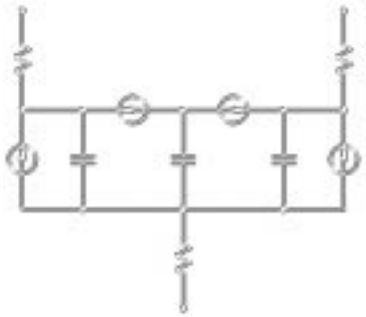
agere systems

Introduction
Model Compiler
  Architecture
  Language
MOSFET model
Bandgap model
Flip-Flop model
Conclusions



# An Extensible Compact Model Description Language and Compiler

**Richard V. H. Booth**
**Agere Systems**
**Allentown, PA**

agere systems




# Model Compiler Approach

- Single description/specification of all aspects of a model
  - Topology: nodes, elements and branch voltages
  - Model identification: naming, versioning, keycodes, polarity
  - Branch voltage initialization, limiting, and collapse
  - Temperature and geometry mapping
  - Element equations
  - Model parameters
  - Simulator, solver, and extractor specifications
- Multiple targets
  - Circuit Simulators
  - Solver
  - Parameter extractor/model playback tool
  - Model documentation

agere systems




# Advantages of Model Compiler Approach

- Development of model can be accomplished very quickly
  - Testing ideas using solver and parameter extractor targets
  - Testing of robustness of the model in circuit simulator targets
- Error-prone model development steps are done automatically
  - Automated dependency and derivative calculations
  - Automated array indexing for parameters, inputs, outputs, intermediate calculations
  - Checking model inconsistencies
  - Correct code generation
- The resulting target need not be any less efficient than a hand-coded model
- The model description/specification is much more easily maintained than target code
- The model is self-documenting
- Once interface to a new target has been developed for one model, all other models can be retargeted

agere systems

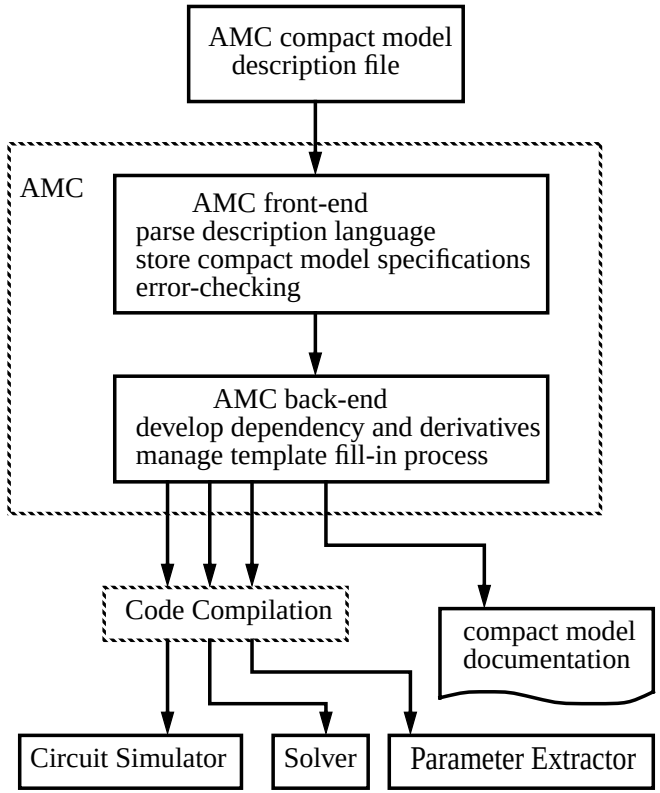



# What you can do with a Model Compiler

- Compact model development
  - Solver, extractor, playback-tool
  - Check circuit simulator performance
  - Elimination of numerical problems
- Compact model support
  - Parameter extraction
  - Documentation
  - Maintain code in multiple simulators
- Support for models in new simulator
  - Write template for new simulator
  - Add target-specific code if necessary
- Consulting support
  - What-if models
  - Simplified models for documentation
  - Simulator problem debugging

# Analog Model Compiler (AMC)

- Front-end: parse/store model information from description file
- Back-end: generate source-code/target-specific files



agere systems




# AMC

- AMC description file is an executable Tcl program
  - Each specification block corresponds to a (specification) method in the Model class
  - After model is specified (front-end), the compile procedure is called (back-end)
  - Target generation is done using template-driven process
  - Each target corresponds to a method in the Model class
- Some benefits of using Tcl/[incr Tcl]/Camelot
  - The Tcl scripting language is already well-documented
  - Model descriptions can make use of programming constructs and procedures provided by Tcl
  - Partitioning of the front-end is naturally partitioned into classes which store and provide information
  - No new parsing engine is required beyond capabilities of Tcl
  - Much error-checking is provided for free by Tcl
  - Adding new capability, checking, and targets doesn't require any compilation

