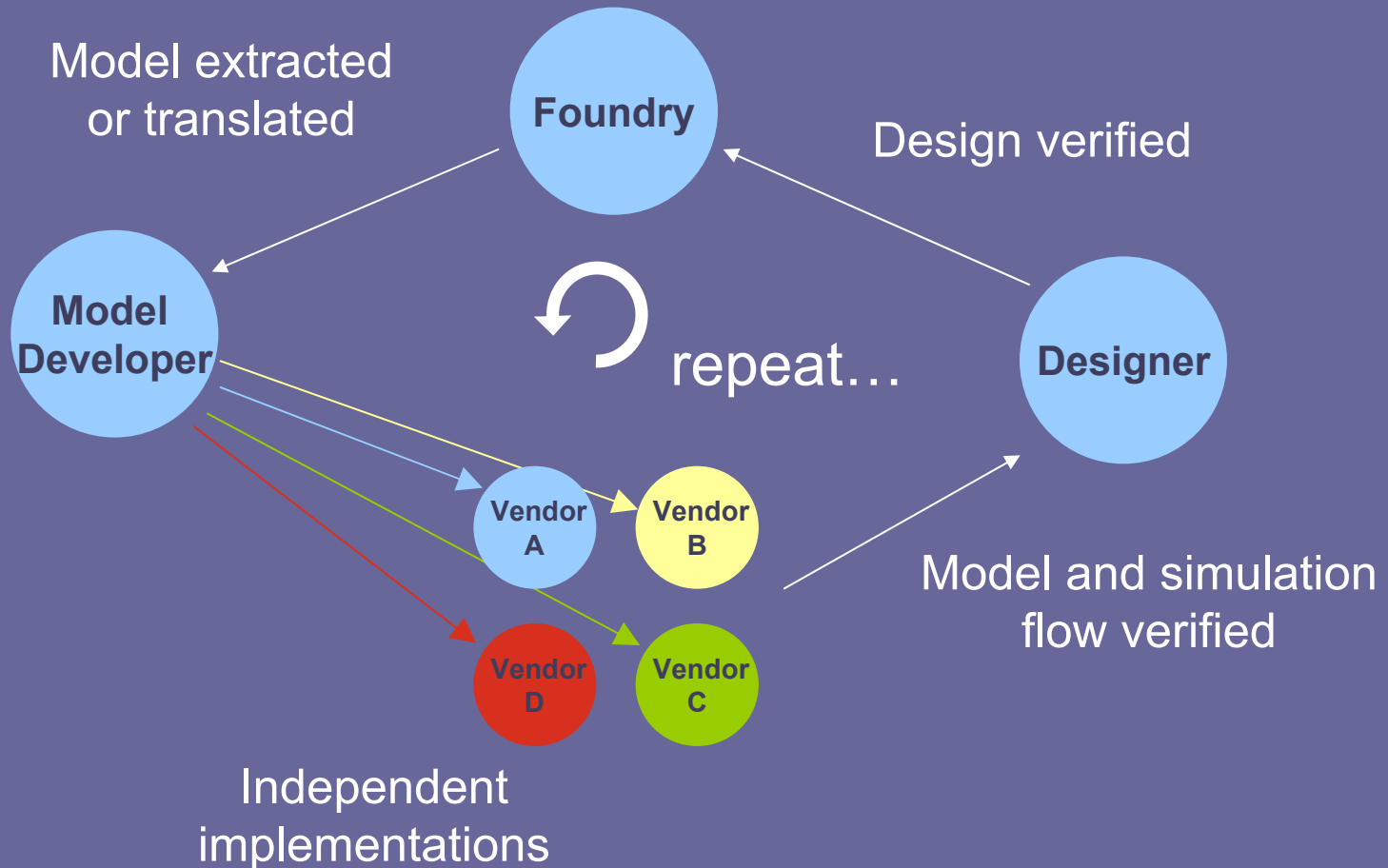
Tiburon Design Automation, Inc.

*Stanford University

Outline

- Current state of compact model development
- Verilog-A for compact modeling
- Examples of compact models in Verilog-A
- The paradigm shift
- Summary and conclusions

Typical model flow



Model Developer issues

- Complexity in model development has shifted from model creation to model implementation
- Fewer models implemented
- Models become more general to try to handle all processes

Model Developer issues

- Simulator model interfaces that are powerful are complex.
- Model interfaces promote errors due to complex math and program requirements.
- Software engineering aspects of implementation are time consuming.

Model Developer Issues

- Existing source-code interfaces are inherently non-portable
- High burden on the developer to:
 - Manually compute matrix stamp
 - Manually compute derivatives
 - Implement multiple, self-consistent entry points for analysis types
- Requires close cooperation with the simulator vendor

End User Issues

- Want accurate, robust, tried and tested models.
- Want to use models confidently across different simulation tools.
- Want all models to simulate efficiently.
- Want a wide range of models to be available and easily obtainable.
- Want to easily modify the underlying equations to suite their particular need.

