

## Hierarchy of simulation models:

1. **Introduction to modeling**
2. **Drift-Diffusion model**
  - Derivation of the current continuity equation
  - Derivation of the electron current equation
  - Validity of the drift-diffusion model
  - Physical limitations of the drift-diffusion model
  - Choice of variables in the drift-diffusion model
  - Scharfetter-Gummel discretization scheme
  - Boundary conditions
  - Generation and recombination model
  - Mobility models
  - Gummel's and Newton's schemes
  - existing drift-diffusion simulators
3. **Hydrodynamic model**
  - derivation of the basic hydrodynamic equations
  - Ensemble relaxation rates
  - Discretization of the balance equations
4. **Introduction to the Silvaco ATLAS tool**
  - Some general comments
  - DeckBuild overview
  - ATLAS syntax
  - Examples

## 1. Introduction to Modeling

### Modeling:

Representation of the physical structure or behavior of a device (or devices) by an abstract mathematical model which approximates this behavior. Such a model may either be a closed form expression (analytical model), or a system of simultaneous equations which are solved numerically.

**Device Modeling** - Modeling of the physical behavior of a semiconductor device. The term is often used in practice to mean the representation of a device in terms of a lumped parameter model used in higher level circuit simulation of complex integrated circuits. In the broader sense it includes both physical simulation and more abstract mathematical representations.

**Device Simulation** - Simulation of the device behavior by the approximate numerical solution of the (approximate) physical transport and field equations governing charge flow in the device, usually represented in a finite space (device domain).

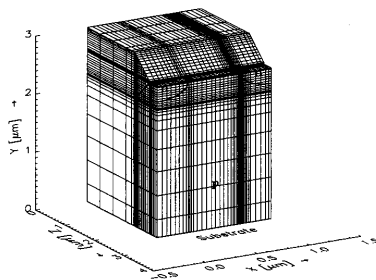
## Goals of Device Modeling

**Analysis** - Simulating the behavior of a device (or circuits) with a physical model to understand the dependencies and limiting physical mechanisms in the device/circuit performance (e.g. effects of noise, limits on frequency/gain, trap effects, effects of geometry).

**Design** - Systematic use of a device/circuit model to achieve a desired functionality. For device design, and low level circuit design, the process is mainly an iterative, trial and error approach prior to actual physical implementation of a device or a circuit.

## Physical Device Simulation

- There are 2 main components in physical device simulation:
  - (1) Charge motion due to driving forces and diffusion (transport)
  - (2) Fields due to charge distribution and motion (i.e current)
- Analytical solutions are only possible in 1D. Numerical solutions require discretization of (1) and (2) above onto a *mesh*, and solution of simultaneous algebraic equations
- (1) and (2) must be solved simultaneously (self-consistently)



Recessed MOSFET represented on 3D mesh over finite domain

### Solution to the Field Equations

- In general, one needs to solve Maxwell's equations inside and outside the device:

$$\begin{aligned} \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} & \nabla \cdot \mathbf{D} &= \rho \\ \nabla \times \mathbf{H} &= \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} & \nabla \cdot \mathbf{B} &= 0 \end{aligned}$$

- Numerical techniques to solve include:

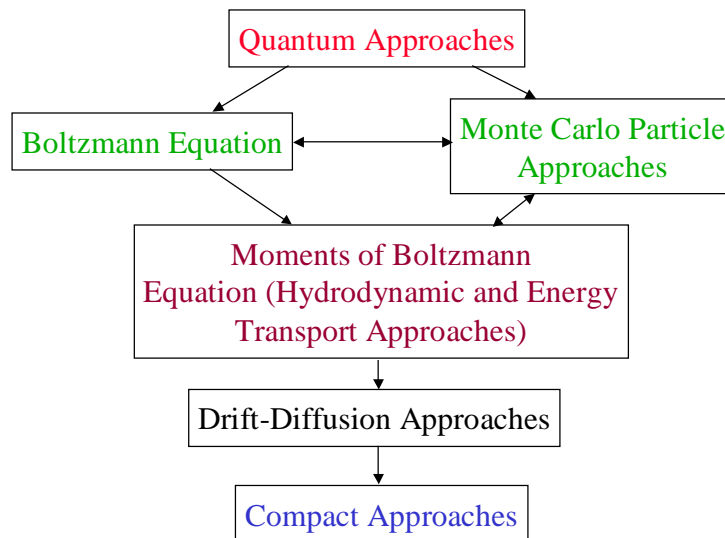
Time domain solutions (FDTD)