# Template-driven target generation

- Most of the template is usually verbatim
- Directives appear at beginning of line as “%-<word>”
  - %-IF %-ELSEIF %-ELSE %-ENDIF
  - %-TCL %-ENDTCL
- Substitution directives appear anywhere within verbatim text as “%<symbol>”
  - Replacement symbol can be established by target-specific Model code
  - Replacement symbol can be established within template %-TCL block by the “export” command
- TCL blocks
  - Symbols exported for use by following substitution directives
  - Strings output to target file using “output” command

agere systems




# Template example

```
#-------------------------------------------------------------------------
# Parameter specifications
#-------------------------------------------------------------------------
set PARA_NAMES {}
foreach spec {
%-TCL
  set fmt "  {%s %g %g %g}"
  foreach name $pinp_pars {
    set nom [$PAROBJ lookup $name nominal]
    set min [$PAROBJ lookup $name minimum]
    set max [$PAROBJ lookup $name maximum]
    output [format $fmt $name $nom $min $max]
  }
%-ENDTCL
} {
  lappend PARA_NAMES [set name [lindex $spec [set i 0]]]
  set PARA_RANG($name) [lrange $spec [incr i] end]
}
```

agere systems




# AMC

- Targets
  - Celerity
  - ADVICE
  - Spectre
  - VHDL-AMS
  - Verilog-AMS
  - input decks for Celerity and ADVICE
  - Solver
  - Parameter extractor
  - HTML documentation
- Code size
  - Library: 6000 lines
  - Templates: 12000 lines

agere systems




# AMC model description language

| Specification | Description |
|---|---|
| componentSpecs | model family/component/member aspects |
| nodeSpecs | nodes in the equivalent circuit |
| branchSpecs | branches in the equivalent circuit |
| elementSpecs | elements in the equivalent circuit |
| branchInit | branch initialization |
| branchLimit | branch limiting |
| branchCollapse | branch collapse conditions |
| temperatureEqns | temperature-mapping equations |
| geometryEqns | geometry-mapping equations |
| elementEqns | element/noise/ac equations |
| parameterSpecs | model parameters |
| simulatorSpecs | simulator-specific configuration |
| solverSpecs | solver-specific configuration |
| extractorSpecs | extractor-specific configuration |
| simulatorOutput | simulator output report format |
| extractorData | extractor data format and screening |
| extractorFrame | extraction frames |
| verbatimLines | pass-through code |

agere systems




# MOSFET model - 1

```
compact-model ymos {
  componentSpecs {family mosfet component ymos version 1.0}
  nodeSpecs external {d "Drain"; g "Gate"; s "Source"; b "Bulk"}
  branchSpecs {
    Vds d s "Drain to source voltage"
    Vgs g s "Gate to source voltage"
    Vbs b s "Bulk to source voltage"
  }
  elementSpecs {Ids d s}
  elementEqns Ids {
    rev  = ISLTZ(Vds)
    rsp  = sqrt(PHI-MIN(Vbs-Vds*rev,0))
    abc  = 1+GAMMA/(2*rsp)
    Vsth = SST/1151
    Vgse = Vgs-VTH-GAMMA*(rsp-sqrt(PHI))+(ETA-(1+2*ETA)*rev)*Vds
    Vgsl = Vsth*log(1+exp(Vgse/Vsth))
    Vdss = Vgsl/(abc+THETAC*Vgsl/2)
    Vdsl = MAX(MIN(Vds,Vdss),-Vdss)
    vmob = 1+THETAS*(Vgsl+2*GAMMA*rsp)
    hmob = 1+THETAC*ABS(Vdsl)
    cls  = 1+LAMBDA*ABS(Vds-Vdsl)
    Ids  = BETA*cls/(vmob*hmob)*(Vgsl-abc*ABS(Vdsl)/2)*Vdsl
  }
  parameterSpecs instance {
    {BETA   1e-3 1e-4 0.1  A/V^2   "transconductance parameter"}
    {VTH    0.5  0.2  2.0  V       "threshold voltage at Vbs=0"}
    {PHI    0.7  0.5  1.2  V       "strong-inversion surface potential"}
    {GAMMA  0.7  0    2    sqrt(V) "body-effect parameter"}
    {LAMBDA 0.01 0    0.5  1/V     "output-conductance parameter"}
    {THETAS 0.01 0    5    1/V     "verti. field mobility parameter"}
    {THETAC 0.10 0    5    1/V     "horiz. field mobility parameter"}
    {ETA    0    0    0.2  #       "dibl parameter"}
    {SST    80   20   200  mV/dec  "subthreshold slope"}
  }
```