Vendor Issues

- Standard models, such as BSIM3, are a necessary part of any analog simulator
 - Supporting standard models offers no competitive advantage.
- Multiple implementations of same standard model increases risk of errors
 - Always requires verification and testing.
- Potential model differences prevent new customers from evaluating their product.

Foundry Issues

- Standard models often do not provide accurate enough representation of the foundry process
- Foundries do not have sufficient resources to support and verify all vendors' tools.
 - This can limit their customer base.

Verilog-A

- Natural language for model development
- Concise
- OVI standard (proposed to IEEE)
- Implemented in many simulators

Existing solutions

Current Verilog-A solutions

- Are not sufficiently fast for simulation with typical compact models
- Are not universally available
- May not offer IP protection
- Don't support all analysis types
- Do not provide a simple distribution mechanism

Proposed Solution

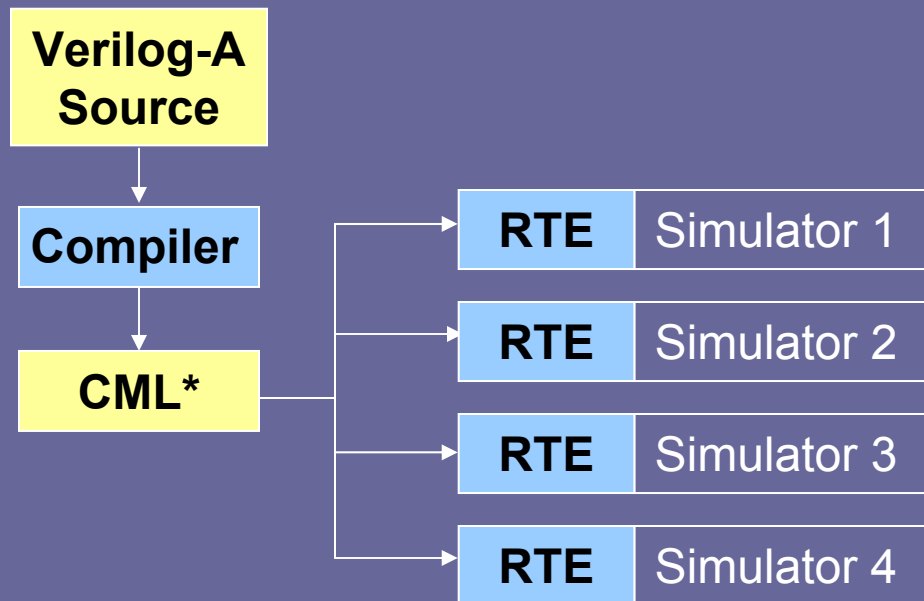
- Use Verilog-A for analog model definition
- Develop a Verilog-A model compiler and support tools to create analog models that can be used in a wide range of simulation platforms.

Proposed Solution

- Maintain simulation performance comparable to existing C/C++ level interfaces.
- Provide robustness better than existing implementations.
- Support for all analysis types, e.g. transient, harmonic balance, shooting, nonlinear noise.

Proposed Architecture

- A Verilog-A compiler and simulator-specific run-time environment (RTE)



* CML = compiled model library

Benefits to End Users

- Solves the problem of model availability and compatibility.
- Models with source code available can be modified.

Benefits to Vendors

- Removes need to support complicated standard models.
- Proprietary models can also be developed Verilog-A and distributed in compiled form
 - Provides IP protection

Benefits to Foundries

- Model definition can be part of foundry kit
 - Can be completely independent of simulators and simulator versions.
- One parameter set to extract, distribute, and support.
- Models can be modified to fit the foundry process
 - Parameter information and model changes can be hidden to protect IP.

