# User's Guide for INDUCTWISE (Ver. 1.00) *

## Inductance-Wise Interconnect Simulation Engine

### - A Multi-function Large-scale Linear Circuit Simulator

**Tsung-Hao Chen** and **Charlie Chung-Ping Chen**

VLSI-EDA Group (http://vlsi.ece.wisc.edu)

Electrical and Computer Engineering, University of Wisconsin-Madison

June 20, 2002

# Contents

**Abstract**

This manual describes INDUCTWISE (Inductance-Wise Interconnect Simulation Engine), a multi-function large-scale linear circuit simulator. This tool is built to efficiently analyze large-scale linear circuits, which is wildly adopted in modeling signal integrity problems such as interconnect simulations and power-grid analyses. The techniques that INDUCTWISE introduces include not only traditional MNA and NA simulation methods, but also excellent matrix solving algorithms, advanced circuit formulation techniques. INDUCTWISE is capable to simulate the novel technology, K-elements [3] (Sparsified inverse inductance matrix), which recently exposed great potential to capture the inductance effect on-chip; it also provides the option to simulate with our IEKS [1] (Improved Extended Krylov Subspace) method, which remarkably improves the transient runtime with almost the same accuracy compared to the exact solution.

This user's guide is broken into five sections. The Introduction section explores the motivation and background behind INDUCTWISE. The rest part of this user's guide describes the input file format, simulation commands, device statements, and how to run INDUCTWISE.

# 1 Introduction

## 1.1 Why Need A Fast Large-scale Linear Circuit Simulator?

While the benefits are enormous, higher circuit density, smaller devices and interconnect sizes, and higher clock rates enabled by the nanometer technologies also create difficult new to today's IC designers. The explosive increase in circuit size and complexity has broken some EDA tools and hence reduced users' ability to accurately predict design problems. For example, general-purpose circuit simulators such as SPICE is not capable to handle several million nodes, which is required by today's full-chip verification.

On the other hand, increased coupling noise causes serious signal integrity problems. No only on-chip coupling capacitances, but parasitic on-chip inductance is growing as another design concern as the VLSI technologies march toward ultra-deep sub-micron and frequency approaches in the gigahertz range. Inductive coupling effect becomes more important because of the higher frequency signal contents, denser geometries, reduction of both resistance and capacitance by copper and low-K devices. Inductance effects present not only in IC packages, but also in on-chip interconnects such as power grids, clock nets and bus structures. It causes overshoot, undershoot and oscillation of signal, aggravates crosstalk and power grid noise, which seriously impact the on-chip signal integrity.

However, the huge number of transistors on a chip causes timing analysis tools facing a fact that it has to solve enormous sparse matrices. General-purpose simulator such as SPICE is not capable to simulator a circuit with millions nodes. Moreover, while the PEEC models is adopted to capture the inductive effects, it leads to a large dense inductance matrix since the long range effects of inductance coupling, and the uncertainty of current return paths require the consideration of millions of mutual inductance terms to capture the inductance

effects. The traditional circuit simulation engines may require hours or even days for such a large-scale dense solution.

Several issues for the existing inductance handling flow. For example, after inductance extraction, it is required to perform circuit simulation to verify signal integrity issues. There is, unfortunately, a lack of effort to fundamentally speed up circuit simulation engine for inductance. Note that, one of the major reason for inductance sparsification is that the traditional circuit simulation engines can not handle large scale dense inductance matrix efficiently. Every sparsification algorithm is more-or-less trade-off runtime with accuracy. A capable circuit simulator can greatly enhance both the analysis of turn-around time and accuracy by including more mutual inductance terms. In addition, the traditional circuit simulation engines can not or do not handle the K-element directly or efficiently. All these issues lead to the urge for capable inductance optimized circuit simulation engines and that is exactly what we intended to develop, a fast large-scale linear-circuit inductance-oriented simulator, INDUCTWISE [2].

## 1.2 INDUCTWISE Features

This tool is built to efficiently analyze large-scale linear circuits, which is wildly adopted in modeling signal integrity problems such as interconnect simulations and power-grid analyses. INDUCTWISE (Inductance-Wise Interconnect Simulation Engine), as implied in the name is optimized for linear circuits with inductance presenting. It has the following features.