Frequency domain solutions (spectral techniques)

- At present, nearly all device simulation tools assume the quasi-static approximation, such that the electric field is obtained from Poisson's equation:

$$\nabla^2 V(\mathbf{r}) = -\frac{\rho(\mathbf{r})}{\epsilon} \dots \dots \mathbf{E} = -\nabla V$$

### Hierarchy of Semiconductor Simulation Models



## 2. Drift-Diffusion Simulator

- 2.1 Derivation of the current continuity equations
- 2.2 Derivation of the electron current equation
- 2.3 Validity of the Drift-Diffusion model
- 2.4 Physical limitations of the Drift-Diffusion model
- 2.5 Choice of variables in the Drift-Diffusion scheme
- 2.6 Sharfetter-Gummel discretization scheme for the continuity equation
- 2.7 Boundary Conditions
- 2.8 Generation and Recombination models
- 2.9 Mobility models
- 2.10 Gummel's and Newton's solution scheme
- 2.11 Commercially available simulation tools

### 2.1 Derivation of the Current Continuity Equations

- Start from Maxwell's equations given in the previous slides:

$$\begin{cases} \nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}, & \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \\ \nabla \cdot \mathbf{D} = \rho, & \nabla \cdot \mathbf{B} = 0 \end{cases}$$

- Applying the divergence operator to the first equation leads to:

$$\nabla \cdot \mathbf{J} + \frac{\partial \rho}{\partial t} = 0$$

- Use:  $\mathbf{J} = \mathbf{J}_n + \mathbf{J}_p$  and  $\rho = e(p - n + N_D^+ - N_A^-)$

- Arrive at the following final results: 
$$\begin{cases} \frac{\partial n}{\partial t} = \frac{1}{e} \nabla \cdot \mathbf{J}_n + U \\ \frac{\partial p}{\partial t} = -\frac{1}{e} \nabla \cdot \mathbf{J}_p - U \end{cases}$$

## 2.2 Derivation of the Electron Current Equation

- Start from the steady-state Boltzmann transport equation (for 1D case) in the relaxation-time approximation

$$v \frac{\partial f}{\partial x} - \frac{eE}{m^*} \frac{\partial f}{\partial v} = -\frac{f - f_0}{\tau}, \quad f_0(x, v) = n(x) \left( \frac{2\pi k_B T}{m^*} \right)^{-1/2} \exp\left(-\frac{m^* v^2}{2k_B T}\right)$$

- Multiply by the velocity  $v$  and integrate over  $v$ , to get:

$$\int v^2 \frac{\partial f}{\partial x} dv - \frac{eE}{m^*} \int v \frac{\partial f}{\partial v} dv = -\frac{1}{\tau} \int v f dv$$

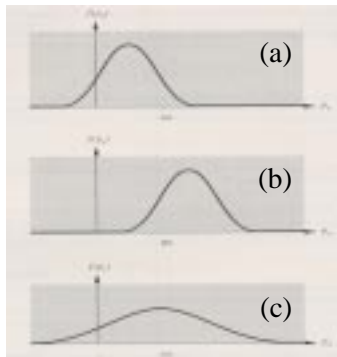
$$\frac{\partial}{\partial x} [n(x) \langle v^2 \rangle] - \frac{eE}{m^*} n(x) = \frac{1}{e\tau} J_n(x)$$

