GSS Version 0.37
User's Guide

Gong Ding
University of Science and Technology of China
Email: gdiso@ustc.edu

Sep 5, 2005

Contents

Acknowledgements

This paper is dedicated to my girl friend daisysjp.

I should like to thank my supervisor Jianguo Wang for patient guide and project support.

I also would like to give thanks to my fellows: Fen Han, Gang Wang, Hongfu Xia and Yue Wang for the endless discussions on CFD, FDTD, girl, money and the meaning of life.

Chapter 1

Physical Description

1.1 The governing equation

1.1.1 The Hydrodynamic model

The semiclassical Boltzmann equation coupled with the Poisson equation provides a general theoretical framework for modeling electron transport in semiconductor. Both Hydrodynamic equations and Drift-Diffusion can be derived from the moments of this equation.

The Boltzmann transport equation for electrons moving with the group velocity $\mathbf{u}$ and effective electron mass m^* in an electric field $\mathbf{E}$ can be represented as

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \bigtriangledown_x f - \frac{e}{m^*}\mathbf{E} \cdot \bigtriangledown_u f = \left(\frac{\partial f}{\partial t}\right)_{coll} \tag{1.1}$$

While Monte Carlo simulations provide a direct numerical solution to this equation, costly computations make their practical usage limited. The finite difference method can also be used to solve this equation, but only for academic aim.

Assuming parabolic energy bands, the first five moments in the velocity space are the balance equations for the flux of electron, momentum, and energy. These equations are represented as follows:

$$\frac{\partial n}{\partial t} + \nabla \cdot (n\mathbf{u}) = \left(\frac{\partial n}{\partial t}\right)_{coll} \tag{1.2}$$

$$\frac{\partial m_n^* n\mathbf{u}}{\partial t} + \nabla \cdot (m_n^* n\mathbf{u}\mathbf{u} + nkT_n) = -en\mathbf{E} + \left(\frac{\partial m_n^* n\mathbf{u}}{\partial t}\right)_{coll} \tag{1.3}$$

$$\frac{\partial n\omega_n}{\partial t} + \nabla \cdot (n\omega_n\mathbf{u} + nkT_n\mathbf{u}) = -en\mathbf{E} \cdot \mathbf{u} - \nabla(-\kappa_n \nabla T_n) + \left(\frac{\partial n\omega_n}{\partial t}\right)_{coll} \tag{1.4}$$

Here, n is the electron concentration, $\mathbf{u}$ is the translational velocity, T_n is the temperature of electron, $\omega_n = \frac{2}{3}kT_n + \frac{1}{2}m_n^* u^2$ is the internal energy, and κ_n is the rate of heat transferring.

Equations 1.2, 1.3 and 1.4 are known as the Hydrodynamic Model in semiconductor simulation. With the same process, the hydrodynamic equations of hole can also be achieved.

$$\frac{\partial p}{\partial t} + \nabla \cdot (p\mathbf{v}) = \left(\frac{\partial p}{\partial t}\right)_{coll} \tag{1.5}$$

$$\frac{\partial m_p^* p\mathbf{v}}{\partial t} + \nabla \cdot (m_p^* p\mathbf{v}\mathbf{v} + pkT_p) = ep\mathbf{E} + \left(\frac{\partial m_p^* p\mathbf{v}}{\partial t}\right)_{coll} \tag{1.6}$$

$$\frac{\partial p\omega_p}{\partial t} + \nabla \cdot (p\omega_p\mathbf{v} + pkT_p\mathbf{v}) = ep\mathbf{E}\cdot\mathbf{v} - \nabla(-\kappa_p \nabla T_p) + \left(\frac{\partial p\omega_p}{\partial t}\right)_{coll} \tag{1.7}$$

These equations are supplemented by the Poisson equation for the electric potential ϕ

$$\nabla \cdot (\epsilon \nabla \phi) = -e(p - n + N_d - N_a) \tag{1.8}$$

The collision term in Eq 1.2 and 1.5 can be replaced with carrier recombination and generation rate.

$$\left(\frac{\partial n}{\partial t}\right)_{coll} = \left(\frac{\partial p}{\partial t}\right)_{coll} = G - R \tag{1.9}$$

Follow the Baccarani and Wordeman model, the collision term in momentum and energy equations are simplified by relaxation times, respectively.

