Mixed Type Simulation Guide

Gong Ding
University of Science and Technology of China

Email: gdiso@ustc.edu

July 19, 2007

Contents

1 Introduction

Since version 0.45, GSS can do device/circuit mixed-type simulation with NGSPICE, gEDA's circuit simulator. A numerical device (NDEV) which is a TCP/IP client socket in fact, has been added to NGSPICE's device library[1][2]. It generally sends device terminal voltage informations to external device simulator (numerical or user defined analytic solver) and receives device's conductance matrix and equivalent current for each terminal calculated by external simulator. Thus, the NDEV interface can be considered as a general interface to user's own model.

At the same time, a special solver was designed for GSS to cooperate with NGSPICE. This solver was configured as a server socket for serving the NDEV connection. Some efforts were made at GSS end for meeting the demands of NGSPICE (please see the following text).

At present, one can do DC and TRANSIENT simulation with NGSPICE/GSS mixed-type simulator. The support of AC sweep will come soon. One should aware that the mixed-type simulation is more to academic purpose because the efficient is about 10^4 times lower than using pure SPICE compact models.

2 Equations for Circuit Simulation

Here we will introduce some principles of circuit simulation. The nonlinear system of equations for the circuit can be represented in the following equation by the nodal analysis method[3]:

$$\mathbf{I}(\mathbf{V}) = 0 \tag{1}$$

where $\mathbf{V}$ is the vector of node voltages and $\mathbf{I}$ is the sum of the currents into each node in the circuit. When the circuit has N modes, both $\mathbf{V}$ and $\mathbf{I}$ have N dimension.

Applying Newton-Raphson method to the above equation yields the linear matrices shown in Equation(2)

$$\mathbf{V}^{n+1} = \mathbf{V}^n - J^{-1}(\mathbf{V}^n)\,\mathbf{I}(\mathbf{V}^n) \tag{2}$$

The index n is the iteration count during NR iterations. And the J is the Jacobian matrix as show in the below.

$$J(\mathbf{V}) = \begin{bmatrix} \frac{\partial I_1}{\partial V_1} & \frac{\partial I_1}{\partial V_2} & \cdots & \frac{\partial I_1}{\partial V_N} \\ \cdots & & & \cdots \\ \frac{\partial I_N}{\partial V_1} & \frac{\partial I_N}{\partial V_2} & \cdots & \frac{\partial I_N}{\partial V_N} \end{bmatrix}$$

For each Newton iteration, the previous voltage $\mathbf{V}^n$ is known and hence $\mathbf{V}^{n+1}$ can be computed by Equation(2). Because each Jacobian element has units of the conductance, we hereafter use G to replace J. By multiplying G on both sides of Equation(2) from the left, one obtains the following set of linear equations at $n-th$ iteration.

$$G^n\mathbf{V}^{n+1} = G^n\mathbf{V}^n - \mathbf{I}^n = \mathbf{I}_{eqv} \tag{3}$$

Here $G = J^{-1}$ is the linear conductance components, including both the linear components in the original circuit and differential conductance at $\mathbf{V}^n$. And $\mathbf{I}_{eqv} = G^n\mathbf{V}^n - \mathbf{I}^n$ is the equivalent current source. The matrix G^n and vector $\mathbf{I}_{eqv}$ are both evaluated at $\mathbf{V}^n$. As a result, the above equation represents a linear circuit at bias point $\mathbf{V}^n$ because G^n and $\mathbf{I}_{eqv}$ are not depended on $\mathbf{V}^{n+1}$.

The NR iteration is terminated when the convergence is reached, i.e., the change in $\mathbf{V}$ between two consecutive iterations is smaller than a predefined tolerance. In SPICE, it is required that the current change in each circuit branch is also below certain criterion when the convergence is considered to be reached.

Transient analysis is performed in a similar manner. For each time step, Newton-Raphson iterations are performed until convergence is met. In addition, the truncation error due to the time discretization is checked by NGSPICE to determine if the time step is acceptable in terms of accuracy. If this error is too large, the time step is reduced automatically and the computation is repeated.