- **Excellent sparse matrix solver** to efficiently simulate large-scale linear circuit. SPICE was developed in 60 to 70's. The matrix solving techniques have been significantly improved in the last two decades. Furthermore, SPICE was mainly developed for human designed semiconductor circuits. It was not optimized for large scale parasitics simulation and did not took advantage of the linear nature of parasitics. INDUCTWISE has an excellent build-in sparse matrix solver which enable itself to perform simulation extremely efficiently in both runtime and memory usage.

- **Novel Technique K-Element** (reluctance or inverse inductance matrix) can be directly simulated by INDUCTWISE with an advanced system formulation. It is shown that using reluctance elements instead of PEEC model benefits a great efficiency on capturing on-chip inductance effect. This function is not capable in any other circuit simulator nowadays.

- **Improved Extended Krylov Subspace (IEKS) method** significantly reduces the transient simulation runtime. Computing the moments of piece-wise linear sources, solving the system moments, and utilizing the model order reduction method, IEKS dramatically improves the simulation speed with almost exact accuracy.

- **Moment computation function** enables INDUCTWISE to calculate moments of a circuit.

- **SPICE language compatibility** makes INDUCTWISE very easy to learn and use.

# 2 Input Conventions

When a simulation is run, the input file is interpreted by the INDUCTWISE parser. For INDUCTWISE to be learned easily and compatible as possible as other versions of SPICE, it is developed to follow exact the same language conventions. However, since INDUCTWISE simulator is under its early stage of development, the parser enforce some more restricted input format than SPICE. These are summarized in this section.

## 2.1 Name

All nodes and devices in the circuit must be identified uniquely by their names. Node and device names have the following features.

1. They are case sensitive. For example, N1345 is different from n1345.
2. Every name is separated by spaces, tabs or enters.
3. Other than the above, any symbol can be part of a name, such as semicolons (;), single quote ('), etc, which are not allowed in SPICE.
4. The length of a name is limited by 48 characters.

The following examples are all valid names:

```
abc    A156s    bus_135    alu/adder/ha1    n$1367
```

## 2.2 Comment

Comment provide information about the circuit, but are not processed as part of the formal circuit description.

INDUCTWISE parser is line-based, which reads in the input file line by line. It determines the type of each line by the first non-space character. If this character is an asterisk (*), the parser considers this line as a comment line. Otherwise, this line would be treated as a valid circuit (i.e. r, l, c) or command (i.e. .tran) expression. For example:

```
* Mash Power-Grid Circuit 100x100
```

is a comment;

```
R1355 n12 n68 10
.tran 10ps 100ps uic
```

are circuit descriptions.

## 2.3 Numbers and Units

Some commands and statements require representing physical quantities with attached units. such number can be express in floating point, scientific, or fixed-point notation, and can be followed by metric abbreviations indicating order or magnitude.

The base units such as s(second), v(volt), $\cdots$ are implicit from the context and are optional (actually not necessary at all). The text after valid numbers (0 $\sim$ 9, +, -, e, E) and the metric abbreviations would be ignored.

Acceptable metric abbreviations are listed as follows. They are non-case sensitive.

| Abbreviation | Prefix | meaning |
|---|---|---|
| t | tera- | $10^{12}$ |
| g | giga- | $10^{9}$ |
| meg | mega- | $10^{6}$ |
| k | kilo- | $10^{3}$ |
| m | milli- | $10^{-3}$ |
| u | micro- | $10^{-6}$ |
| n | nano- | $10^{-9}$ |
| p | pico- | $10^{-12}$ |
| f | femto- | $10^{-15}$ |

For example, the following expressions can all specify 2.3 nano-henry in an appropriate inductance expression.

```
2.3nh    2.3n    23e-10    2300p
```

## 2.4 Running INDUCTWISE

To run INDUCTWISE, simply type the following command under a command window (i.e. xterm in UNIX or cmd in Windows2000).

```
inductwise100 [input file] [output file]
```

Inductwise will read run the [**input file**] and write the output into [**output file**]. An execution example is shown as follows.

```
D:\inductwise100>inductwise100 test100.sp out.txt

******************** INDUCTWISE 1.00 ********************
****        Large-Scale Linear Circuit simulator        **
**** Programming: Tsung-Hao Chen (tchen@cae.wisc.edu) ****
**** Supervision: Charlie Chen   (chen@engr.wisc.edu) ****
****         University of Wisconsin-Madison          ****
****************** < UW VLSI-EDA Group > ******************
```

```
 Input filename : test100.sp
 Output filename : out.txt
 Number of Nodes = 10105
 Number of passive elements = 30103
 Number of mutual elements = 0
 Number of voltage sources = 4
 Number of current sources = 5000

Parse time =      2.123
initialize =      0.28
calculate basis = 3.956
reduce sys time = 0.3
simulation time = 1.442
output time =     0.02
---------------------------
Total runtime =   8.201
```

The circuit and runtime information will be displayed on the screen while output the simulation result into file **out.txt** in this example.

# 3 Simulation Commands

This chapter describes the syntax and examples for simulation commands that activate different functions of INDUCTWISE. Each simulation command can be anywhere in the input file and started with a dot(.). Most of the commands have parameters and options, which follow right after the commands in the same line. Two commands have to separate in two different lines.

## 3.1 .dc

The command performs DC analysis for the given circuit, and output voltages of all the nodes to the output file.

Syntax

```
.dc
```

No option and parameter are needed for this function. If no simulation command presents in the input file, the default function is DC analysis.

## 3.2 .tran

This command allows the simulator performing large-signal time-domain (transient) simulation of the given circuit. The result node voltages specified by **.print** command will be write to the output file.

Syntax

```
.tran STEP TOTAL [uic]
```

STEP: This parameter specifies the time step used to perform the transient analysis; its unit is second. INDUCTWISE adopts fixed time step to ensure the operation is as user specified since this simulator is for academic usage in this stage. Fixed time step is easier for users to compare the benefits of different algorithms.

TOTAL: This parameter specifies the total simulation time; its unit is second.

uic: This is an option. If this option presents the program will use zero as the initial condition of all the nodes. Otherwise, INDUCTWISE will solve the DC solution as in **.dc** commands first and the perform the transient analysis.

Example

```
.tran 0.5p 10p uic
```

This command performs the transient simulation lasting for 10 pico-second and solve Trapezoidal approximation every 0.5 pico-second. Since the uic option is activated, all the dc voltage sources will be treated as step functions instead of constant values.

## 3.3 .moment

This command allows the simulator performing moment calculation of the given circuit. The resulting node moments specified by **.print** command will be write to the output file.

Syntax

```
.moment ORDER
```

ORDER: This parameter specifies highest order of moment interested. The simulator will calculate -1 $\sim$ ORDER$^{th}$ moments.

Example

```
.moment 5
```

In this example, the output file looks like

```
 Order    M(1)          M(2)
-------  ----------     ----------
  -1    1.00000e+000    1.00000e+000
   0    0.00000e+000   -8.95000e-013
   1    0.00000e+000    5.78917e-024
   2    0.00000e+000   -3.32157e-035
   3    0.00000e+000    1.39184e-046
   4    0.00000e+000   -4.59104e-058
   5    0.00000e+000    1.25529e-069
```

## 3.4 .ieks

This command allows the simulator performing transient analysis by IEKS model reduction method. The result node voltages specified by **.print** command will be write to the output file. The syntax of **.ieks** is similar to **.tran** except one more parameter, ORDER, required.

Syntax

```
.tran STEP TOTAL ORDER [uic]
```

STEP: This parameter specifies the time step used to perform the transient analysis; its unit is second.

TOTAL: This parameter specifies the total simulation time; its unit is second.

ORDER: This parameter specifies the number of basis vectors used to reduce the system, which means the dimension of reduced system will be ORDER$\times$ORDER . Typically 10 $\sim$ 20 is enough to obtain excellent accuracy.

uic: This is an option. If this option presents the program will use zero as the initial condition of all the nodes. Otherwise, INDUCTWISE will solve the DC solution as in **.dc** commands first and the perform the transient analysis.