Benefits to Model Developers

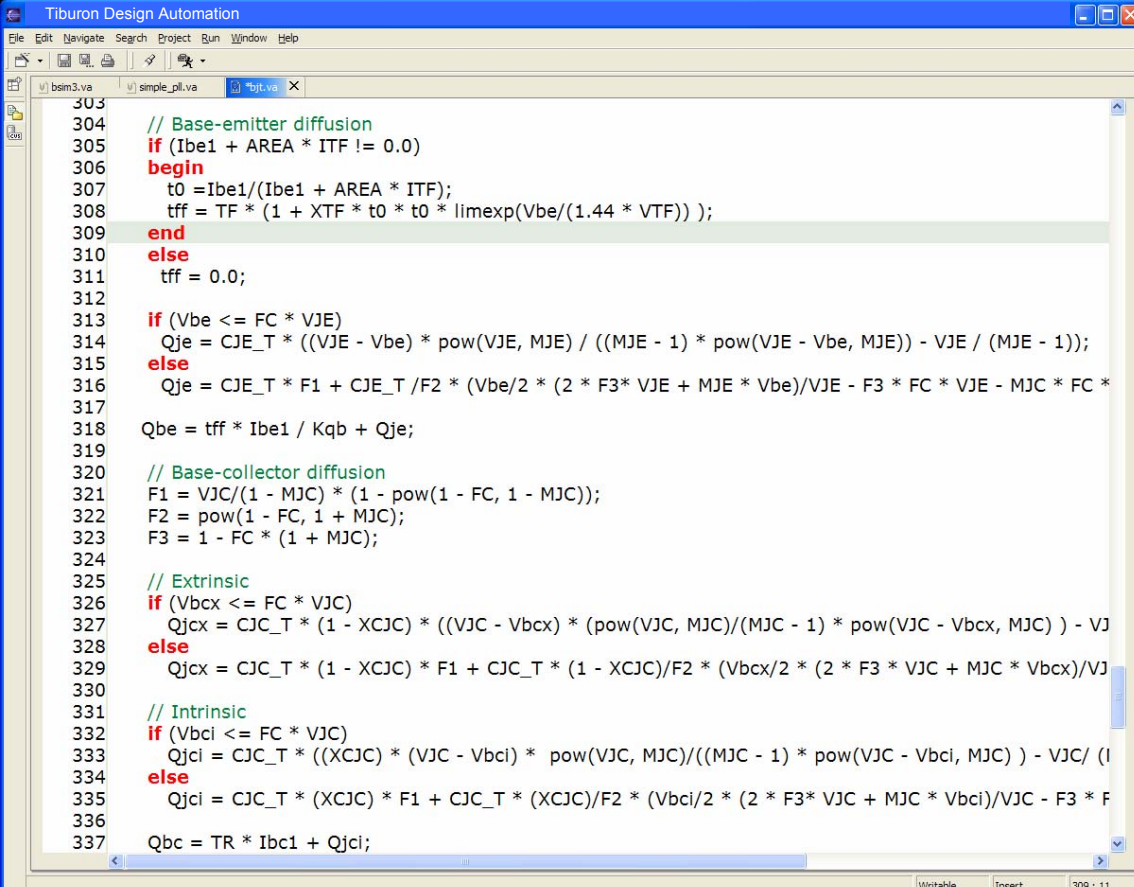
- Concentrate on model development, not implementation issues.
- Models and updates can be made available immediately on all platforms.
- Model IP can be protected and licensed.

Demonstration of Verilog-A based Compact Models

- Support for popular models is necessary for language acceptance
- Provides templates for generation of similar models in Verilog-A
- Demonstration of language capability
- RTE implemented in two simulators:
 - SPICE 3F5, HBsim.