- Final expression:

$$J_n(x) = en(x)\mu_n E + e\tau \langle v^2 \rangle \frac{\partial n(x)}{\partial x} + e\tau n(x) \frac{\partial \langle v^2 \rangle}{\partial x}$$

- Neglect the drift energy and use the Einstein relation for the consideration of the diffusion coefficient and mobility, to get:

$$J_n(x) = en(x)\mu_n E + eD_n \frac{\partial n}{\partial x} + e \underbrace{\frac{D_n n(x)}{T_e}}_{\text{Soret Coefficient}} \frac{\partial T_e}{\partial x}$$



- ← (a) Low-field distribution function
- (b) High-field distribution function when the kinetic energy gained appears mostly as drift energy
- (c) High-field distribution function for conditions under which the kinetic energy gained from the field appears mostly as thermal energy

## 2.3 Validity of the Drift-Diffusion Model

### (a) Approximations made in its derivation

- Temporal variations occur in a time-scale much longer than the momentum relaxation time.
- The drift component of the kinetic energy was neglected, thus removing all thermal effects.
- Thermoelectric effects associated with the temperature gradients in the device are neglected, i.e.

$$\mathbf{J}_n(x) = en(x)\mu_n\mathbf{E} + eD_n\nabla n$$

$$\mathbf{J}_p(x) = ep(x)\mu_p\mathbf{E} - eD_p\nabla p$$

- The spatial variation of the external forces is neglected, which implies slowly varying fields.
- Parabolic energy band model was assumed, i.e. degenerate materials can not be treated properly.

### (b) Extension of the capabilities of the DD model

- Introduce field-dependent mobility  $\mu(\mathbf{E})$  and diffusion coefficient  $D(\mathbf{E})$  to empirically extend the range of validity of the DD Model.
- An extension to the model, to take into account the overshoot effect,\* has been accomplished in 1D by adding an extra term that depends on the spatial derivative of the electric field

$$J_n(x) = en(x) \left[ \mu_n(E) + \mu_n(E)L(E) \frac{\partial E}{\partial x} \right] + eD_n(E) \frac{\partial n}{\partial x}$$

1. K.K. Thornber, IEEE Electron Device Lett., Vol. 3 p. 69, 1982.
2. E.C. Kan, U. Ravaioli, and T. Kerkhoven, Solid-State Electron., Vol. 34, 995 (1991).

## 2.4 Physical Limitations of the DD Model (discretization)

- The complete DD Model is summarized below:

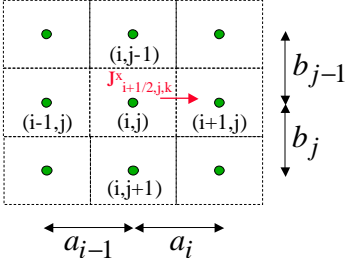
$$\begin{aligned} \text{Current equations: } \mathbf{J}_n &= en\mu_n\mathbf{E} + eD_n\nabla n \\ \mathbf{J}_p &= ep\mu_p\mathbf{E} - eD_p\nabla p \end{aligned}$$

$$\begin{aligned} \text{Continuity equations: } \frac{\partial n}{\partial t} &= \frac{1}{e}\nabla \cdot \mathbf{J}_n + U \\ \frac{\partial p}{\partial t} &= -\frac{1}{e}\nabla \cdot \mathbf{J}_p - U \end{aligned}$$

$$\text{Poisson Equation: } \nabla^2 V = \frac{e}{\epsilon} (n - p + N_D^+ - N_A^-)$$

- A numerical scheme that solves the continuity equation should:
  - Conserve the number of particles in the device,
  - Respect local positivity of the carrier density, and
  - Respect monotonicity of the solution.

