Coho Circuit Modeling (Coho-Model) Tool Manual

Chao Yan

November 25, 2014 (Version 1.0)

Contents

Chapter 1

Introduction

A fundamental step of circuit simulation is to model circuit devices accurately and efficiently. Usually, circuit modeling is a complicate and tedious work for researchers and simulator developers. COHO *Circuit Modeling* (CCM) is a tool for extracting mathematical models from circuit layout automatically. It has features:

- CCM uses fabrication raw data, instead if SPICE-like models. SPICE-like models are abstraction of physical devices. They are not accurate enough for analog designs, especially for deep sub-micron process, *eg* 14nm. Due to the inaccuracy of these models, the simulation results usually do not match with real circuits. Our approach uses data from factory, thus could be arbitrary accurate.
- CCM provide mathematical models for both simulators and formal verification tools. For simulators, it provide ODE models; and for formal verification, it provides *Linear Differential Inclusion (LDI)* models. This enables to compare the results of simulation and reachability analysis.
- CCM is indepdent of the circuit simulator or reachability analysis tools (*e.g.* CRA). The interface is simple mathematical models *i.e.* ODE for simulators and LDI models for reachability analysis tools.
- CCM is very flexiable: easy to add or change device models or circuit libraires. It is also easy to be extended to support cutomized features.

1.1 Installation

CCM is open-sourced. Users can download from *github* by:

```
git clone https://github.com/dreamable/ccm.git ccm
```

It is purely MATLAB programs, thus support all operation systems. It can be installed by:

```
cd ccm
sh install.sh
```

1.2 Simple Usage

CCM is a MATLAB package. To use it, please first start MATLAB, then run your MATLAB codes by:

```
cra_open
%user_code
cra_close
```

1.3 Examples

Examples are available under the following directories in the source code:

- example
- MSpice/test
- libs/coho/test

1.4 Organization

2 describes the usage of CCM and its mathematical model interface for simulators reachability analysis tools. 3 describes the default circuit libraries and how to add users' own libraries. 4 describes the default device models and how to add new models.

1.5 Learn More

There are documents in the *doc* directories. There are also papers published which are good sources to understand the higher level ideas.

Table 1.1: Publications

Publication	**Note**
[1]	This is Chao Yan's PhD thesis. It is a comprehensive document with most details. Circuit modeling are covered in chapter 3.

To understand implementation details, please use the MATLAB help files by

```
help funcName
```

Chapter 2

MSpice

MSpice is the interface of CCM provided to users. It is based on two classes *circuit* and *testbench*.

Here is an example of how to simulate an inverter circuit

```
c = inverter('inv',1e-5);
p = vpulse('input',[0,vnn],[2,8,2,8]*1e-10,1e-10);
s = testbench(c,{c.i},{p});
[t,v] = s.simulate([0,1e-8]);
```

First, we create an *inverter* circuit, *'inv'* is the name of the circuit, and the size is 1e-5. Second, we proivde an input waveform by 'vpulse', where the rising/falling time is 2e-10, and high/low time is 8e-10. Then, we can create a testbench for the inverter circuit, with input connect to the vpulse. Finally, we can get the simulation by calling the *simulate* function.

2.1 Interface

The *circuit* class is the base class of all circuits. It provides the following functions

- Properties:
  - name: circuit name
  - nodeNum: number of circuit nodes
  - ports: visible circuit nodes
- Simulation interface:
  - i = I(v): node current
  - c = C(v): node capacitance
  - dv = dV(v): ODE models

  - v = V(t): node voltage, for voltage sources only

- Formal verification interface:
  - ldi = I_ldi(region): LDI models of node currents.
  - ldi = dV_ldi(region): LDI models of circuit dynamics.
- Small signal interface:
  - j = Jac(v): Jacobian matrix of circuit dynamics
- Utilities:
  - ifc_simu: support simulation interface
  - is_vsrc: is voltage source or not
  - ifc_verify: support verification interface
  - ifc_ssan: support small signal analysis interface
  - print_circuit_tree: show circuit hierarchy tree
  - print_nodes: show circuit nodes with descriptions

The *testbench* class is a wrapper of *circuit*, driving the input signals for constructing a complete simulation/verification environment. For each input signal, user should provide a voltage source to drive it. The *testbench* class provides:

- $[t, v]$ = simulate(tspan,v0,opt): default simulator
- dv = dV(v): full dimensional ODE models for simulator.
- ldi = dV_ldi(region): full dimensional LDI models for verification.