Examples: Gummel-Poon BJT

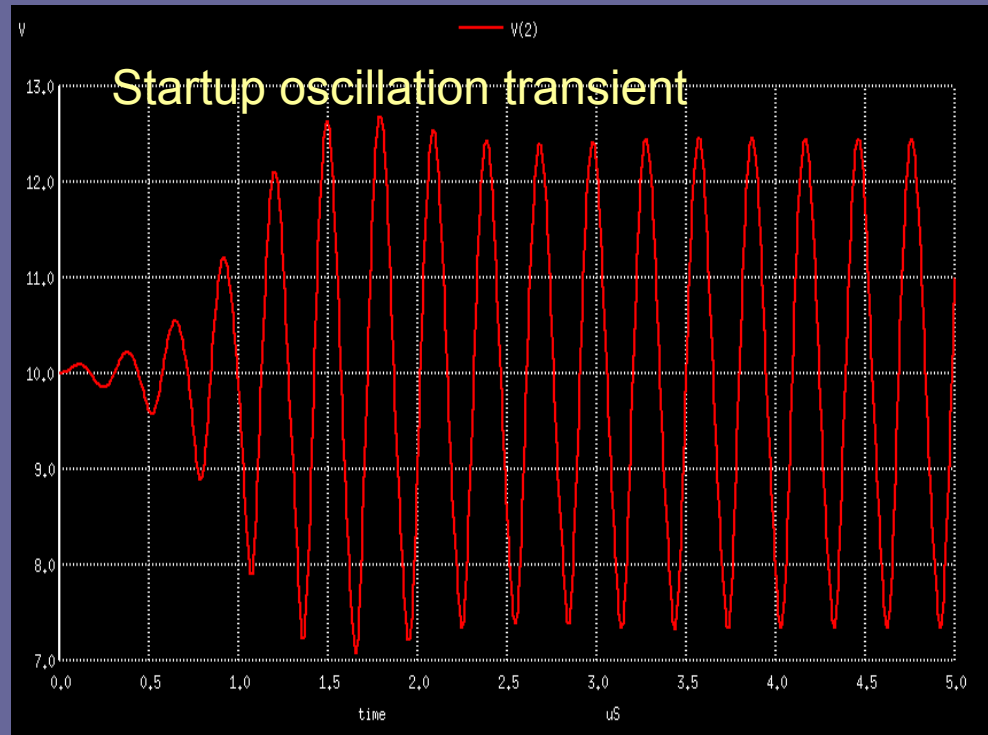
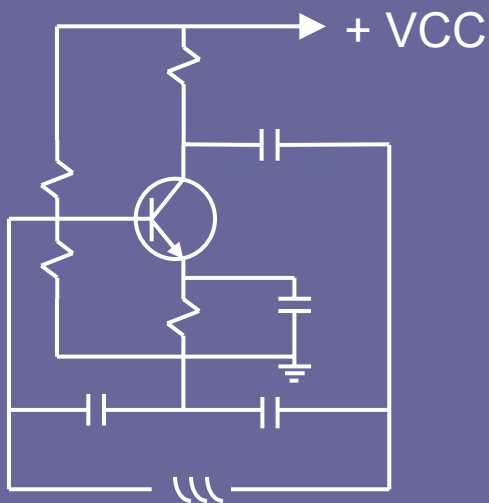
- SPICE Gummel-Poon
- Very old model
 - Yet very popular
- Users often like to modify model slightly
 - Kull-Nagel
 - Thermal
 - Avalanche
- ~300 lines in Verilog-A



```
303
304 // Base-emitter diffusion
305 if (Ibe1 + AREA * ITF != 0.0)
306 begin
307   t0 = Ibe1 / (Ibe1 + AREA * ITF);
308   tff = TF * (1 + XTF * t0 * t0 * limexp(Vbe / (1.44 * VTF)));
309 end
310 else
311   tff = 0.0;
312
313 if (Vbe <= FC * VJE)
314   Qje = CJE_T * ((VJE - Vbe) * pow(VJE, MJE) / ((MJE - 1) * pow(VJE - Vbe, MJE)) - VJE / (MJE - 1));
315 else
316   Qje = CJE_T * F1 + CJE_T / F2 * (Vbe / 2 * (2 * F3 * VJE + MJE * Vbe)) / VJE - F3 * FC * VJE - MJC * FC *
317
318 Qbe = tff * Ibe1 / Kqb + Qje;
319
320 // Base-collector diffusion
321 F1 = VJC / (1 - MJC) * (1 - pow(1 - FC, 1 - MJC));
322 F2 = pow(1 - FC, 1 + MJC);
323 F3 = 1 - FC * (1 + MJC);
324
325 // Extrinsic
326 if (Vbcx <= FC * VJC)
327   Qjcx = CJC_T * (1 - XCJC) * ((VJC - Vbcx) * (pow(VJC, MJC) / (MJC - 1) * pow(VJC - Vbcx, MJC)) - VJ
328 else
329   Qjcx = CJC_T * (1 - XCJC) * F1 + CJC_T * (1 - XCJC) / F2 * (Vbcx / 2 * (2 * F3 * VJC + MJC * Vbcx)) / VJ
330
331 // Intrinsic
332 if (Vbci <= FC * VJC)
333   Qjci = CJC_T * ((XCJC) * (VJC - Vbci) * pow(VJC, MJC) / ((MJC - 1) * pow(VJC - Vbci, MJC)) - VJC / (1
334 else
335   Qjci = CJC_T * (XCJC) * F1 + CJC_T * (XCJC) / F2 * (Vbci / 2 * (2 * F3 * VJC + MJC * Vbci)) / VJC - F3 * F
336
337 Qbc = TR * Ibc1 + Qjci;
```