# MOSFET model - 2

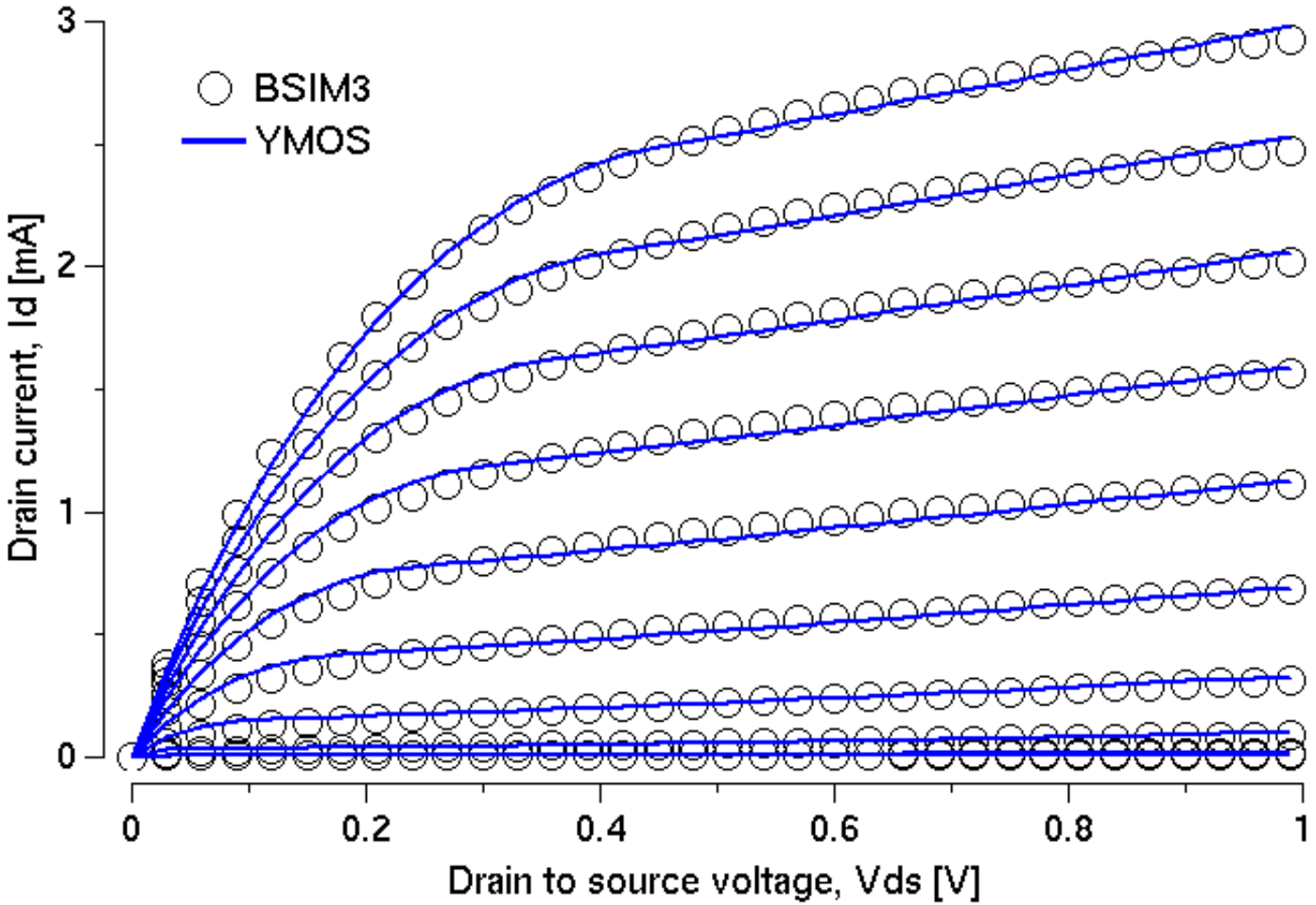
```
  extractorData ts {
    interface {MODEL = Id*1E6}
    screen    {MEAST = Id*1E6}
    plots     {xy Vd MEAST MODEL -xtitle Vds -ytitle Id}
  }
  extractorFrame TS {
    fit       {ts}
    evaluate {ts th st}
    residual {conductance}
    include  {LAMBDA THETAC}
  }
}
eval amc-compile $argv
exit 0
```

- No temperature or geometry model
- YMOS is symmetric about drain and source
- YMOS can be used as a “documentation” model
- Fit YMOS to simulations using Foundry-supplied BSIM3 models for 1.0V n-channel 0.13μm channel-length MOSFET from TSMC CL013LV process

agere systems



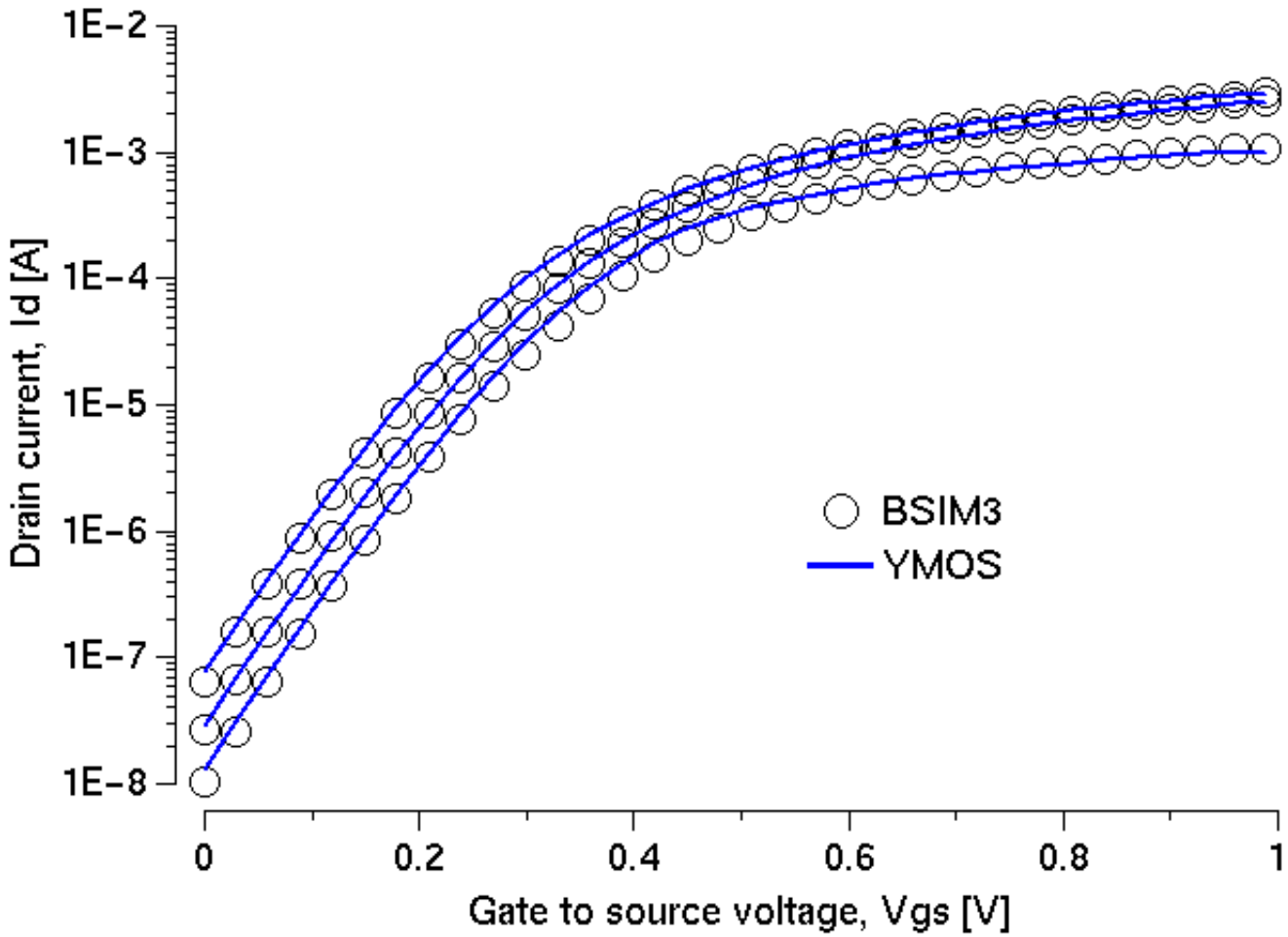

# YMOS output characteristics



agere systems

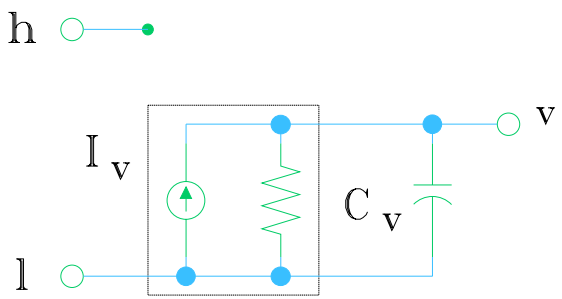



# YMOS subthreshold characteristics



# Bandgap reference circuit compact model - 1

```
Macro ISGT {X Y} {
  ISGT = 0.5*(1-signum(Y-X))
}
Constant Tabs 273.16
compact-model bg {
  componentSpecs {
    family bg
    component bg
    version 1.0
  }
  nodeSpecs external {
    v "bandgap reference voltage"
    h "high reference"
    l "low  reference"
  }
  branchSpecs {
    Vsup h l
    Vv   v l
    Vq   v h
  }
  elementSpecs {
    Iv   v l
    Qv   v h
  }
```



agere systems





# Bandgap reference circuit compact model - 2

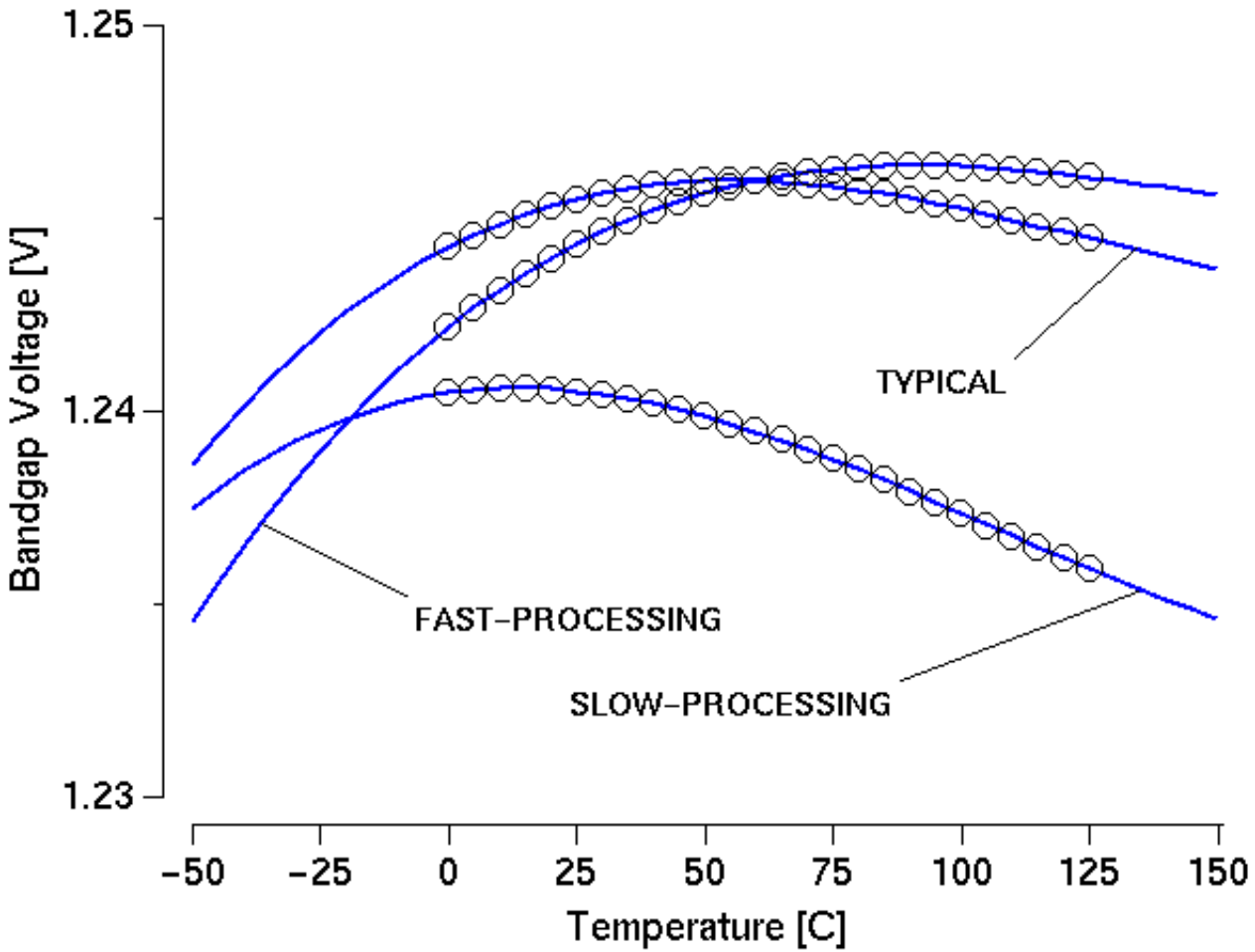
```
    parameterSpecs process {
      {VF   1.2   0.5     2       V    "flat voltage"}
      {TF    65   -50   125       C    "flat temperature"}
      {A1     0   -20    20       #    "temperature coefficient 1"}
      {A2    50     0   200       #    "temperature coefficient 2"}
      {A3    50     0   200       #    "temperature coefficient 3"}
      {VDO  2.0     0    20       V    "drop-out voltage"}
    }
    parameterSpecs temperature_derived {VREF}
    temperatureEqns {
      tn = (TEMPERATURE-TF)/(TF+Tabs)
      VREF = VF*(1+1e-3*(A1+(-A2+A3*tn)*tn)*tn)
    }
    elementEqns Iv {Iv = Vv-VREF*ISGT(Vsup,VDO)}
    elementEqns Qv {Qv = Vq*1e-13}
  }

  eval amc-compile $argv
  exit 0
```

agere systems




# BG model temperature characteristics



# Flip-flop compact model - 1

```
Macro Dig {v vdd} {
  b   = v-vdd*0.5
  s   = signum(b)
  u   = exp(-460.5*s*b)
  Dig = 0.5*(1+s*(1-u)/(1+u))
}
Constant Glow 1e-8

compact-model ff {
  componentSpecs {
    family ff
    component ff
    version 1.0
  }
  nodeSpecs external {
    c "clock"
    d "data"
    r "reset"
    q "noninverted output"
    b "inverted    output"
    h "high reference"
    l "low  reference"
  }
  nodeSpecs internal {
    m "master output"
    s "slave  output"
  }
```

```
branchSpecs {
  Vsup  h l
  Vc    c l
  Vd    d l
  Vr    r l
  Vm    m l
  Vs    s l
  Vqu   q h
  Vqd   q l
  Vbu   b h
  Vbd   b l
}
elementSpecs {
  Im    m l
  Is    s l
  Iqu   q h
  Iqd   q l
  Ibu   b h
  Ibd   b l
  Qm    m l
  Qs    s l
  Qc    c l
  Qd    d l
  Qr    r l
}
```

agere systems

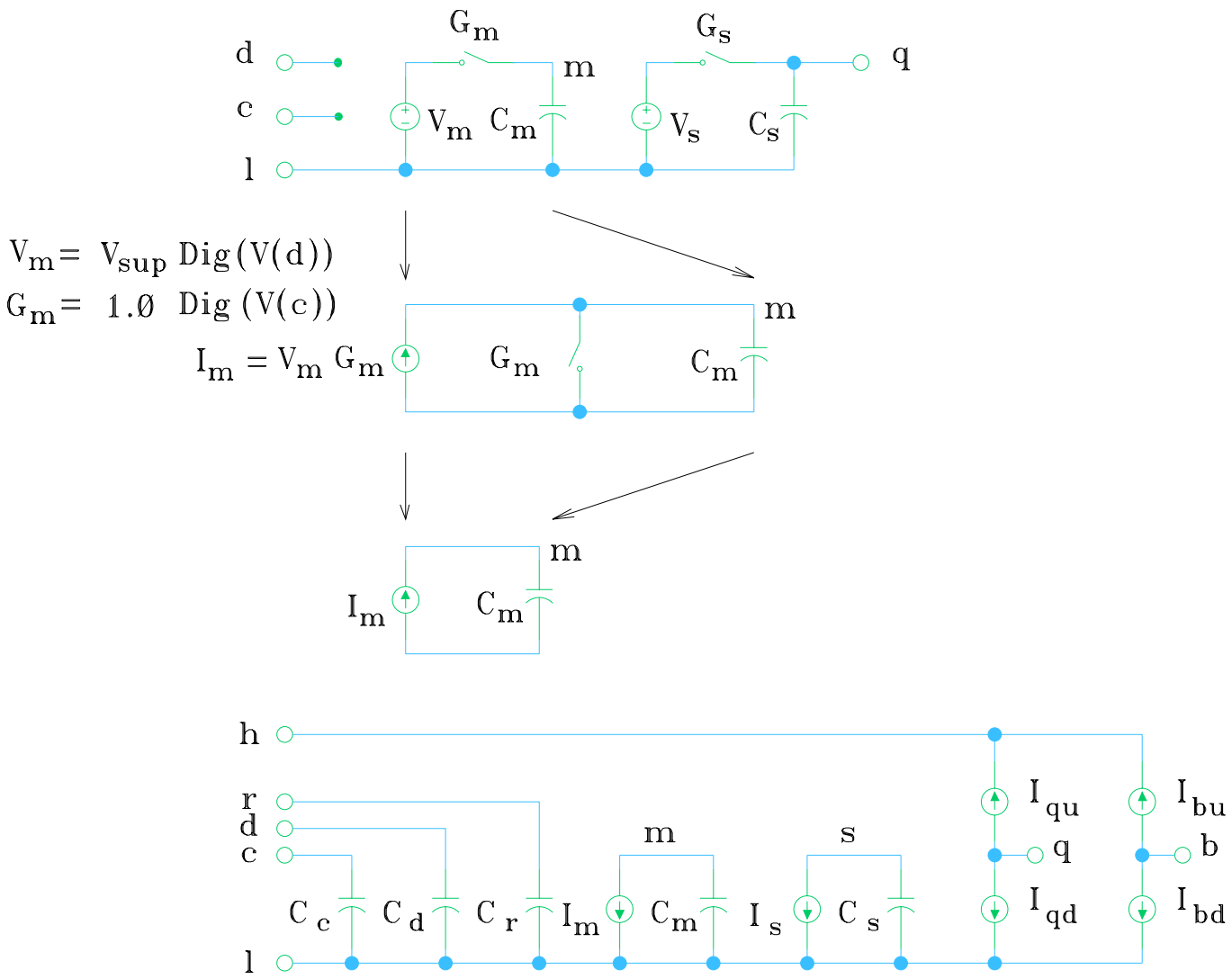



# Flip-flop compact model - 2

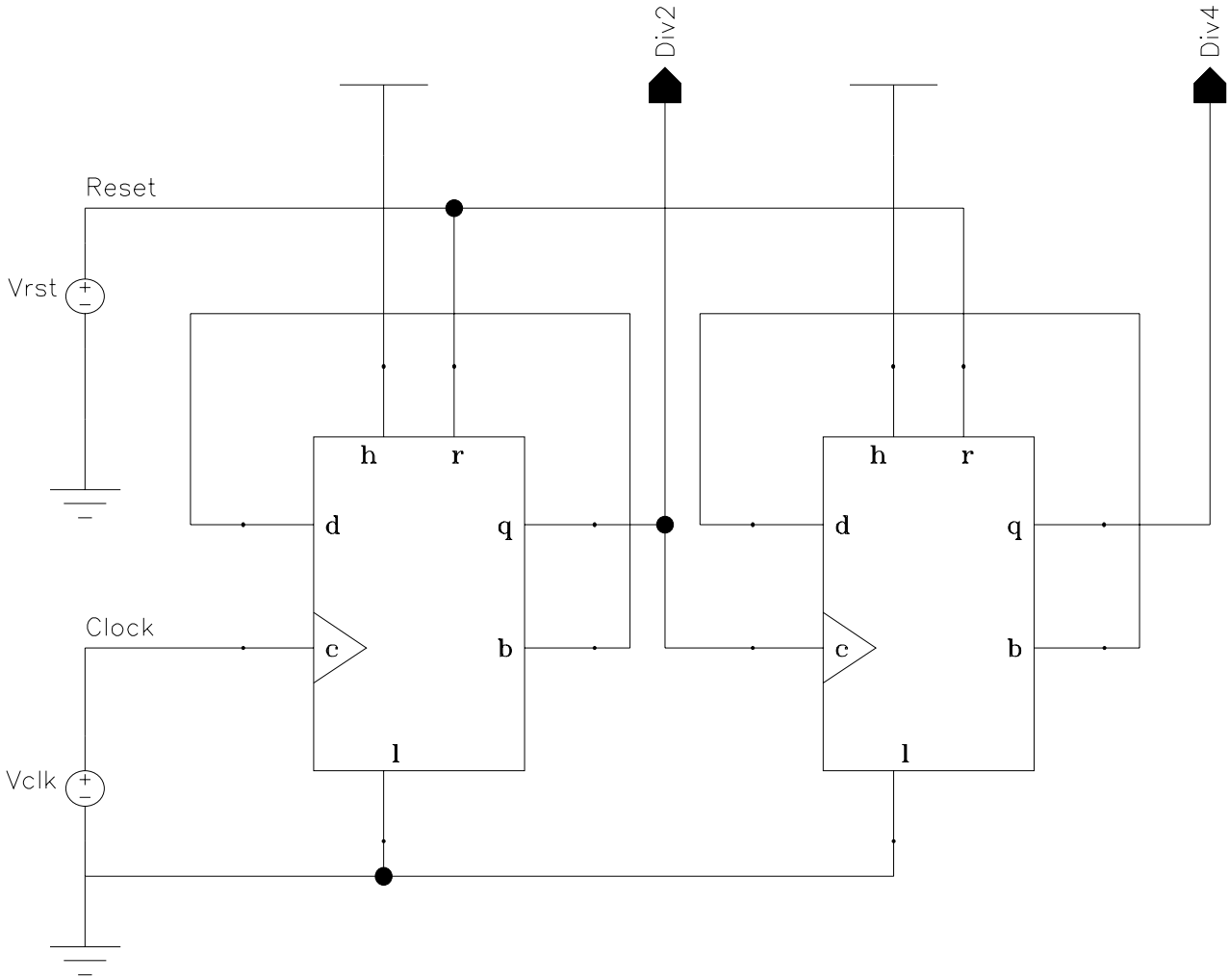
```
  parameterSpecs instance {
    {DELAY    1e-9  1e-12 100      s    "clock to Q delay"}
    {CIN      1e-14 0     1e-9     F    "input capacitance"}
  }
  elementEqns Im {
    chi = Dig(Vc,Vsup)
    rhi = Dig(Vr,Vsup)
    gr  = rhi+(1-rhi)*Glow
    Im  = (chi+(1-chi)*Glow)*(Vm-Vsup*Dig(Vd,Vsup)) + gr*Vm
  }
  elementEqns Is  {
    Is  = ((1-chi)+chi*Glow)*(Vs-Vsup*Dig(Vm,Vsup)) + gr*Vs
  }
  elementEqns Iqu {
    shi = Dig(Vs,Vsup)
    gq  = shi+(1-shi)*Glow
    gb  = (1-shi)+shi*Glow
    Iqu = gq*Vqu
  }
  elementEqns Iqd {Iqd = gb*Vqd}
  elementEqns Ibu {Ibu = gb*Vbu}
  elementEqns Ibd {Ibd = gq*Vbd}
  elementEqns Qm {Qm = DELAY*1.5*Vm}
  elementEqns Qs {Qs = DELAY*1.5*Vs}
  elementEqns Qc {Qc = CIN*Vc}
  elementEqns Qd {Qd = CIN*Vd}
  elementEqns Qr {Qr = CIN*Vr}
}

eval amc-compile $argv
exit 0
```

# FF model formulation



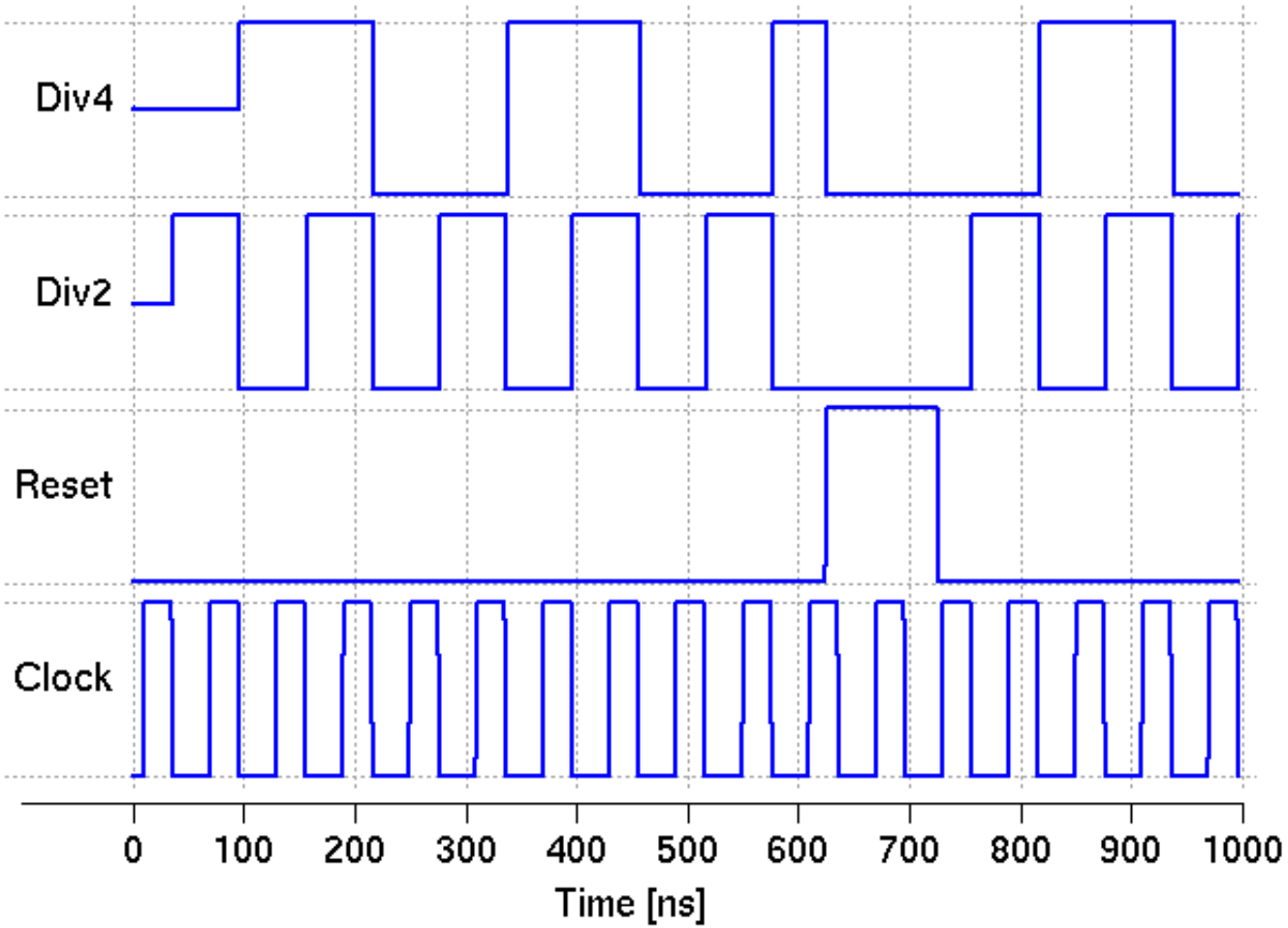
# Ripple-counter using FF model



agere systems



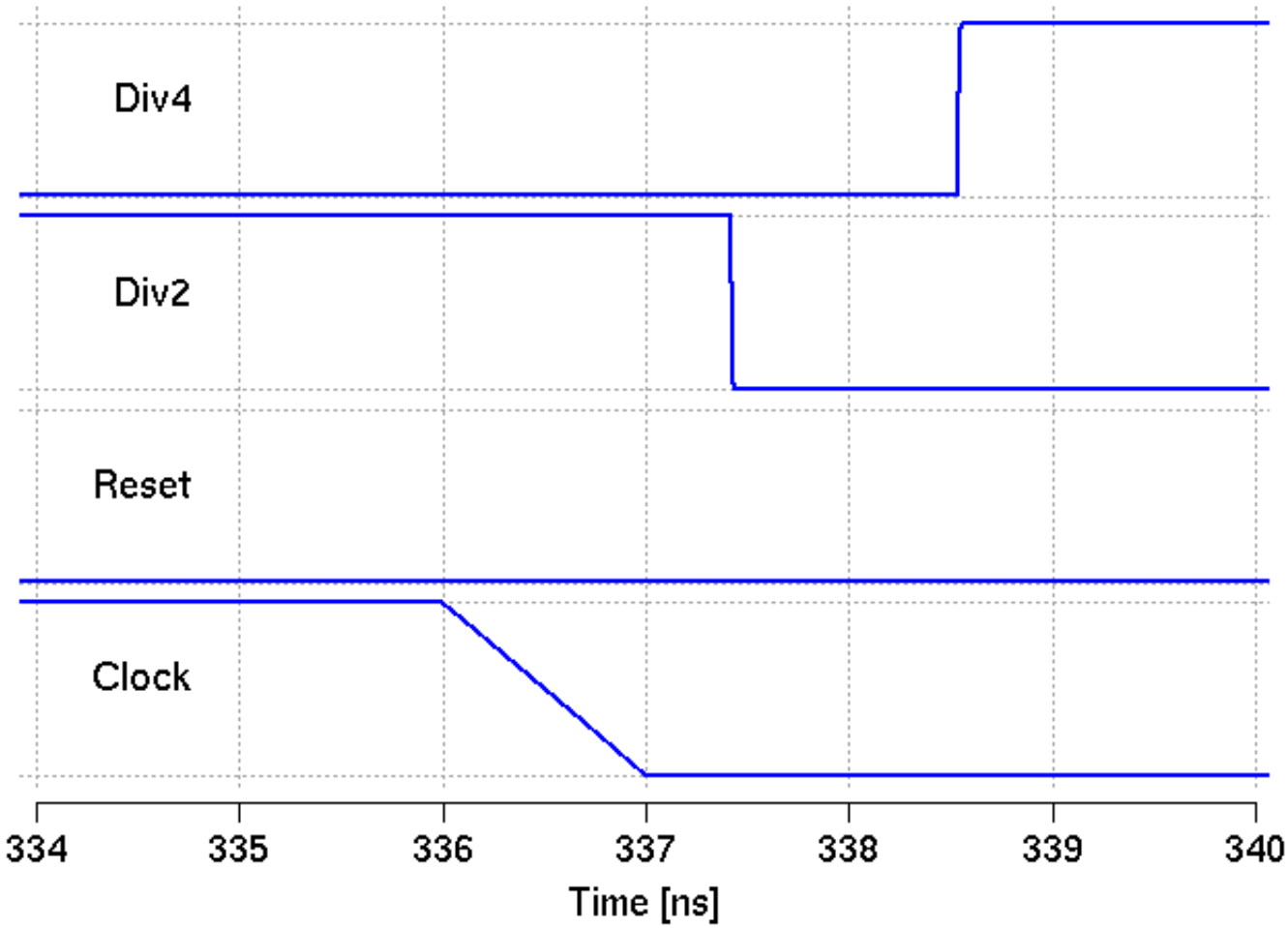

# Ripple-counter output signals - 1



agere systems




# Ripple-counter output signals - 2



agere systems



# Conclusions

- Model compiler technology has long been successfully used for device compact model development and support at AT&T, Lucent, Agere
- AMC can also be used for high-level circuit modules
- AMC for model development, model support, model porting, consulting support:
  - Fewer resources are required for development, maintenance, support, consulting
  - Only a single source must be maintained
  - Development is rapid and correct by construction
  - Support for other simulators can be accomplished quickly
- AMC approach - Tcl/Tk, [incr Tcl], templates - greatly improves the maintainability of the compiler itself

agere systems




# Acknowledgements

Kishore Singhal, Sani Nassif, Colin McAndrew, David Lee, Jeffrey Hantgan, Kartikeya Mayaram, Michael McLennan, Changlin Ma, Averill Bell, Shahriar Moinian, George Howlett, David Goldthorp, Russel Pierce, Margaret French, Kathy Krisch