3 Build Conductance Matrix for Numerical Device

From above discussion, the key to add a device to NGSPICE is offering the conductance matrix G and equivalent current $\mathbf{I}_{eqv}$ for each terminal of device while terminal voltages $\mathbf{V}_n$ (and time step size, if NGSPICE is doing a transient simulation) are known. The SPICE compact models use analytic $\mathbf{I}(\mathbf{V})$ equations, thus they can get conductance matrix G by symbolic differentation. However, the numerical device simulator should use some more complicated method to get it[4][5].

The Level 1 drift-diffusion method (DDM) equations arising from device simulation are listed here:

$$\begin{cases} \dfrac{\partial n}{\partial t} = \nabla \cdot \left(\mu_n n \mathbf{E}_n + \mu_n \dfrac{k_b T}{q} \nabla n \right) - (U - G) \\ \dfrac{\partial p}{\partial t} = -\nabla \cdot \left(\mu_p p \mathbf{E}_p - \mu_p \dfrac{k_b T}{q} \nabla p \right) - (U - G) \\ \nabla \cdot \varepsilon \nabla \psi = -q(p - n + N_D - N_A) \end{cases} \tag{4}$$

with the boundary conditions i.e. ohmic electrode contact BC:

$$n = \frac{N_D - N_A + \sqrt{(N_D - N_A)^2 + 4n_{ie}^2}}{2} \tag{5}$$

$$p = \frac{N_A - N_D + \sqrt{(N_D - N_A)^2 + 4n_{ie}^2}}{2} \tag{6}$$

$$\psi = V_{app} + \frac{k_b T}{q} \operatorname{asinh}\left(\frac{N_D - N_A}{2n_{ie}} \right) \tag{7}$$

Here, n, p and n_{ie} are electron, hole and intrinsic carrier density. ψ is electrostatic potential. N_D and N_A are the doping concentration, respectively. $Vapp$ is the applied voltage to ohmic electrode contact.

The above equations can be rewritten as below:

$$\mathbf{F}(\mathbf{w}, \mathbf{V}) = 0 \tag{8}$$

Here, $\mathbf{w}$ stands for the basic semiconductor variables, i.e. electrostatic potential, electron density and hole density at each mesh point for drift-diffusion model and $\mathbf{V}$ is the external application voltage applied on each electrode of device. Because the basic semiconductor variables are dependent on electrode voltages, the $\mathbf{w}$ can be written as $\mathbf{w}(\mathbf{V})$. The semiconductor simulator solve the Equation(8) for an applied voltage $\mathbf{V_0}$ by Newton's method whereby

$$\Delta \mathbf{w} = -J_{\mathbf{w}}^{-1} \mathbf{F}(\mathbf{w}, \mathbf{V}_0) \tag{9}$$

is solved for each iteration. Here $J_{\mathbf{w}} = \dfrac{\partial \mathbf{F}}{\partial \mathbf{w}}$ is the Jacobian matrix for device equations(8).

The current flow in/out from the electrode can be calculated summing the current density around the electrode. The current density expressions for all the types of electrode supported by GSS (ohmic, schottky and gate electrode) have the same form as

$$I = I(\mathbf{w}) \tag{10}$$

which means electrode current I depends on $\mathbf{w}$, but not $\mathbf{V}$.

When the solution of semiconductor device equations is achieved, $\mathbf{F}(\mathbf{w}, \mathbf{V_0}) = 0$ holds and $I(\mathbf{w})$ can be calculated for each electrode. To calculate the element of linearized conductance matrix $G_{i,j} = \dfrac{\partial I_i}{\partial V_j}$, one can use chain difference rule which gives

$$G_{i,j} = \frac{\partial I_i}{\partial V_j} = \frac{\partial I_i}{\partial \mathbf{w}} \cdot \frac{\partial \mathbf{w}}{\partial V_j} \tag{11}$$

where $\dfrac{\partial I_i}{\partial \mathbf{w}}$ can be obtained by symbolic differentiation from $I(\mathbf{w})$. $\dfrac{\partial \mathbf{w}}{\partial V_j}$ is calculated by solving a linear system as follows. The partial decicative of Equation(8) with respect to V_j is

$$J_{\mathbf{w}} \frac{\partial \mathbf{w}}{\partial V_j} + \frac{\partial \mathbf{F}}{\partial V_j} = 0 \tag{12}$$

Since $\dfrac{\partial \mathbf{F}}{\partial V_j}$ can also be gotten from symbolic differentiation, one can solve $\dfrac{\partial \mathbf{w}}{\partial V_j}$ by Equation(12). This analytic method requires exact Jacobian matrix, or the final result may be uncertain. Fortunately, GSS meets the requirement. Because Equation(12) needs to be solved several times, the LU factor method is suitable here, which only needs forward and backward substitutions after first time fractionation.

4 Mixed-Mode Interface

NGSPICE and GSS works as two level Newton iterative schemes[6][7]. The outer is the circuit iteration which executed by NGSPICE to determine node voltages. For each outer iterations, terminal voltages of numerical device (and time step size, if transient simulation is desired) are sent to GSS. GSS solves semiconductor equations using Newton iteration method (the inner level) and sends the equivalent terminal currents and conductance matrix back to NGSPICE. Figure(1) shows how the two simulators communicate with each other. The two level structure has a nature parallel characteristic. For a complex circuit which has several numerical devices, NGSPICE can communicate with several GSS processes each handles one numerical device. The speed is only limited by the slowest GSS process.

At present, GSS offers two mixed type solvers for basic DD method and lattice temperature corrected DD method. Please refer to the user's guide of GSS.

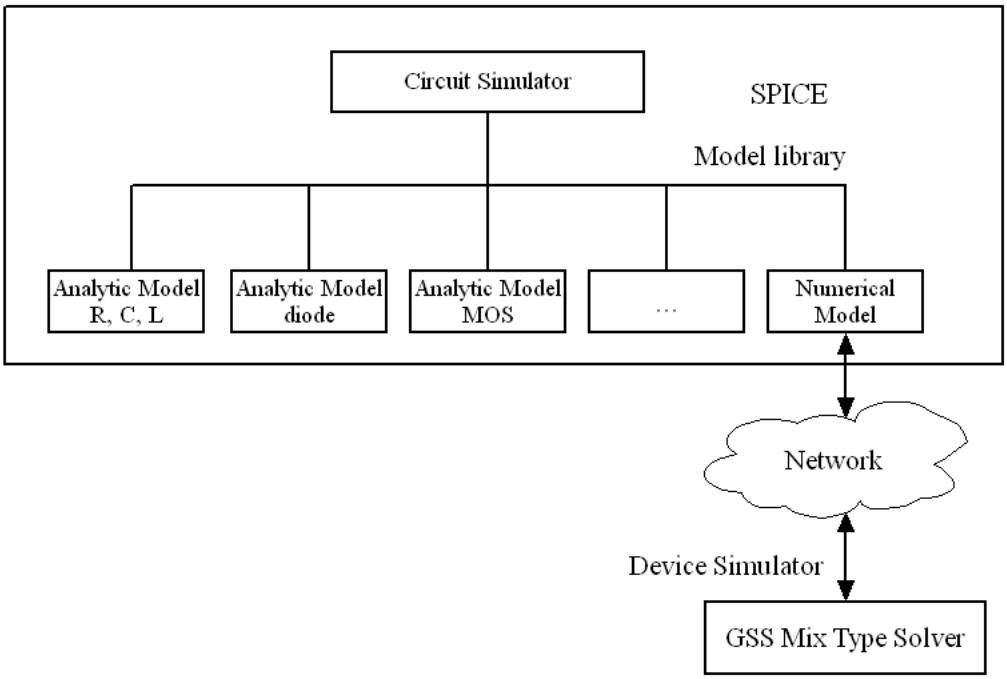

Figure 1: The structure of mixed solver

5 PN diode Examples

We will give a transient diode simulation example first to indicate the usage of mixed-type solver. The circuit for simulation is shown as follows