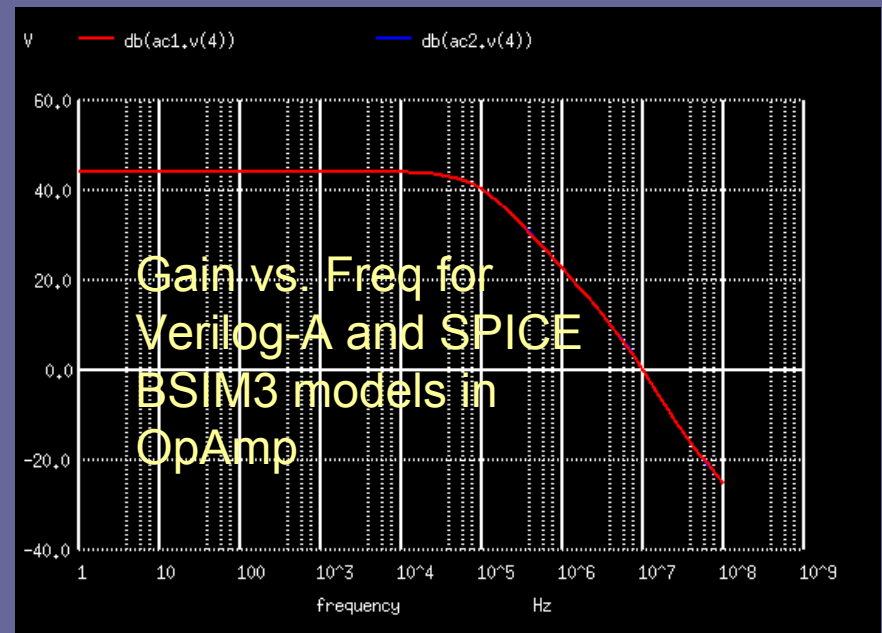
Examples: MEXTRAM BJT

- Philips MEXTRAM 504 BJT in Verilog-A
- Colpitts oscillator



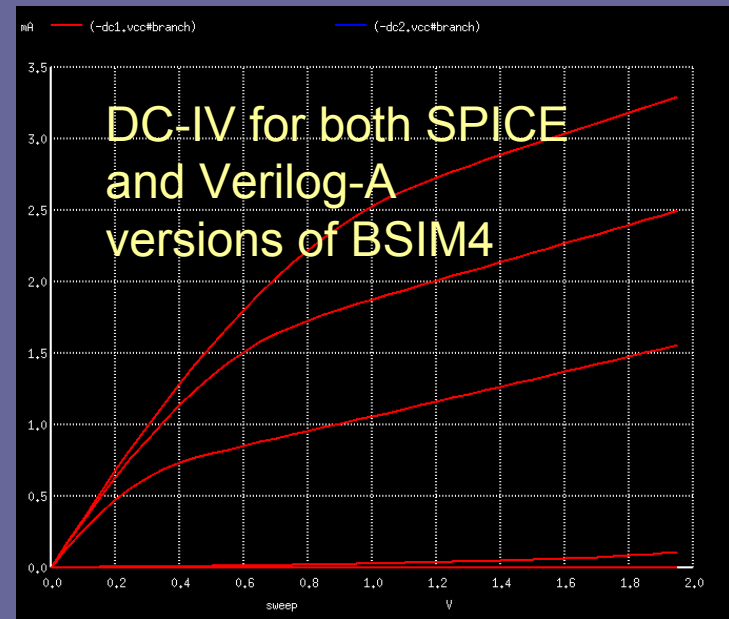
Examples: Compact models MOS

- UC Berkeley BSIM3 MOSFET
 - Only CMC “standard” model
 - Typically tens of thousands of lines of C-code
 - Example of simple op-amp using Verilog-A
 - Same results as SPICE built-in



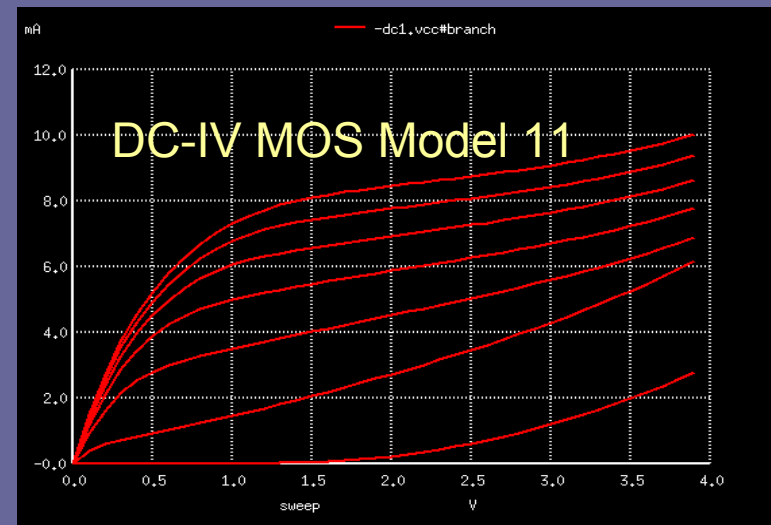
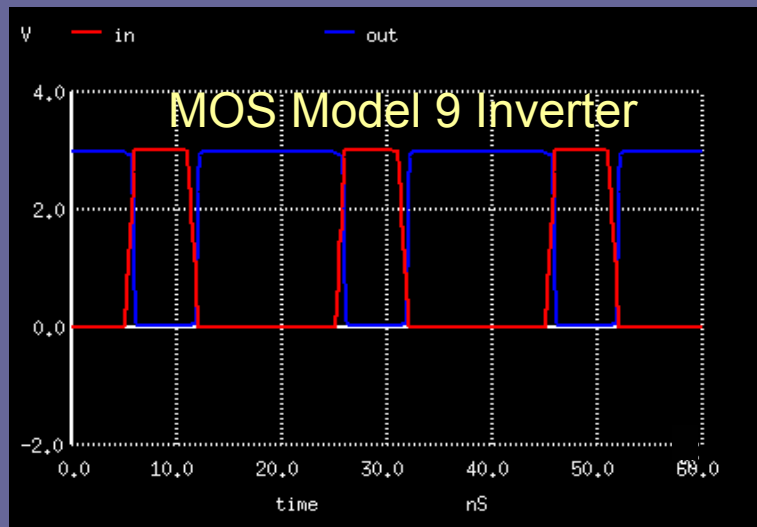
Examples Compact models MOS

- UC Berkeley BSIM4 MOSFET model
 - Next generation BSIM
 - Many more parameters
 - First released March 2000
 - April 2002 CMC still working on implementation problems between vendors.