Example

```
.tran 0.5p 10p 10
```

## 3.5 .print

This command specifies the voltages of interested nodes in the circuit. The simulator will print the interested value into the output file. Some features of **.print** command are listed as follows.

1. For **.dc** analysis, the simulator automatically print voltages of all nodes. Any **.print** command is ignored in this mode.
2. For **.tran** and **.eks** modes, the voltages of nodes specified by **.print** command will be written for each time step.
3. For **.moment**, the desired orders of moment of nodes specified by **.print** command will be written into the output file.
4. Multiple **.print** commands are allowed in the same input file.

Syntax

```
.print v(NODENAME1) v(NODENAME2) ...
```

NODENAME: This argument is the name of an interested node in the circuit.

Example

```
.print v(n1355) v(out) v(port2)
```

## 3.6 .end

Once the parser reaches this commands, the rest part of the input file will not be parsed any more. This command is optional. If there is no **.end** command in the input file, the parser reads the whole file.

Syntax

```
.end
```

# 4 Device Statements

This section explains how to describe a circuit element in the INDUCTWISE input file. Most of them are the same as SPICE language. Two new elements are introduced; they are self and mutual reluctances, Y and W respectively. Each line describes one element, and they are identified by the first character. The key character is non-case sensitive.

## 4.1 Resistor (R)

This stands for a two-terminal resistor, which satisfies the following equation.

$$v = Ri$$

where $i$ is the branch current, $v$ is the branch voltage drop, and R is the value of the resistor.

Syntax

```
RNAME NODE1 NODE2 VALUE
```

NAME: The element name of the resistor.

NODE1: The positive terminal of the resistor.

NODE2: The negative terminal of the resistor.

VALUE: The value of the resistor; its unit is Ω.

Example

```
R1 n1355 n1356 0.1
```

## 4.2 Capacitor (C)

This stands for a two-terminal capacitor, which satisfies the following equation.

$$i = Cdv/dt$$

where $i$ is the branch current, $v$ is the branch voltage drop, and C is the value of the capacitor.

Syntax

```
CNAME NODE1 NODE2 VALUE
```

NAME: The element name of the capacitor.

NODE1: The positive terminal of the capacitor.

NODE2: The negative terminal of the capacitor.

VALUE: The value of the capacitor; its unit is F.

Example

```
C1 n1355 0 10pf
```

## 4.3 Self Inductor (L)

This stands for a two-terminal inductor, which satisfies the following equation.

$$v = Ldi/dt$$

where $i$ is the branch current, $v$ is the branch voltage drop, and L is the value of the inductor. Mutual inductors can be defined with **K** statement (will discuss later).

Syntax

```
LNAME NODE1 NODE2 VALUE
```

NAME: The element name of the inductor.

NODE1: The positive terminal of the inductor.

NODE2: The negative terminal of the inductor.

VALUE: The value of the inductor; its unit is Henry.

Example

```
L1 n1355 n1356 0.1nh
```

## 4.4 Mutual Inductor (K)

While more than one inductance present, the inductance equation can be written into matrix form:

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} L_{11} & M_{12} & \cdots & M_{1n} \\ M_{21} & L_{22} & \cdots & M_{2n} \\ \vdots & & & \vdots \\ M_{41} & M_{42} & \cdots & L_{nn} \end{bmatrix} \begin{bmatrix} \dot{i}_1 \\ \dot{i}_2 \\ \vdots \\ \dot{i}_n \end{bmatrix} \tag{1}$$

The diagonal terms of the inductance matrix are self inductances, which can be define as **L** statements. **K** statements define the off-diagonal terms that are mutual inductances.

Syntax

```
KNAME INDUCTOR1 INDUCTOR2 VALUE
```

NAME: The element name of the mutual inductor.

INDUCTOR1: The first coupled self inductor.

INDUCTOR2: The second coupled self inductor.