$$\left(\frac{\partial m_n^* n\mathbf{u}}{\partial t}\right)_{coll} = -\frac{m_n^* n\mathbf{u}}{\tau_p^n} \qquad \tau_p^n = m_n^* \frac{\mu_{n0}}{e}\frac{T_n}{T_0} \tag{1.10}$$

$$\left(\frac{\partial m_p^* p\mathbf{v}}{\partial t}\right)_{coll} = -\frac{m_p^* p\mathbf{v}}{\tau_p^p} \qquad \tau_p^p = m_p^* \frac{\mu_{p0}}{e}\frac{T_p}{T_0} \tag{1.11}$$

$$\left(\frac{\partial n\omega_n}{\partial t}\right)_{coll} = -\frac{n\omega_n - \frac{3}{2}nkT_0}{\tau_\omega^n} \qquad \tau_\omega^n = \frac{m_n^*}{2}\frac{\mu_{n0}}{e}\frac{T_0}{T_n} + \frac{3}{2}\frac{\mu_{n0}}{ev_{ns}^2}\frac{T_nT_0}{T_n+T_0} \tag{1.12}$$

$$\left(\frac{\partial p\omega_p}{\partial t}\right)_{coll} = -\frac{p\omega_p - \frac{3}{2}pkT_0}{\tau_\omega^p} \qquad \tau_\omega^p = \frac{m_p^*}{2}\frac{\mu_{p0}}{e}\frac{T_0}{T_p} + \frac{3}{2}\frac{\mu_{p0}}{ev_{ps}^2}\frac{T_pT_0}{T_p+T_0} \tag{1.13}$$

Where T_0 is the ambient temperature of device, μ_{n0} and μ_{p0} are the low field mobility of electron and hole, v_{ns} and v_{ps} are the saturation velocity. This model includes carrier-phonon and carrier-impurity collisions.

1.1.2 Reduce HDM to DDM

A simplification of the HDM will lead to the DDM. We take equations of electron for example. The energy balance equation is completely removed from the set of equations; therefore, it is no longer possible to include the electron temperature T_e

in the current equation. T_e is simply replaced by the lattice temperature T_L. We assume that convective term $\nabla \cdot (m_n^* n\mathbf{uu})$ and the time derivative of the electron current $\frac{\partial m^* nu}{\partial t}$ are small compared to the other terms. Neglecting the time derivative of the current density is equivalent to the assumption that the electron momentum is able to adjust itself to a change in the electric field within a very short time. This leads to the current equation:

$$\mathbf{J}_n = -en\mathbf{u} = \frac{e^2}{m_n^*}\tau_p^n n\mathbf{E} + \frac{e}{m_n^*}kT_L\tau_p^n\nabla n \tag{1.14}$$

By the same procedure, we can get the current equation for hole.

$$\mathbf{J}_p = \frac{e^2}{m_p^*}\tau_p^p p\mathbf{E} - \frac{e}{m_p^*}kT_L\tau_p^p\nabla p \tag{1.15}$$

For getting a more general form, we first define the carrier mobility rate

$$\mu_n = \tau_p^n \frac{e}{m_n^*} \tag{1.16}$$

$$\mu_p = \tau_p^p \frac{e}{m_p^*} \tag{1.17}$$

then we can rewrite the current equations as follows.

$$\mathbf{J}_n = e\mu_n n\mathbf{E} + e\mu_n(\frac{kT_L}{e})\nabla n \tag{1.18}$$

$$\mathbf{J}_p = e\mu_p p\mathbf{E} - e\mu_p(\frac{kT_L}{e})\nabla p \tag{1.19}$$

Continuity equation 1.2, 1.5 and Poisson equation 1.8 are of course still valid in the DDM.

1.2 Recombination and Generation Rate

GSS supports Shockley-Read-Hall, Auger, and direct recombination (also known as band-to-band or optical recombination).

$$R = R_{SRH} + R_{Auger} + R_{dir} \tag{1.20}$$

$$R_{SRH} = \frac{pn - n_{ie}^2}{\tau_p[n + n_{ie}\exp(\frac{\mathbf{ETRAP}}{kT_L})] + \tau_n[p + n_{ie}\exp(\frac{-\mathbf{ETRAP}}{kT_L})]} \tag{1.21}$$

$$R_{Auger} = \mathbf{AUGN}(pn^2 - nn_{ie}^2) + \mathbf{AUGP}(np^2 - pn_{ie}^2) \tag{1.22}$$

$$R_{dir} = \mathbf{DIRECT}(np - n_{ie}^2) \tag{1.23}$$

In the above, n_{ie} is the effective intrinsic concentration and τ_n and τ_p are the concentration dependent electron and hole lifetimes.

$$\tau_n = \frac{\textbf{TAUN0}}{1 + N_{total}/\textbf{NSRHN}} \tag{1.24}$$

$$\tau_p = \frac{\textbf{TAUP0}}{1 + N_{total}/\textbf{NSRHP}} \tag{1.25}$$

The value of parameters are listed below.

	Unit	Silicon	GaAs	Ge
ETRAP	eV	0	0	0
DIRECT	cm^3s^{-1}	1.1e-14	7.2e-10	6.41e-14
AUGN	cm^6s^{-1}	1.1e-30	1e-30	1e-30
AUGP	cm^6s^{-1}	0.3e-30	1e-29	1e-30
TAUN0	s	1e-7	5e-9	1e-7
TAUP0	s	1e-7	3e-6	1e-7
NSRHN	cm^{-3}	5e16	5e17	5e16
NSRHP	cm^{-3}	5e16	5e17	5e16

Currently, GSS dose not support impact ionization. This feature may be implemented in next edition.

1.3 Mobility Model

GSS uses Analytic Mobility as its low field mobility model. These are given by

$$\mu = \mu_{min} + \frac{\mu_{max}(\frac{T_L}{300})^{\alpha} - \mu_{min}}{1 + (\frac{T_L}{300})^{\beta}(\frac{N_{total}}{N_{ref}})^{\gamma}} \tag{1.26}$$

	Unit	Silicon:n	Silicon:p	GaAs:n	GaAs:p
μ_{min}	$cm^2 \cdot (V \cdot s)^{-1}$	55.24	49.70	0.0	0.0
μ_{max}	$cm^2 \cdot (V \cdot s)^{-1}$	1429.23	479.37	8500.0	400.0
α	$-$	-2.3	-2.2	-1.0	-2.1
β	$-$	-3.8	-3.7	0.0	0.0
γ	$-$	0.73	0.70	0.436	0.395
N_{ref}	cm^{-3}	1.072e17	1.606e17	1.69e17	2.75e17

The high field mobility models are not supported by current version of GSS yet.

Chapter 2

Numerical Method

2.1 The Numerical Solution of DDM equations

Parabolical System

Finite Volume Method

The Scharfetter-Gummel Discretization

The solution of Nonlinear Systems of Algebraic Equations

Newton Iterative method

Nonlinear Solvers in PETSC

2.2 The Numerical Solution of HDM equations

Hyperbolical System

Finite Volume Method

The Roe and AUSM flux

The Operator-split method

The dual time marching method

The solution of linear Algebraic Equations

KSP Solvers in PETSC

2.3 Boundary Condition

Ohmic Contract

Schottky Contact

Insulator Contact

Insulator Interface

Neumann Boundary

Chapter 3

Mesh Generation

3.1 The Format of Mesh File

GSS uses CGNS(CFD General Notation System) as standard input/output file. This file format provides the ability to store grid, solution data, material information, boundary condition and connectivity in a single, well-defined and easy-to-use form. More important, CGNS has been accepted and supported by most of the commotional CFD corporations. Actually, it has become the industrial standard among CFD society.

A CGNS file is an entity that is organized (inside the file itself) into a set of "nodes" in a tree-like structure, in much the same way as directories are organized in the UNIX environment. The top-most node is referred to as the "root node". Each node below the root node is defined by both a name and a label, and may or may not contain information or data. The utility ADFviewer was created to allow users to easily view CGNS files.

The source code of CGNS can be downloaded from *http://sourceforge.net/projects/cgns*. Also, two free software ADFviewer and CGNSplot which are very useful for showing the mesh and debugging are available there. The detailed document of CGNS can be found in NASA at *http:// www.grc.nasa.gov/www/cgns/*.

There are two ways to generate initial file in which contains grid, region information, boundary and doping profile. One is to use SGframework and another is converting Medici's output file TIF to CGNS. We will explain later.

3.2 Mesh Generation by SGFramework

The most convenient way for mesh generation is employing SGFramework. The original edition of SGFramework developed by Kevin M. Kramer dose not support CGNS. The edition which I did some improvement can be found under \preprocess.

For installing SGFramework

1 just untar the package;

2 do necessarily changes in Makefile.common;
the default installation path is /usr/local/SGframework
3 type 'make config' to generate script;
4 type 'make' to compile the source code;
5 if everything is ok, type 'make install' to install the software.

For more detailed information please refer the book *Semiconductor Devices A Simulation Approach* written by Kevin M.Kramer and W. Nicholas G. Hitchon.

Here, I will take some examples to show how to generate the CGNS file. These examples are stored under /SGframework/examples.

3.2.1 Mesh of PN diode

The mesh description file pn.sk is listed below. This example is under /SGframework/examples/PN

```
const Wdiode   = 9.0e-4;
const Ddiode   = 9.0e-4;
const Wanode   = 2.0e-4;
const Wcathode = 2.0e-4;
const Wspacing = 0.3e-4;

//
//       Anode                                 Cathode
//      A-----B-----------------------------C---------D
//      |                                             |
//      |                                             |
//      |                                             |
//      |                                             |
//      |                                             |
//      |                                             |
//      |                                             |
//      E-----F-----------------------------G---------H

point pA = (0.0e-4,           0.0e-4);
point pB = (Wanode,           0.0e-4);
point pC = (Wdiode-Wcathode,  0.0e-4);
point pD = (Wdiode,           0.0e-4);
point pE = (0.0e-4,          -Ddiode);
point pF = (Wanode,          -Ddiode);
point pG = (Wdiode-Wcathode, -Ddiode);
point pH = (Wdiode,          -Ddiode);


edge eBC = WALL    [pB, pC] (Wspacing, 0.0);
edge eEF = WALL    [pE, pF] (Wspacing, 0.0);
edge eFG = WALL    [pF, pG] (Wspacing, 0.0);
edge eGH = WALL    [pG, pH] (Wspacing, 0.0);
edge eAE = WALL    [pA, pE] (Wspacing, 0.0);
edge eBF =         [pB, pF] (Wspacing, 0.0);
edge eCG =         [pC, pG] (Wspacing, 0.0);
edge eDH = WALL    [pD, pH] (Wspacing, 0.0);
edge eAB = Anode   [pA, pB] (Wspacing, 0.0);
edge eCD = Cathode [pC, pD] (Wspacing, 0.0);

region rAEFB = Si {eAE, eEF, eBF, eAB};
region rBFGC = Si {eBF, eFG, eCG, eBC};
region rCGHD = Si {eCG, eGH, eDH, eCD};

const Na  = 1.00e+19;
const Nd  = 1.00e+15;
const Rx  = 2.00e-04;
const Ry  = 2.50e-04;
const Ax  = ln(Na/Nd)/sq(Rx);
const Ay  = ln(Na/Nd)/sq(Ry);

coordinates x, y;

refine C (SignedLog, 1.0) = Nd-(Na+Nd)*ngdep(x,y,2.0*Wanode,Ax,Ay);

set minimum divisions = 0;
set maximum divisions = 1;
```

To generate mesh, a UNIX script is offered in file 'run'. It contains

```
#do syntax check
mesh pn.sk
#build initial grid
sggrid pn.xsk
#build code for mesh refinement
sgbuild ref pn_ref
#do mesh refinement
#argument -c   generate CGNS file
#argument -ps  generate PostScript file
pn_ref -c -ps
```

Because the syntax of mesh description can be found in the User's Guide of SGframework, I will only mention some notice for compatible with GSS.

```
 1. GSS always considers the unit of length as centimeter. Please
    follow this rule all the time.
 2. The edge label will be converted to boundary label. So don't give
    label to any inner edge (eBF,eCG), or GSS will be confused.
 3. If two different boundary edge share the same point (eAE,eAB),
```

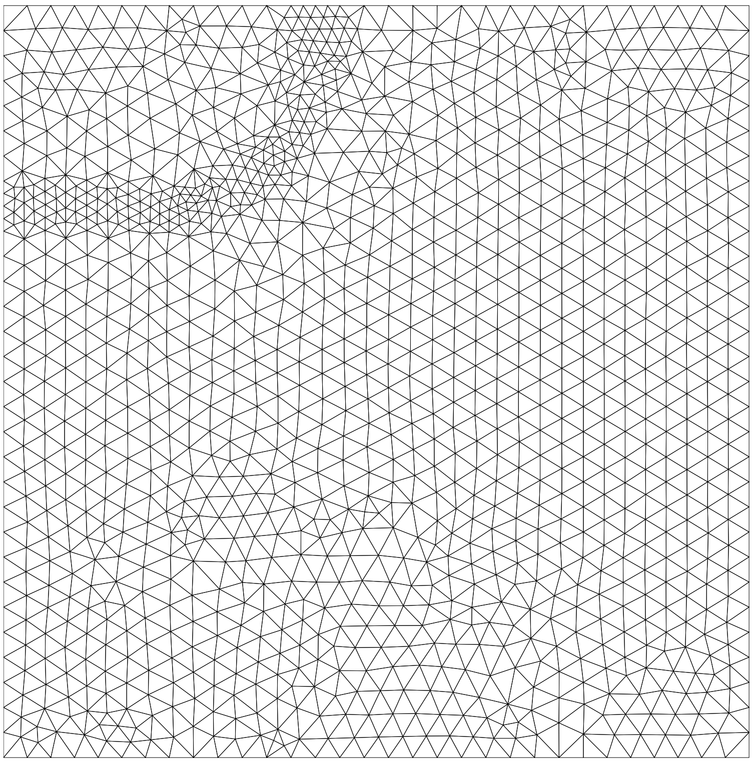

Figure 3.1. The mesh of PN diode

```
   the later will overwrite the former. Put eAB after eAE when you
   think point A should belongs to Anode.
4. The region label denotes the material type of this region. This is
   very important to GSS. For silicon bulk, this label can only be 'Si'
   or 'Silicon'. Other acceptable labels are 'GaAs' for Gallium Arsenide,
   'Ge' for Germanium and 'SiO2' or 'Ox' for Oxide.
5. Positive value of refine function will be considered as Nd, negative
   value will be converted to Na.
```

The mesh is shown on figure 3.1.

3.2.2 Mesh of NMOS

Another complex example about NMOSFET is listed here. This example can be found under /SGframework/examples/NMOS

```
// mesh constants
```

```
const WDEV  = 2.0e-4;     // device width           | (cm) | segment 1
const DDEV  = 2.0e-4;     // device depth           | (cm) | segment 2
const WOX   = 1.4e-4;     // oxide width            | (cm) | segment 3
const DOX   = 0.2e-6;     // oxide depth            | (cm) | segment 4
const WCONT = 0.2e-4;     // contact width          | (cm) | segment 5
const DRECT = 0.2e-4;     // depth of channel       | (cm) | segment 6

//       |--5--|   |------------3------------|   |--5--|
//
//                 A-------------------------B              -
//                 |\\\\\\\\\\\\\\\\\\\\\\\\\|              4
//   -   C-----D---E-------------------------F---G-----H    -
//   6   |         |                         |         |    |
//   -   |         I-------------------------J         |    |
//       |                                             |    |
//       |                                             |    2
//       |                                             |    |
//       |                                             |    |
//       |                                             |    |
//       K---------------------------------------------L    -
//
//       |----------------------1----------------------|

// define points
point pA = ((WDEV-WOX)/2, DOX), pG = (WDEV-WCONT, 0.0);
point pB = ((WDEV+WOX)/2, DOX), pH = (WDEV, 0.0);
point pC = (0.0, 0.0),          pI = ((WDEV-WOX)/2, -DRECT);
point pD = (WCONT, 0.0),        pJ = ((WDEV+WOX)/2, -DRECT);
point pE = ((WDEV-WOX)/2, 0.0), pK = (0.0, -DDEV);
point pF = ((WDEV+WOX)/2, 0.0), pL = (WDEV, -DDEV);

// define edges
edge eDE = WALL     [pE, pD] (WOX/50, 0.2);
edge eFG = WALL     [pF, pG] (WOX/50, 0.2);
edge eIJ =          [pI, pJ] (WOX/40, 0.0);
edge eAE = WALL     [pE, pA] (1.0e-7, 0.5);
edge eBF = WALL     [pF, pB] (1.0e-7, 0.5);
edge eCK = WALL     [pC, pK] (WCONT/15, 0.2);
edge eEI =          [pE, pI] (WOX/50, 0.1);
edge eHL = WALL     [pH, pL] (WCONT/15, 0.2);
edge eFJ =          [pF, pJ] (WOX/50, 0.1);
edge eKL = SUB      [pK, pL] (WDEV/8, 0.0);
edge eEF = ISGATE   [pE, pF] (WOX/50, 0.0);
edge eCD = DRAIN    [pC, pD] (WCONT/8, 0.0);
edge eGH = SOURCE   [pG, pH] (WCONT/8, 0.0);
edge eAB = GATE     [pA, pB] (WOX/50, 0.0);

//define regions
```

```
region r1 = SiO2 {eAE, eEF, eBF, eAB} RECTANGLES;
region r2 = Si   {eEI, eIJ, eFJ, eEF} ;
region r3 = Si   {eCK, eKL, eHL, eGH, eFG, eFJ, eIJ, eEI, eDE, eCD};

// define coordinate labels
coordinates x, y;

// physical constants and properties of Si and SiO2
const T     = 300.0;              // operating temperature
const e     = 1.602e-19;          // electron charge            (C)
const kb    = 1.381e-23;          // Boltzmann's constant      (J/K)
const e0    = 8.854e-14;          // permittivity of vacuum   (F/cm)
const eSi   = 11.8;               // dielectric constant of Si
const eSiO2 = 3.9;                // dielectric constant of SiO2

// doping constants
const NS = 1.0e16;                // substrate doping        (cm^-3)
const NC = 1.0e19;                // contact doping          (cm^-3)
const WDIFF = (WDEV-WOX)/2;       // diffusion width            (cm)
const DDIFF = 0.25e-4;            // diffusion depth            (cm)
const DT    = 3.0e-12;            // diffusion coef. * time   (cm^2)

// doping profile
refine C (SignedLog, 3.0) = (y <= 0.0) * { -NS          +
  (NC+NS) * nsdep(x,     2*WDIFF,DT) * nsdep(y,2*DDIFF,DT) +
  (NC+NS) * nsdep(WDEV-x,2*WDIFF,DT) * nsdep(y,2*DDIFF,DT) };

// set min/max edge spacing and min/max refinement levels
// the +1.0 in the next line is important to avoid divided by a very small #
set minimum length = sqrt(e0*eSi*(kb*T/e)/e/abs(C+1.0));
set maximum length = 1.0;
set minimum divisions = 0;
set maximum divisions = 3;
```

In this example, oxide layer is in an septated region which has a label 'SiO2' and it is divided into rectangles. Edge eEF is the isolator interface between Si bulk and SiO2. Again, for define point C as Drain, the definition of edge eCD is after eCK.

3.3 Mesh Generation by Medici

Avant! Medici© is a powerful two dimensional semiconductor simulator. Unfortunately it is a commotional software and in high price. Our institute spent about 50 000 US dollars on it.

Medici uses Technology Interchange Format (TIF) for its data file. I developed

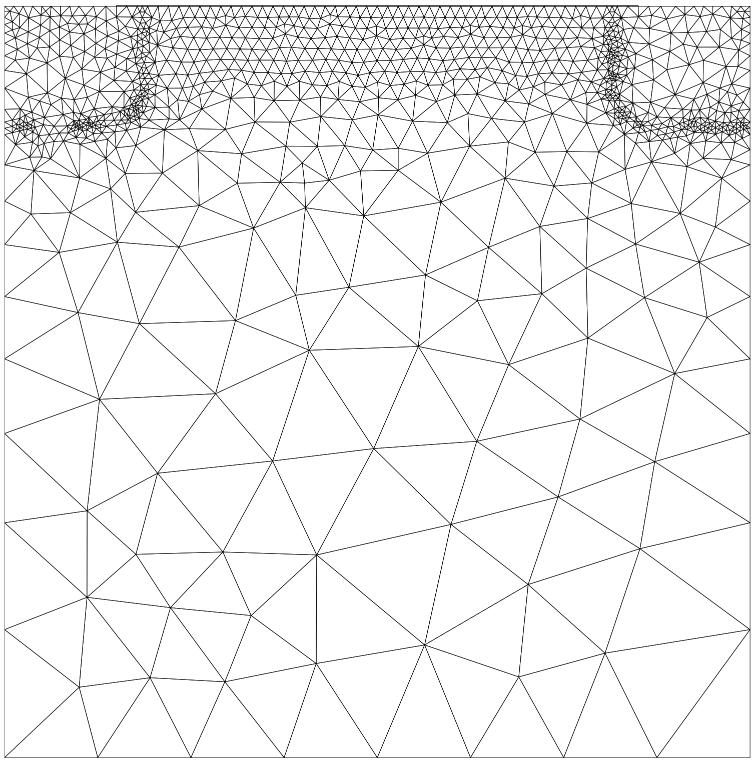

Figure 3.2. The mesh of NMOS

a small command line tool 'dumptif' for converting it to CGNS. The tool, examples and the introduction can be found under /preprocess/TIF.tar.gz.

Chapter 4

Input Statement Descriptions

4.1 Introduction

GSS is directed via input statements. These statements are stored in an input file. Each input statement must be written in the same line. The max characters in one line and the max lines in the input file don't have any limits essentially.

Each statement has the syntax as *keyword parameter=*[*string*|*number*]. Keywords and parameters are reserved words which user should not use them. String should begin with character or underline. Character, digital, underline and dot are allowed in string and the length of string is limited to 30 characters. Some of the strings like file name or boundary identifier can be specified by user; but others like solver type are fixed. The number expression supports C syntax double precision float point number.

4.2 Input example

```
# common line
# All the command has the syntax: keyword  parameter=string|double
# String must begin with character or underline.
# Character, digital, underline and dot are allowed in string.
# The length of string is limited to 30 characters.
# Numerical value double support c syntax float point number.
# The Unit in command file.:
# Time Unit:ps  voltage Unit:V  Freq Unit:THz Length Unit:cm
#=======================================================================
# static command, in lower case
set MeshFile    = mesh.cgns # specify cgns file which contains mesh,doping and boundary lable
set ModelFile   = model.dat # specify physical model
set Carrier     = pn        # specify carrier type support p,n or pn
set DeviceDepth = 0.01      # device depth in Z dimension. Unit:cm
set LatticeTemp = 3e2       # specify initial temperature of device. Unit:K
#-----------------------------------------------------------------------
# voltage source.
# ID can be specified by user, limited to 32 characters, c syntax.
# reference: voltage source in spice model
vsource Type = VDC    ID = GND  Tdelay=0 Vconst=0
vsource Type = VDC    ID = VCC  Tdelay=0 Vconst=5
vsource Type = VSIN   ID = Vs   Tdelay=1 Vamp=0.1 Freq=1e-6
vsource Type = VEXP   ID = V1   Tdelay=0 TRC=1 TFD=3 TFC=1 Vlo=0 Vhi=1
vsource Type = VPULSE ID = V2   Tdelay=0 Tr=1 Tf=1 Pw=5 Pr=10 Vlo=0 Vhi=1
#-----------------------------------------------------------------------
# specify boundary condition.
# ID must accord with the boundary name in cgns file
# electrode boundary
#   OhmicContract|SchottkyContract|GateContract|InsulatorContract
# they have parameters of parasite res,cap and ind  unit:Om,F,H
# SchottkyContract has an extra parameter,the barrier height.
# GateContract can specify the workfunc of gate electrode.
# InsulatorContract offers a simple way for describing Si/SiO2 interface,
#   the Thick of oxide must be specified,
# NeumannBoundary has heat transfer rate parameter Kapa. unit ?
# InsulatorInterface is the interface of Si/SiO2, has a fixed charge density QF,unit ?

boundary Type = InsulatorContract  ID = SiSiO2    Res=0 Cap=0 Ind=0 Thick=1e-6 QF=0
boundary Type = InsulatorInterface ID = IFACE     QF=0
boundary Type = GateContract       ID = GATE      Res=0 Cap=0 Ind=0 WorkFunction=0
boundary Type = NeumannBoundary    ID = WALL      Kapa=0
boundary Type = SchottkyContract   ID = sgate     Res=0 Cap=0 Ind=0 Vbarrier=-0.8
boundary Type = OhmicContract      ID = OMANODE   Res=0 Cap=0 Ind=0
boundary Type = OhmicContract      ID = OMCATHODE Res=0 Cap=0 Ind=0
boundary Type = OhmicContract      ID = OMSOURCE  Res=0 Cap=0 Ind=0
boundary Type = OhmicContract      ID = OMDRAIN   Res=0 Cap=0 Ind=0
boundary Type = OhmicContract      ID = OMSUB     Res=0 Cap=0 Ind=0
#FloatMetalGate   may support later

#============================================================================
# drive command, specify the solving process. # keyword is in upper care
# reference: medici user's guide
#--------------------------------------------
# METHOD Type = DDM scheme = [Newton|Gummel] &
#        SNES = [LineSearchCubic|LineSearchQuadratic|LineSearchNo|TrustRegion]  &
#        TStep = time_number
# METHOD Type = HDM scheme = [Implicit|Explicit] FluxFunc = [AUSM|Roe]  &
#        Reconstruct = [FirstOrder|SecondOrder] CFL = cfl_number
#--------------------------------------------
# ATTACH Electrode = electrode_name  VApp = vsource_name1 VApp = vsource_name2 ...
#--------------------------------------------
# SOLVE Type = EQUILIBRIUM
# SOLVE Type = STEADYSTATE
# SOLVE Type = DCSWEEP  VScan = electrode_name  IVRecord = electrode_name  &
#       IVFile = file_name VStart = v_number VStep = v_number VStop = v_number
# SOLVE Type = TRANSIENT   AUTOSAVE = time_number   IVRecord = electrode_name &
#       IVFile = file_name TStart = time_number    TStop = time_number
#---------------------------------------------
# MODELS not supported yet
#---------------------------------------------
# IMPORT CoreFile = file_name
#---------------------------------------------
# EXTRACT CoreFile = file_name    AscFile = file_name
#---------------------------------------------
# REFINE Variable = [Doping|Potential] Measure = [Linear|SignedLog] Dispersion = number
#---------------------------------------------
# PLOT   Variable = Mesh Resolution=[Low|Middle|High] PSFile=file_name
# PLOT   Variable =[Na|Nd|ElecDensity|HoleDensity|Potential|EFieldX|EFieldy|Temperature] &
#        Resolution=[Low|Middle|High] PSFile=file_name Measure=[Linear|SignedLog]        &
#        AzAngle=angle_number ElAngle=angle_number Style=[Scale|Color|GrayLevel]
#============================================================================
METHOD   Type = DDM   Scheme = Newton   TStep=1e3         #DDM method is the default solver
PLOT     Variable=Mesh
REFINE   Variable=Doping Measure=SignedLog Dispersion=1   #refine by doping
SOLVE    Type=EQUILIBRIUM                                 #compute equilibrium state
REFINE   Variable=Potential Measure=Linear Dispersion=0.1 #refine by potential
PLOT     Variable=Mesh
SOLVE    Type=EQUILIBRIUM                                 #compute equilibrium state again
PLOT     Variable=Na Resolution=Middle    AzAngle=240  ElAngle=40  Style=Scale
PLOT     Variable=Nd Resolution=Middle    AzAngle=240  ElAngle=40  Style=Scale
PLOT     Variable=ElecDensity PSFile=electron Resolution=High    AzAngle=240  ElAngle=40  Style=Colo
PLOT     Variable=HoleDensity PSFile=hole     Resolution=High    AzAngle=240  ElAngle=40  Style=Colo
PLOT     Variable=Potential   Resolution=High    AzAngle=240  ElAngle=40  Style=GrayLevel
# extract mesh and solution
EXTRACT  CoreFile=init.cgns

IMPORT   CoreFile=init.cgns                    # import result
ATTACH   Electrode=OMCATHODE  VApp=GND         # attach vsource to boundary(electrode)
# DC sweep
SOLVE    Type=DCSWEEP  VScan=OMANODE  IVRecord=OMANODE  IVFile=iv.txt VStart=0 VStep=1e-2 VStop=1

#IMPORT CoreFile=break.cgns
ATTACH   Electrode=OMANODE    VApp=VCC         VApp=Vs
# specify HDM method
METHOD   Type=HDM             Scheme=Explicit  FluxFunc=AUSM  Reconstruct=FirstOrder CFL=0.1
SOLVE    Type=TRANSIENT       AUTOSAVE=1   TStart=0    TStop=10

METHOD   Type=HDM             Scheme=Implicit  FluxFunc=AUSM  Reconstruct=SecondOrder CFL=2
SOLVE    Type=TRANSIENT       AUTOSAVE=1   TStart=10    TStop=20
```