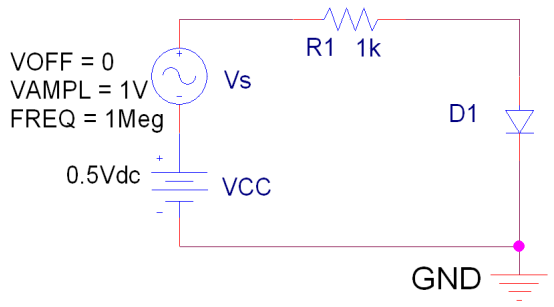

Figure 2: Circuit scheme for transient simulation of PN diode

Here is the input desk for NGSPICE. Line 6 specifies the numerical diode and line 9 is the model statement for the diode. The syntax of a general numerical device (NDEV) is fairly simple:

```
Nxxxxx    circuit_node=electrode_name ... model_name
.MODEL    model_name NDEV remote=<ipv4_address> port=<port_number>
```

The statement for NDEV component should begin with **Nxxxxx**, followed by circuit node label as general SPICE component except each node must have a corresponding electrode name for identify the electrode in GSS code. The number of electrode is limited to 7 at present and this number can be enlarged by modifying NGSPICE definition in the source code. The last element int the NDEV statement is always the model name. Each NDEV should have its own model when using GSS as external device solver, because GSS can only handle one instance for one model while SPICE analytic model supports multi-device instances. The **.MODEL** statement only contains IPv4 address and TCP port number for connecting to remote GSS routine. For this example, GSS will run at local computer and listen at port 17001.

```
1  Mixed Device/Circuit simulation of Diode
2
3  VDC 1 0 0.5V
4  Vs 2 1 SIN(0 1.0V 1MEGHz)
5  R1 2 3 1k
6  N1 3=Anode 4=Cathode M_N1
7  Vnn 4 0 0.0V
8
9  .MODEL M_N1 NDEV remote=localhost port=17001
10
11 .option acct itl2=100
12 .tran 0.001us 3us
13 .END
```

The diode comes from GSS's PN diode example. Here shows the mesh structure.

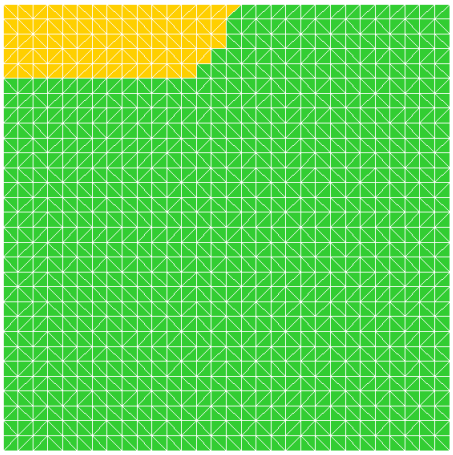

Figure 3: Initial Mesh

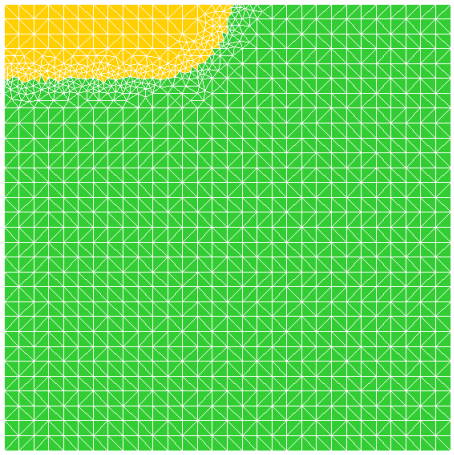

Figure 4: Refined Mesh

The mixed-type simulation input file for GSS shows as below. It tells GSS to import diode model from previous built file "pn.cgns". Since GSS is a two dimensional device simulator, one should give the width in the 3rd dimension. This variable can be set within GSS's input file by **Z.Width** parameter. GSS routine should be run before NGSPICE since it needs to initialize the device information. When everything is done, GSS will halt and listen at port 17001. From this port, NGSPICE will exchange data with GSS after it works.

```
1  #=============================================================================
2  # GSS example: PN Diode Mix Type Simulation
3  # GSS will provide numerical diode model to NGSPICE
4  #=============================================================================
5  set Carrier = pn # specify carrier type support p,n or pn
6  set Z.Width = 100 # device width in Z dimension. Unit: um
7  set LatticeTemp = 3e2 # specify initial temperature of device. Unit:K
8  set DopingScale = 1e16
9
10 #-----------------------------------------------------------------------------
11 # specify boundary condition.
12 boundary Type = OhmicContact ID = Anode Res=0 Cap=0 Ind=0
13 boundary Type = OhmicContact ID = Cathode Res=0 Cap=0 Ind=0
14
15 #-----------------------------------------------------------------------------
16 # Import CGNS file generated at first step
17 IMPORT CoreFile=pn.cgns
18
19 # GSS use port 17001 to exchange data with NGSPICE
20 METHOD Type=DDML1MIX Scheme=Newton NS=LineSearch LS=LU ServerPort=17001
```

```
21 # The solve process is controled by NGSPICE.
22 SOLVE
23 END
```

The transient results can be displayed by NGSPICE's **.PLOT** command. Figure(5) shows the diode current vs time when the frequency of sin voltage source is 1MHz. In this low-frequency situation, the diode acts as a good rectifier. If we increase the frequency (by modifying the NGSPICE input file), the result may be totally difference. Figure(6) shows the current vs time when the frequency up to 1GHz. And for this situation, the diode acts like a pure capacitance.

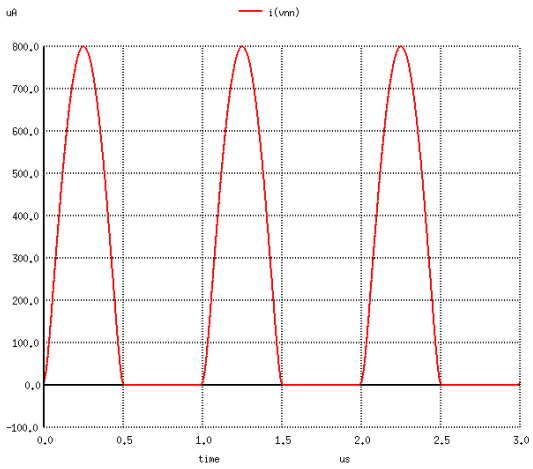

Figure 5: The diode current when f=1MHz

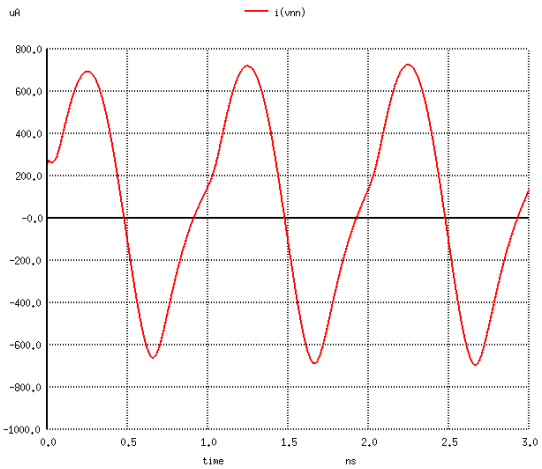

Figure 6: The diode current when f=1GHz

6 CMOS digital circuit Examples

We will demonstrate a more complicated CMOS invertor circuit examples in this section. The NMOS and PMOS transistors built by GSS are showed with Figure(7) and Figure(8). The width

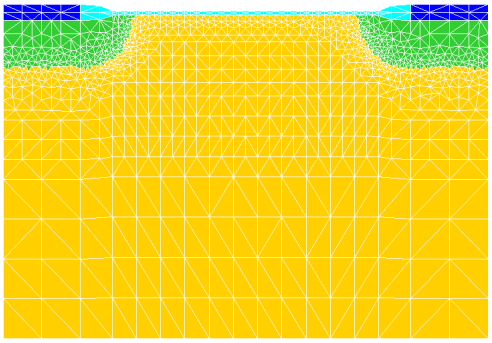

Figure 7: Mesh of NMOS Transistor

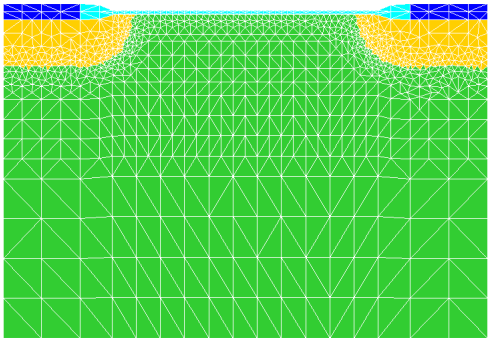

Figure 8: Mesh of PMOS Transistor

of NMOS in the Z-dimension is set to 2μm. And for load balance reason, the Z-width of PMOS is set as twice as NMOS.

The circuit scheme and the wafer layout of a typical CMOS invertor are show as Figure(9) and Figure(10).

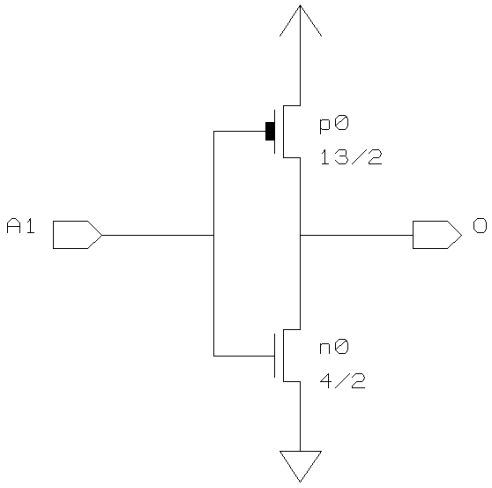

Figure 9: The scheme of CMOS invertor

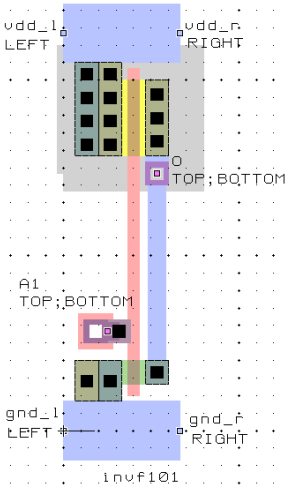

Figure 10: CMOS invertor layout

Figure(11) shows the scheme of simulation circuit. The 100Ohm resistors and 20fF capacitor represent parasitic interconnect resistance and capacitance. The power supply to all invertors is 5V.

The two transistors of the first invertor (U1A) are simulated by GSS while SPICE analytic models are performed to the second (load) invertor (U1B) for saving total simulation time. A pulsed voltage source will provide the input signal. The input file for NGSPICE is shown as

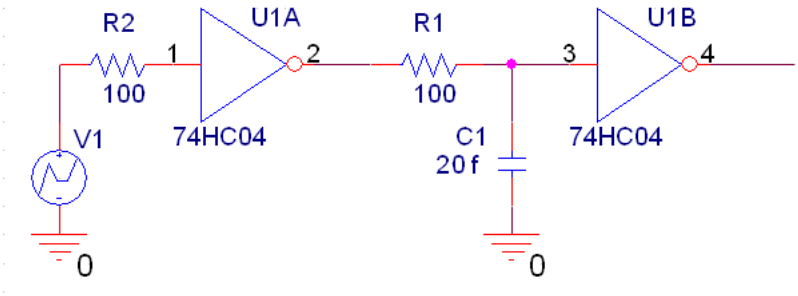

Figure 11: The circuit scheme of invertor pare

below:

```
CMOS Invertor mix-type transient simulation

VCC P_VCC 0 5V
VIN P_R2 0 0V (PULSE 0 5 0n 20p 20p 2n 4n)
R2 P_R2 N_R2 100

* numerical PMOS
NPMOS1 P_R1=PDrain N_R2=PGATE P_VCC=PSource P_VCC=PSubstrate PMOS
* numerical NMOS
NNMOS1 P_R1=NDrain N_R2=NGATE 0=NSource 0=NSubstrate NMOS
R1 P_R1 P_C1 100
C1 P_C1 0 20f

* P channel compact load transistor
MB1 VOUT P_C1 P_VCC P_VCC MPM PS=8u PD=8u AS=6p AD=6p W=3u L=1.25u
* N channel compact load transistor
MB2 VOUT P_C1 0 0 MNM PS=5u PD=5u AS=2p AD=2p W=1.5u L=1.25u

* Models to use for the compact MOS transistors
.MODEL MNM NMOS LEVEL=2 TOX=1.5e-7 NSUB=3E15 LD=.15u UO=600
+ VMAX=1E7 XJ=.3 JS=1E-15 VTO=.7
.MODEL MPM PMOS LEVEL=2 TOX=1.5e-7 NSUB=3E15 LD=.15u
+ XJ=.3 UO=300 JS=1E-15 VMAX=5E6 VTO=-.7

* Models for numerical MOS transistors
.MODEL NMOS NDEV remote=localhost port=17001
.MODEL PMOS NDEV remote=localhost port=17002

.OPTION acct itl2=100 icstep=2e12
.TRAN 1ps 9ns
```

```
.plot tran v(vout)
.END
```

Two GSS processes are invoked to provide terminal information about both NMOS and PMOS transistor. The input files are listed as follows.

```
#=============================================================================
# GSS mix-type simulation file for 1.5 Micron N-Channel MOSFET
#=============================================================================

set Carrier = pn
set Z.width = 2
set LatticeTemp = 3e2
set DopingScale = 1e18

#-----------------------------------------------------------------------------
# specify boundary condition.
boundary Type = InsulatorInterface ID=IF_NOxide_to_NSilicon QF=0
boundary Type = GateContact ID=NGate WorkFunction=4.17
boundary Type = OhmicContact ID=NSubstrate Res=0 Cap=0 Ind=0
CONTACT Type = OhmicContact ID=NSource Res=0 Cap=0 Ind=0
CONTACT Type = OhmicContact ID=NDrain Res=0 Cap=0 Ind=0

#-----------------------------------------------------------------------------
# specify physical model, use Lucent mobility here.
PMIS material=Si mobility=Lucent

#-----------------------------------------------------------------------------
# IMPORT NMOS MODEL
IMPORT CoreFile=nmos_init.cgns
METHOD Type = DDML1MIX Scheme = Newton NS=Basic LS=GMRES ServerPort=17001
SOLVE

END
```

```
#=============================================================================
# GSS mix-type simulation file for 1.5 Micron P-Channel MOSFET
#=============================================================================

set Carrier = pn
set Z.width = 4
set LatticeTemp = 3e2
set DopingScale = 1e18

#-----------------------------------------------------------------------------
```

```
11 # specify boundary condition.
12 boundary Type = InsulatorInterface ID=IF_POxide_to_PSilicon QF=0
13 boundary Type = GateContact ID=PGate WorkFunction=5.25
14 boundary Type = OhmicContact ID=PSubstrate Res=0 Cap=0 Ind=0
15 CONTACT Type = OhmicContact ID=PSource Res=0 Cap=0 Ind=0
16 CONTACT Type = OhmicContact ID=PDrain Res=0 Cap=0 Ind=0
17
18 #-------------------------------------------------------------------------------
19 # specify physical model, use Lucent mobility here.
20 PMIS material=Si mobility=Lucent
21
22 #-------------------------------------------------------------------------------
23 # IMPORT PMOS MODEL
24 IMPORT CoreFile=pmos_init.cgns
25 METHOD Type = DDML1MIX Scheme = Newton NS=Basic LS=GMRES ServerPort=17002
26 SOLVE
27
28 END
```

We use a workstation with dual Xeon 3.6GHz CPU to simulate the circuit. The total simulation time is more than 1 hour. During the simulation, the CPU time for NGSPICE is neglectable while each GSS process takes 90% of single CPU resource. It means the computation is well parallelized.

Figure(12) shows the in and out voltage waveform of two cascade invertors. The delay time is approximate 0.2ns. Figure(13) shows the current pass though the first invertor. This current only exists when the invertor changing its state.

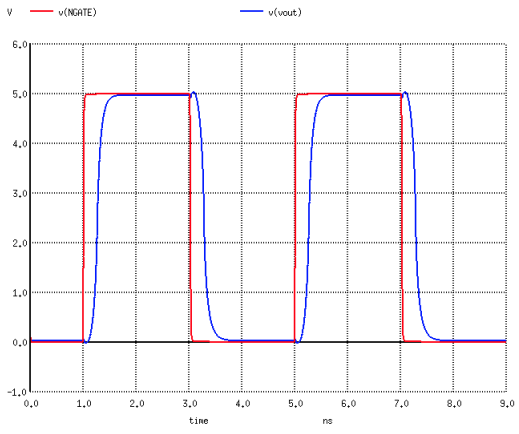

Figure 12: In/Out waveform of invertor pare

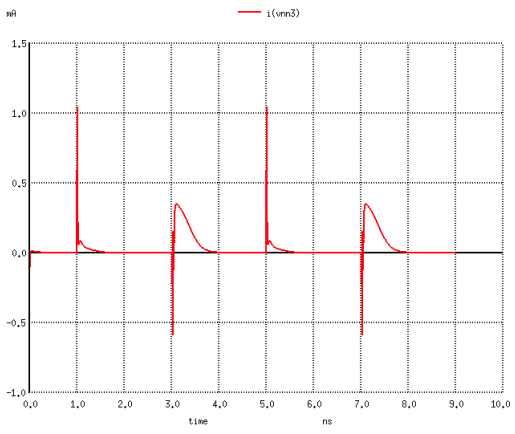

Figure 13: Current of invertor U1A

References

[1] Thomas L. Quarles. *Adding Devices to SPICE3*. Memorandum No. UCB/ERL M89/45. Berkeley: University of California, 1989. 2

[2] Thomas L. Quarles. *SPICE3 Implementation Guide*. Memorandum No. UCB/ERL M89/44, Berkeley: University of California, 1989. 2

[3] William J. McCalla. *Fundamentals of Computer-Aided Circuit Simulation*. Boston: Kluwer Academic Publishers, 1993. 3

[4] Kartikeya Mayaram. *CODECS: A Mixed-Level Circuit and Device Simulator*. Memorandum No. UCB/ERL M88/71, Berkeley: University of California, 1988. 4

[5] David Alan Gates. *Design-Oriented Mixed-level Circuit and Device Simulation*. Memorandum No. UCBERL M93/51, Berkeley: University of California, 1993. 4

[6] Mayaram Kartikeya and Donald O. Pederson. Coupling algorithms for mixed-level circuit and device simulation. *IEEE Trans. Computer-Aided Design*, II(8):1003–1012, 1992. 6

[7] Yu Zhiping, Robert W. Dutton, and Hui Wang, editors. *A Modularized, Mixed IC Device/ Circuit Simulation System*, Japan, April 1992. Proceedings of the Synthesis and Simulation Meeting and International Exchange. 6