VALUE: The coupling coefficient (0<VALUE≤1). This value is not $M_{ij}$ itself, but satisfies the following equation.

$$M_{ij} = VALUE \times \sqrt{L_{ii}L_{jj}}$$

Example

```
L1 n1355 n1356 0.1nh
L2 n1357 n1358 0.2nh
K12 L1 L2 0.8
```

This example is equivalent to the following equation.

$$\begin{bmatrix} v_{n1355\to n1356} \\ v_{n1357\to n1358} \end{bmatrix} = \begin{bmatrix} 1\times10^{-10} & 1.6\times10^{-10} \\ 1.6\times10^{-10} & 2\times10^{-10} \end{bmatrix} \begin{bmatrix} \dot{i}_{L1} \\ \dot{i}_{L2} \end{bmatrix}$$

## 4.5 Self Reluctance (Y)

Inverting the inductance matrix, Equation (1) can be written as follows.

$$\begin{bmatrix} \dot{i}_1 \\ \dot{i}_2 \\ \vdots \\ \dot{i}_n \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & & & \vdots \\ K_{41} & K_{42} & \cdots & K_{nn} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{2}$$

The diagonal elements, self reluctance, can be defined by **Y**, and the off-diagonal elements, mutual reluctance, can be defined by **W**. The syntax is similar to **L** and **K**.

Syntax

```
YNAME NODE1 NODE2 VALUE
```

NAME: The element name of the reluctance element.

NODE1: The positive terminal of the reluctance element.

NODE2: The negative terminal of the reluctance element.

VALUE: The value of the reluctance element; its unit is $H^{-1}$.

## 4.6 Mutual Reluctance (W)

This represents the off-diagonal element of the inverse inductance matrix.

Syntax

```
WNAME RELUCTANCE1 RELUCTANCE2 VALUE
```

NAME: The element name of the mutual reluctance element.

RELUCTANCE1: The first coupled self reluctance element.

RELUCTANCE2: The second coupled self reluctance element.

VALUE: The coupling coefficient. Similar to **K**, the value satisfies the following equation. Notice that it is usually negative.

$$K_{ij} = VALUE \times \sqrt{K_{ii}K_{jj}}$$

Example

```
Y1 n1355 n1356 4.545e10
Y2 n1357 n1358 2.273e10
K12 Y1 Y2 -0.8
```

This example is equivalent to the example in Section 5.4.

## 4.7 Voltage Source (V)

A two terminal ideal voltage supply. Two different types of voltage sources, DC and piecewise linear, are available.

Syntax for DC Sources

```
VNAME NODE1 NODE2 DC_VALUE
```

Syntax for PWL Sources

```
VNAME NODE1 NODE2 PWL
+ (TIME1, VALUE1) (TIME2, VALUE2) ...
```

NAME: The element name of the voltage source.

NODE1: The positive terminal of the voltage source.

NODE2: The negative terminal of the voltage source.

DC_VALUE: The value of the DC voltage source; its unit is Volt.

TIME1, TIME2: Times at corner 1, 2, and so on; the unit is second.

VALUE1, VALUE2: Voltages at corner 1, 2, and so on; the unit is Volt.

Example

```
Vdd n1 0 1V
Vin n5 0 pwl
+ (0, 0) (1p, 0) (1.2p, 1) (2.8p, 1)
+ (3p, 0) (4p, 0)
```

### 4.8 Current Source (I)

A two terminal ideal current supply, whose usage is similar to voltage sources.

Syntax for DC Sources

```
INAME NODE1 NODE2 DC_VALUE
```

Syntax for PWL Sources

```
INAME NODE1 NODE2 PWL
+ (TIME1, VALUE1) (TIME2, VALUE2) ...
```

## References

[1] Yahong Cao, Yu-Min Lee, Tsung-Hao Chen, and Charlie Chung-Ping Chen. Hiprime: Hierarchical and passivity reserved interconnect macromodeling engine for rlkc power delivery. *DAC*, June 2002.

[2] Tsung-Hao Chen, Hyoung-Suk Kim, and Charlie Chung-Ping Chen. L extracted? now what? introduction to an inductance-wise interconnect simulation engine (inductwise). *Technical Report, University of Wisconsin-Madison*, April 2002.

[3] A. Devgan, H.Ji, and W. Dai. How to efficiently capture on-chip inductance effects: Introducing a new circuit element k. *ICCAD*, November 2000.

[4] Tanner EDA. *T-SPICE User Guide*.