- Conservative schemes for the continuity equation are achieved with the following discretization scheme:



$$\begin{aligned} n_{i,j}^{k+1} - n_{i,j}^k &= U_{i,j}^k + \frac{1}{e} \frac{J_{i+1/2,j}^k - J_{i-1/2,j}^k}{0.5(a_{i-1} + a_i)} \\ &+ \frac{1}{e} \frac{J_{i,j+1/2}^k - J_{i,j-1/2}^k}{0.5(b_{j-1} + b_j)} \end{aligned}$$

- Requirements:
  - the mesh size must be smaller than the Debye length  $L_D$
  - Time step must be smaller than the dielectric relaxation time:  $t_{dr} = \epsilon / (eN\mu)$

## 2.5 Choice of Variables in the Drift-Diffusion Model

- Natural variable formulation:  $(V, n, p)$
- Quasi-Fermi level formulation:  $(V, \phi_n, \phi_p)$

$$n = n_i \exp\left[\frac{e(V - \phi_n)}{k_B T}\right], \quad p = n_i \exp\left[\frac{e(\phi_p - V)}{k_B T}\right]$$

- Slotboom variables:  $(V, \Phi_n, \Phi_p)$

$$\Phi_n = n_i \exp\left(-\frac{e\phi_n}{k_B T}\right) \rightarrow n = \Phi_n \exp\left(\frac{eV}{k_B T}\right)$$

$$\Phi_p = n_i \exp\left(\frac{e\phi_p}{k_B T}\right) \rightarrow p = \Phi_p \exp\left(-\frac{eV}{k_B T}\right)$$

- Standard way of scaling due to de Mari:

Space:	Intrinsic $L_D$ ( $N=n_i$ ) Extrinsic $L_D$ ( $N=N_{max}$ )	$L_D = \sqrt{\epsilon k_B T / (e^2 N)}$
Potential:	Thermal voltage	$V_T = k_B T / e$
Carrier density:	Intrinsic density Extrinsic density	$N=n_i$ $N=N_{max}$
Diffusion Coeff:	$D_0$	$1 \text{ cm}^2 / \text{s}$
Mobility	$\mu$	$\mu = D_0 / V_T$
Recomb./Gen.	$R$	$R = D_0 N / L_D^2$
Time:	$T$	$T = L_D^2 / D_0$

## 2.6 Sharfetter-Gummel discretization scheme

- The discretization of the continuity equation in conservative form requires the knowledge of the current densities

$$J_n(x) = en(x)\mu_n E + eD_n \nabla n$$

$$J_p(x) = ep(x)\mu_p E - eD_p \nabla p$$

on the mid-points of the mesh lines connecting neighboring grid nodes. Since solutions are available only on the grid nodes, interpolation schemes are needed to determine the solutions.

- There are two schemes that one can use:
  - Linearized scheme:  $V$ ,  $n$ ,  $p$ ,  $\mu$  and  $D$  vary linearly between neighboring mesh points
  - Scharfetter-Gummel scheme: electron and hole densities follow exponential variation between mesh points

### (a) Linearized scheme

- Within the linearized scheme, one has that

$$J_{i+1/2} = -en_{i+1/2}\mu_{i+1/2} \frac{V_{i+1}-V_i}{a_i} + eD_{i+1/2} \nabla n|_{i+1/2}$$

$$\begin{array}{ccc} \downarrow & & \downarrow \\ \frac{n_{i+1} + n_i}{2} & & \frac{n_{i+1} - n_i}{a_i} \end{array}$$

$$J_{i+1/2} = n_{i+1} \left[ -\frac{e\mu_{i+1/2}}{2} \frac{V_{i+1}-V_i}{a_i} + \frac{eD_{i+1/2}}{a_i} \right]$$

$$- n_i \left[ \frac{e\mu_{i+1/2}}{2} \frac{V_{i+1}-V_i}{a_i} + \frac{eD_{i+1/2}}{a_i} \right]$$

- This scheme can lead to substantial errors in regions of high electric fields and highly doped devices.