2.2 How to create new circuits

Circuit is a connected set of sub-circuits. To create a new circuit, a subclass of the *circuit* class should be defined. As illustrated in the *NAND* example below, there are three steps: 1. define the circuit nodes; 2. define the sub-circuits; and 3. define the connections.

```
classdef NAND < circuit
  properties (GetAccess='public', SetAccess='private');
    i1,i2; o; % input/output
  end
  methods
    function this = NAND (name,wid,rlen)
      this = this@circuit(name);  % super-class constructor
```

```
      % define circuit nodes
      this.i1  = this.add_port(node('i1'));
      this.i2  = this.add_port(node('i2'));
      this.o   = this.add_port(node('o'));
      x        = this.add_port(node('x'));

      % define sub-circuits
      n1 = nmos('n1',wid(1),0,rlen); this.add_element(n1);
      n2 = nmos('n2',wid(1),1,rlen); this.add_element(n2);
      p1 = pmos('p1',wid(2),1,rlen); this.add_element(p1);
      p2 = pmos('p2',wid(2),1,rlen); this.add_element(p2);

      % define connections
      this.connect(this.i1,n1.g,p1.g);
      this.connect(this.i2,n2.g,p2.g);
      this.connect(this.o,n1.d,p1.d,p2.d);
      this.connect(x,n1.s,n2.d,c.x);

      this.finalize; % complete
    end
  end
end
```

Chapter 3

Circuit Libraries

3.1 Coho Libraries

The default COHO libraries provides basic MOS circuits and RC circuits. For details, please check *./libs/coho*. The library supports all interfaces of the *circuit* class. It also support independent current sources, and voltage sources including: independent voltage source, voltage pulse, sine/cosine voltage waveform, and the Brockett's annulus for both simulation and verification.

3.2 How to add new libraries

New libraries should provides functionalities for all leaf-circuits, which have no sub-circuits. For non-leaf circuits, the *circuit* abstract class will provide all functions automatically. The leaf circuit subclass must override the following functions:

- I: (if ifc_simu & is_vsrc)
- C: (if ifc_simu & is_vsrc)
- V: (if ifc_simu & is_vsrc)
- V_ldi: (if ifc_verify)
- dIdV: (if ifc_ssan)
- Static.I_objs (for performance)
- Static.C_objs (for performance)
- vectorize_subtype_id (only if class may have different configurations)

To add the new library, the library path should be set by

```
ccm_cfg('set','libRoot','<libpath>');
```

Chapter 4

Device Models

4.1 Default models

CCM provide PTM process models by defaults. It provides two types of models: 'simu' and 'quad' models. The 'simu' type of models are raw data from factory process, which are essentially a huge table of sampled data. The 'quad' models are typically smaller. It use quadratic polynomial functions to approximate the raw data. It provide models for leaf circuits 'nmos', 'pmos', 'inverter', 'nmos with source connected to groud', and 'pmos with source connected to power'.

4.2 How to add new models

To add new device models, user need to add the path by

```
ccm_cfg('set','matRoot','<matpath>');
```

By default, the mat file is located by '¡matRoot¿/uppper(fab)/lower(process)/lower(type)/lower(circuit).mat', e.g './mat/PTM/180nm/simu/nmos.mat'. This can be changed by

```
ccm_cfg('set','matFileFunc','<func>');
```

Chapter 5

Advanced configuration

Please check ' help ccm_cfg.m' for details.

Chapter 6

Examples

Bibliography

[1] Chao Yan. *Reachability Analysis Based Circuit-Level Formal Verification.* PhD thesis, The University of British Columbia, 2011. 6