Examples Compact models MOS

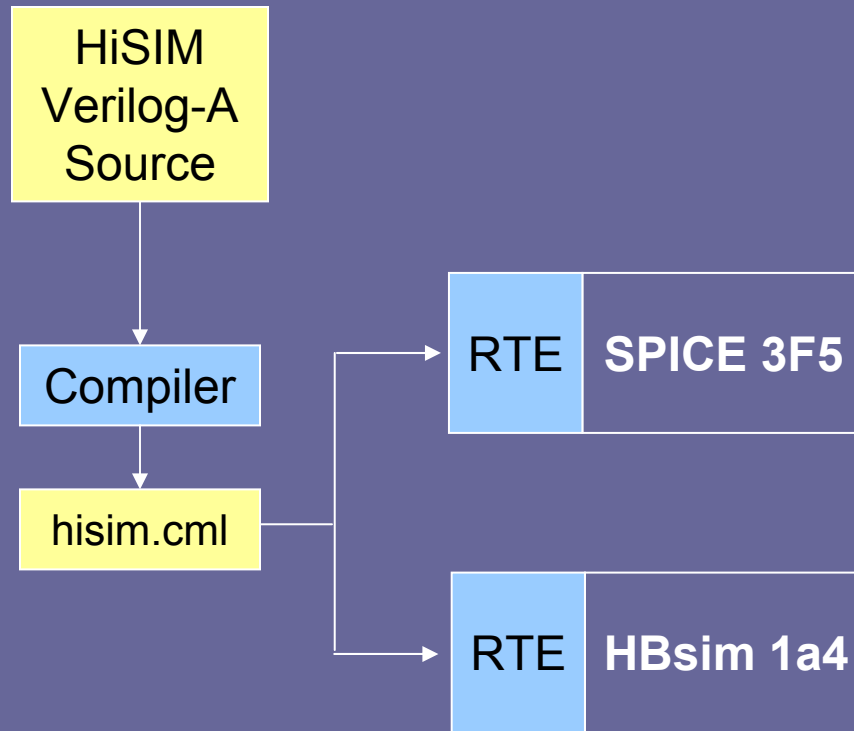
- Philips MOS Model 9
 - BSIM3 counterpart
- Philips MOS Model 11
 - Next generation MOS model



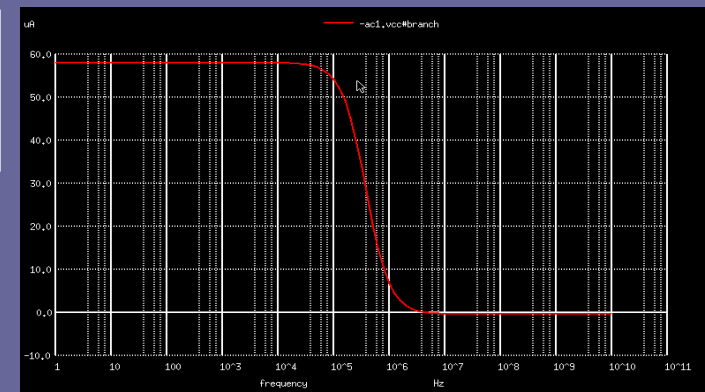
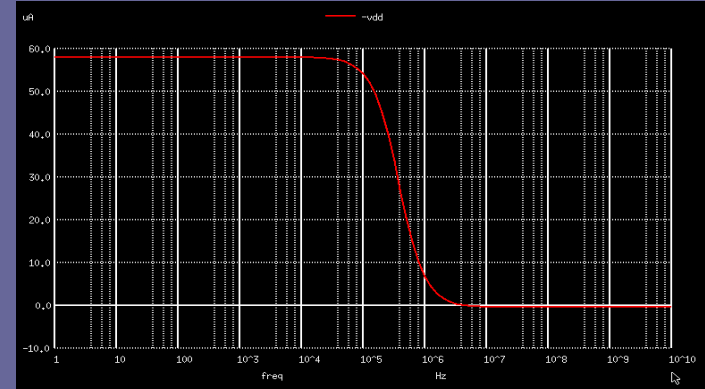
Example of single model library used in multiple simulators

- HISIM
 - Hiroshima University STARC IGFET Model
- Popular drift-diffusion/surface potential based model
 - Smaller parameter set size
 - Physically based
- Uses local Newton-Raphson loop

HiSIM Implementation

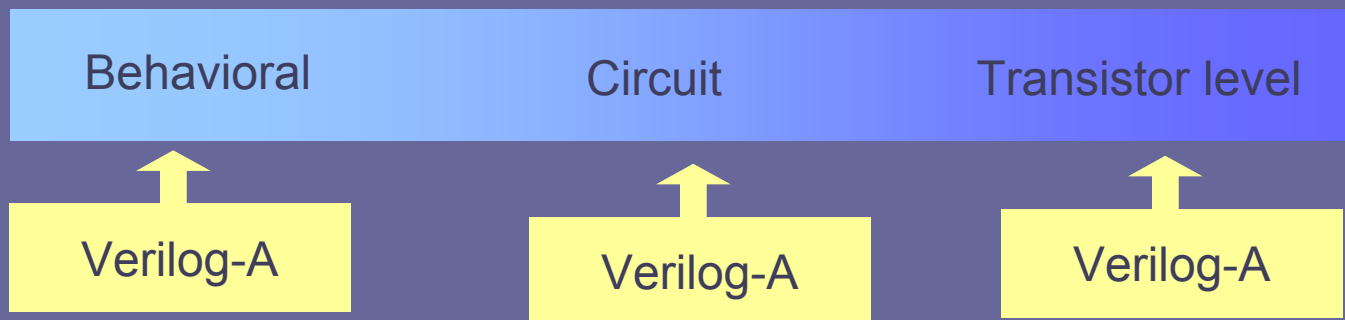


One compiled model library file shared by both simulators



New design capability

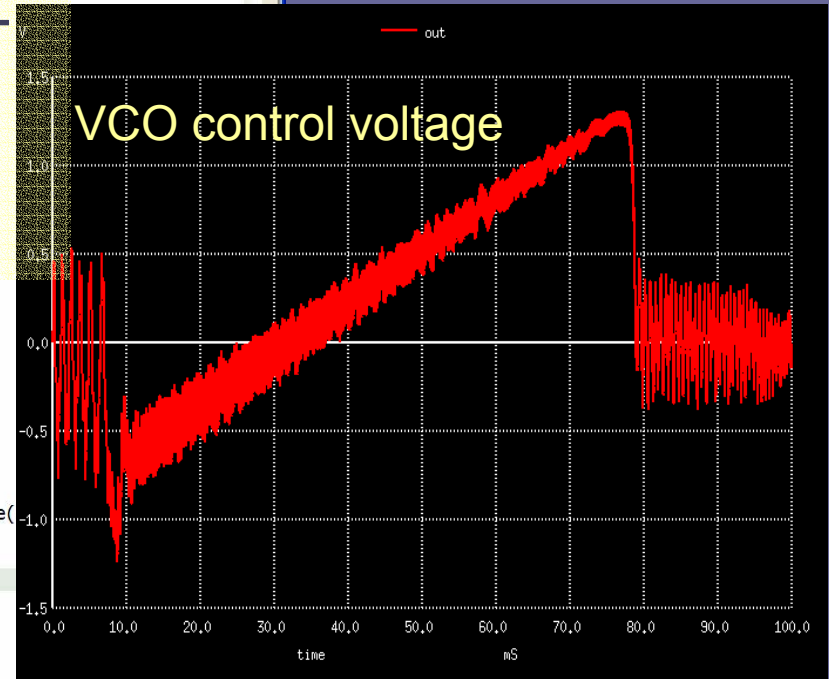
- Circuit designers can control level of abstraction
- All levels can be combined in one hierarchical implementation



Behavioral model example: PLL

```
Tiburon Design Automation
File Edit Navigate Search Project Run Window Help
Navigator
SpiceModules
  project
  bjt.va
  bsm3.va
  simple_pll.va
simple_pll.va
36 parameter real TC = 2.0K from (0:int);
37 real cap ;
38 real res ;
39 electrical lo;
40
41 phaseDetector #(.gain(2)) pd(0,0,if_);
42 vco #(.gain(loopGain/2), .fc(fc)) vco1(out, lo);
43
44 analog begin
45   cap = 150e-9;
46   res = tau/cap;
47
48   V(out, if_) <+ I(out, if_)*res;
49   I(out, ref) <+ ddt(cap*V(out,ref));
50 end
51 endmodule
52
53 // swept frequency sinusoidal source used as input to the PLL
54 module sinRampSrc(p,n);
55   inout p,n;
56   electrical p,n;
57   parameter real fStart = 1, hzPerSec = 1;
58
59   analog
60     V(p,n) <+ sin(2*_M_PI*(fStart+hzPerSec/2*$realtime())*$realtime(
61 endmodule
62
63 // Test module combining the PLL and the swept source
64 module testPLL(rf, out, ref,if_);
65   inout rf, out, ref, if_;
66   electrical rf, out, ref, if_;
67   parameter real tStop=1e-3, fStart=0, fStop=0;
68   ground gnd;
69
```

Complete PLL
behavioral
definition in
one library



Performance

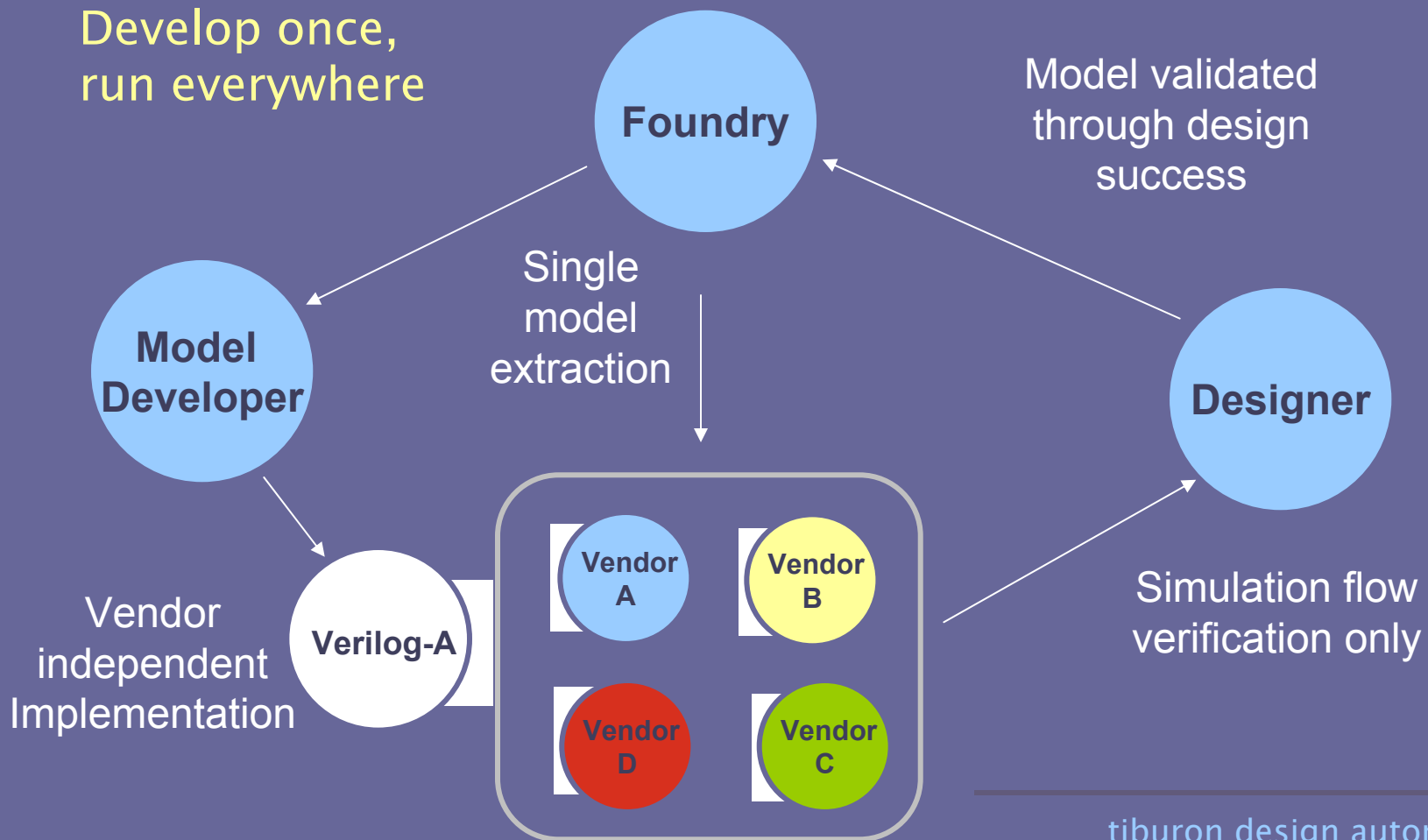
- Simulation speed of Verilog-A based compact models is critical for acceptance
- Myth of hand-coded equations

Model	Simulation	Total simulation time Verilog-A / SPICE
Resistor	DC	1.3
	Transient	1.2
Diode	DC	1.3
	Transient	1.5
BSIM3	DC	1.2
	Transient	1.5
BJT adder	Transient	1.6

Outstanding issues

- Verilog-A is quite capable of describing compact models
- Addition improvements
 - Parameter attributes
 - Additional program functionality
 - Function parameters
 - Local declarations

New Modeling Ecosystem



New Paradigm

- Model interface is standardized on Verilog-A
- Model definition no longer needs to be static
 - Fit model to process
 - Remove unneeded functionality of standard models
- Models can be a competitive advantage
- Models can have revenue potential
- Users can choose simulators based on analysis algorithms rather than model set.

Conclusions

- Verilog-A has the capability to support complex compact model implementations
- A Verilog-A compiler provides fast execution and support for all analysis types
- Virtually all popular compact models have been implemented as a demonstration.
- Model implementation and distribution can be greatly simplified.
 - Model developers, simulation vendors, end users all benefit.