### **(b) Scharfetter-Gummel Scheme**

- One solves the electron current density equation

$$J_{i+1/2} = -en\mu_{i+1/2} \frac{V_{i+1} - V_i}{a_i} + eD_{i+1/2} \frac{\partial n}{\partial x}$$
$$= -en\mu_{i+1/2} \frac{V_{i+1} - V_i}{a_i} + eD_{i+1/2} \frac{\partial n}{\partial V} \frac{\partial V}{\partial x}$$

for  $n(V)$ , subject to the boundary conditions

$$n(V_i) = n_i \quad \text{and} \quad n(V_{i+1}) = n_{i+1}$$

- The solution of this first-order differential equation leads to

$$n(V) = n_i [1 - g(V)] + n_{i+1} g(V), \quad g(V) = \frac{e^{(V-V_i)/Vt} - 1}{e^{(V_{i+1}-V_i)/Vt} - 1}$$

$$J_{i+1/2} = \frac{eD_{i+1/2}}{a_i} \left[ n_{i+1} B\left(\frac{V_{i+1} - V_i}{Vt}\right) - n_i B\left(\frac{V_i - V_{i+1}}{Vt}\right) \right]$$

$$B(x) = \frac{x}{e^x - 1} \quad \text{is the Bernoulli function}$$

### **2.7 Boundary Conditions**

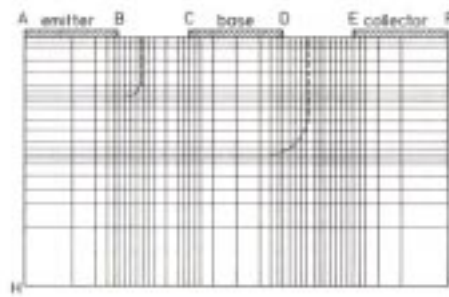
From the aspect of device physics, one can distinguish between the following device boundaries:

- (1) Contacts, which allow a current flow in and out of the device
  - Ohmic contacts: purely voltage or purely current controlled
  - Schottky contacts
- (2) Contacts where only voltages can be applied
- (3) Interfaces, where current flow disappears
- (4) Artificial boundaries, where neither electric field nor current flow exists

Examples of different boundary conditions:



**MOSFET**



**Lateral BJT**

**(A) Boundary conditions for Ohmic contacts**

**Electrostatic potential:**

- Voltage-controlled ohmic contact (Dirichlet boundary conditions):

$$V(t) = V_b + V_{applied}$$

**n-type sc:**  $N_D \gg N_A \rightarrow V_b = V_t \ln(N_D / n_i)$

**p-type sc:**  $N_A \gg N_D \rightarrow V_b = -V_t \ln(N_A / n_i)$

- Current-controlled contact (integral boundary condition):

$$\int_{D_0} (\mathbf{J}_n + \mathbf{J}_p) \cdot d\mathbf{A} - I(t) = 0, \quad V(t) = V_b + const.$$

### Electron and hole densities:

- The electron and hole densities are determined by assuming charge neutrality and thermal equilibrium

$$n - p + N_A^- - N_D^+ = 0, \quad np = n_i^2$$

$$n\text{-type sc: } n = \frac{1}{2} \left[ \sqrt{N_D^2 + 4n_i^2} + N_D \right], \quad p = n_i^2 / n$$

$$p\text{-type sc: } p = \frac{1}{2} \left[ \sqrt{N_A^2 + 4n_i^2} + N_A \right], \quad n = n_i^2 / p$$

- Low temperatures:

$$N_D^+ = \frac{N_D}{1 + 2 \exp\left(\frac{E_D - E_F}{k_B T}\right)}, \quad N_A^- = \frac{N_A}{1 + 4 \exp\left(\frac{E_A - E_F}{k_B T}\right)}$$

Donor and acceptor energy levels for common semiconductors in Si [eV]



Arsenic	Phosphorus	Antimony	Boron
0.054	0.045	0.039	0.045

### (B) Boundary conditions for the Schottky contacts

#### Electrostatic potential:

Dirichlet boundary condition:  $V(t) = V_{schottky} + V_{applied}$

Material:	Aluminum	Platinum	Titanium
$V_{schottky}$ [V]	0.68	0.8	0.6
$V_{n,p}$ [cm/s]	$5 \times 10^6$	$5 \times 10^6$	$5 \times 10^6$

#### Current:

Neumann boundary conditions (thermionic-emission and diffusion theory):

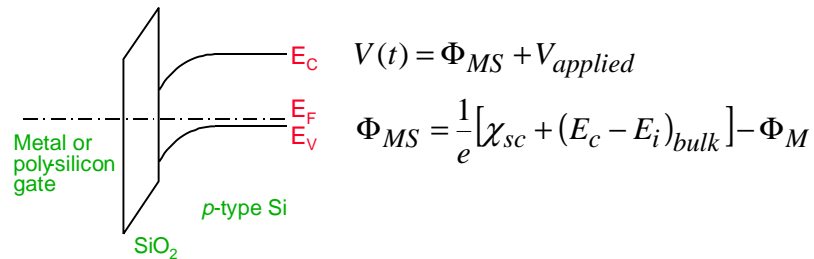
$$\mathbf{J}_n \cdot \vec{\mathbf{n}} = -ev_n(n - n_0)$$

$$\mathbf{J}_p \cdot \vec{\mathbf{n}} = ev_p(p - p_0)$$

- R. Hattori, J. Shirafuji, Jpn. J. Appl. Phys., Vol. 33, pp. 612-618, 1994.
- J.R. Tucker, C. Wang, and P.S. Carney, Appl. Phys. Lett., Vol. 65, pp. 618-620, 1994.

### (C) Gate contact (only voltage can be applied)

- Dirichlet boundary condition for the potential:



- Values of the work-function for different gate materials:

Material:	$n^+$ -poly	$p^+$ -poly	Aluminum
$F_{MS}$ [V]	0.55	-0.50	0.50

### (D) Semiconductor/Oxide Interfaces

#### Electrostatic potential:

- The normal components of the dielectric displacement vector must satisfy the Gauss law in its differential form:

$$\epsilon_{sc} \frac{\partial V}{\partial \mathbf{n}} \Big|_{sc} - \epsilon_{ox} \frac{\partial V}{\partial \mathbf{n}} \Big|_{ox} = Q_{int}$$

- For MOSFETs, simplified boundary condition would be:

$$\epsilon_{sc} \frac{\partial V}{\partial \mathbf{n}} \Big|_{sc} - \epsilon_{ox} \frac{V_G - V}{t_{ox}} = Q_{int}$$

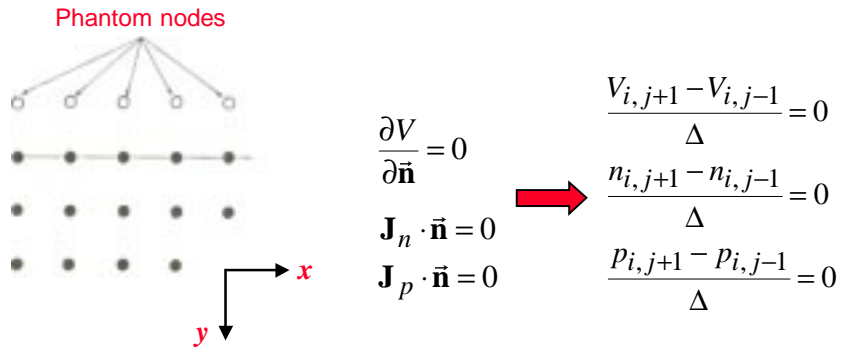
#### Current:

- Neumann boundary conditions:

$$\begin{aligned} \mathbf{J}_n \cdot \mathbf{\bar{n}} &= -eR^{surf} & \mathbf{J}_n \cdot \mathbf{\bar{n}} &= 0 \\ \mathbf{J}_p \cdot \mathbf{\bar{n}} &= eR^{surf} & \mathbf{J}_p \cdot \mathbf{\bar{n}} &= 0 \end{aligned} \quad \rightarrow$$

### (E) Artificial boundaries

This type of boundary is not based on physical consideration. