MAPP: The Berkeley Model and Algorithm Prototyping Platform

Jaijeet Roychowdhury  Tianshi Wang

EECS Department
University of California at Berkeley
Motivation for MAPP

- Developing **good** compact models: many pitfalls
  - examples: discontinuities/smoothness, well-posedness (has transient solution?)
    - problems usually discovered at deployment (i.e., during simulation)
  - problems often hard to debug and resolve
    - compact model developer blames simulator
    - simulator people blame the model

- One goal of MAPP: to ease this problem
  - In MATLAB
    - empowers non-programmers to debug models
    - case studies + knowledge base → find solution to common problems (and try in MAPP) quickly
MAPP for Device Model Development

- **Compact Model Equations**
- **Write in MATLAB** (ModSpec format)
- **Test immediately (standalone)**
- **Run Small Circuits in MAPP**

- Problems? Yes
  - model doesn't evaluate
  - overflow/domain
  - DC conv. failure
  - transient timestep too small
  - unphysical results
  - voltage/current blows up

- Problems? No
  - model does't evaluate
  - DC conv. failure
  - transient timestep too small
  - unphysical results
  - voltage/current blows up

- format itself eliminates some common modelling mistakes

- DC/AC/TRAN in MATLAB
- code/facilities for inspection and debugging

J. Roychowdhury, University of California at Berkeley
Glimpse: ModSpec Device Definition

- Write in MATLAB (ModSpec format)
- Test immediately (standalone)
- Run Small Circuits in MAPP

Format itself eliminates some common modelling mistakes

Compact Model Equations

```
function MOD = diodeCapacitor_ModSpec_wrapper()
  MOD = ee_model();
  MOD = add_to_ee_model(MOD, 'external_nodes', {'p', 'n'});
  MOD = add_to_ee_model(MOD, 'explicit_outs', {'ipn'});
  MOD = add_to_ee_model(MOD, 'params', {'C', 2e-12, 'Is', 1e-12, 'VT', 0.025});
  MOD = add_to_ee_model(MOD, 'f', @f);
  MOD = add_to_ee_model(MOD, 'q', @q);
end

function out = f(S)
  v2struct(S);
  out = Is*(exp(vpn/VT)-1);
end

function out = q(S)
  v2struct(S);
  out = C*vpn;
end
```

Fig. 1: ModSpec definition example in MATLAB.
Compact Model Equations

Write in MATLAB (ModSpec format)

Run Small Circuits in MDE (Model Development Environment)

Test immediately (standalone basic debug)

➔ format itself eliminates some common modelling mistakes

Glimpse: Circuit Netlist in MAPP
MAPP Model Development Flow (2)

Step 1: model developed in ModSpec/MAPP

ModSpec Model (in MATLAB) hand-coded

translate manually
to NEEDS-compatible Verilog-A

compare code, check consistency

double-check in MAPP

end of Step 2: high level of confidence Verilog-A model is correct/debugged

ModSpec Model (in MATLAB) auto-generated from Verilog-A

Step 2

NEEDS-compatible Verilog-A model

automatic translator
// Series RLC
// Version 1a, 1 June 04
// Ken Kundert
// Downloaded from The Designer's Guide Community
// Taken from "The Designer's Guide to Verilog-AMS"

`include "disciplines.vams"

module series_rlc2 (p, n);
    parameter real r=1000; // resistance (Ohms)
    parameter real l=1e-9; // inductance (H)
    parameter real c=1e-6; // capacitance (F)
    inout p, n;
    electrical p, n, i;
    branch (p, i) rl, (i, n) cap;

    analog begin
        V(rl) <+ r*I(rl);
        V(rl) <+ ddt(l*I(rl));
        I(cap) <+ ddt(c*V(cap));
    end
endmodule

function MOD = series_rlc2_ModSpec_wrapper ()
    MOD = ee_model();
    MOD = add_to_ee_model (MOD, 'terminals', {'p', 'n'});
    MOD = add_to_ee_model (MOD, 'explicit_outs', {'ipn'});
    MOD = add_to_ee_model (MOD, 'internal_unks', {'Irl'});
    MOD = add_to_ee_model (MOD, 'parms', {'r', 1000}); // resistance (Ohms)
    MOD = add_to_ee_model (MOD, 'parms', {'l', 1e-9}); // inductance (H)
    MOD = add_to_ee_model (MOD, 'parms', {'c', 1e-6}); // capacitance (F)
    MOD = add_to_ee_model (MOD, 'fqei_all', @fqei);
    MOD = finish_ee_model(MOD);
end

function [Vrl, Frl, Icap, Qcap] = series_rlc2(Irl, Vcap, r, l, c)
    Vrl = r*Irl;
    Frl = l*Irl;
    Icap = 0;
    Qcap = c*Vcap;
end

function [fe, qe, fi, qi] = fqei(S)
    v2struct(S);
    Vcap = vpn;
    [Vrl, Frl, Icap, Qcap] = series_rlc2(Irl, Vcap, r, l, c);
    fe(1,1) = Irl + Icap;
    qe(1,1) = 0 + Qcap;
    fi(1,1) = vpn - Vrl;
    qi(1,1) = 0 - Frl;
end
MAPP Model Development Flow (3)

Step 3

NEEDS-compatible Verilog-A model (from Step 2)

automatic translator

ModSpec Model (C++ API)

confirm model with DC/AC/TRAN in C++ MAPP

use model in Commercial Simulators

model supported in Open-source Simulators (Xyce)

“simulation-ready” model deployed

proof of model goodness
Glimpse: ModSpec Model in Xyce

1 *** Test-bench for generating dc response of an inverter
2 *** Create sub-circuit for the inverter
3 .subckt inverter Vin Vout Vvdd Vgnd
4
5 yModSpec_Device X1 Vvdd Vin Vout Vvdd MVSmod_type=-1 W=1.0e-4
6 Lgdr=32e-7 dLg=8e-7 Cg=2.57e-6 beta=1.8 alpha=3.5 Tjun=300
7 Cif = 1.38e-12 Cof=1.47e-12 phib=1.2 gamma=0.1 mc=0.2
8 CTM_select=1 Rs0=100 Rd0 = 100 n0=1.68 nd=0.1 vxo=7542204
9 mu=165 Vt0=0.5535 delta=0.15
10
11 yModSpec_Device X0 Vout Vin Vgnd Vgnd MVSmod_type=1 W=1e-4
12 Lgdr=32e-7 dLg=9e-7 Cg=2.57e-6 beta=1.8 alpha=3.5 Tjun=300
13 Cif = 1.38e-12 Cof=1.47e-12 phib=1.2 gamma=0.1 mc=0.2
14 CTM_select=1 Rs0=100 Rd0=100 n0=1.68 nd=0.1 vxo=1.2e7
15 mu=200 Vt0=0.4 delta=0.15
16
17 .model MVSmod MODSPEC_DEVICE SONAME=MVS_ModSpec_Element.so;
18 .ends
19
20 *** circuit layout
21 Vsup sup 0 1
22 Vin in 0 0
23 Vsource source 0 0
24 X2 in out sup 0 inverter
25
26 *** simulation
27 .dc Vin 0 1 0.01
28 .print dc V(in)
29 *** END
30 .end

**Xyce netlist for inverter**
(using MVS ModSpec/C++ model)
ModSpec: Multi-Physics Support

Optical
Network Interface Layer
electric fields, polarizations, modes, wavelengths, wave continuity, ...

Electrical
Network Interface Layer
node voltages, branch currents, KCL, KVL

Mechanical NIL

Spintronic NIL

Biochemical NIL

Thermal NIL

ModSpec Core (Equations)

opto-electronic devices
Optical System Modelling/Simulation Example

$E_{in}$ → wave guide → splitter → wave guide → joiner → wave guide → light sink

sweep $n$ (reflective index) →

$|E_{out}|$ vs $n$ with $E_{in} = 1$, $l = 100 \mu m$

$|E_{out}|$ vs $n$ with $\lambda = 1550 nm$ and $\lambda = 1625 nm$
MAPP: First Public Release

- **Open Source download:** [http://mapp.eecs.berkeley.edu](http://mapp.eecs.berkeley.edu)
  - mailing list (MAPP announcements/discussion)
  - bug reporting and tracking site
  - git repository access (you can contribute)

- **License**
  - primary: GPL-v3
  - alternative licensing available
    - eg, SRC contract terms apply for SRC company use
  - contributors can specify their own alternative licensing terms for their contributions
MAPP: Features

- **Entirely in MATLAB**
- **Help system** *(start with `help MAPP`)*
  - quick start walk-through
- **Automatic differentiation** *(vecvalder)*
  - `help MAPPautodiff`
- **Executable device specification** *(ModSpec)*
  - examples, tutorial: part of help
- **DC, AC, transient analyses**
  - also noise, homotopy, HB, shooting, PPV, MOR, etc. (not released yet)
- **Automated testing system** exercising suite of tests
MAPP: Intended Uses

- **Developing simulation-ready device models**
  - including multi-physics devices, network connectivity

- **Quickly prototyping new simulation algorithms**
  - hours/days to implement a new analysis
    - assess strengths/limitations before investing resources to implement in “real simulators”
  - MAPP's structuring is different from SPICE's
    - devices “don't know about” algorithms; and vice-versa
    - central concept: DAEs (connect analyses and devices)

- **Learning or teaching modelling/simulation**
  - MATLAB → broadly accessible
  - help system, tutorials, supporting resources
mapp.eecs.berkeley.edu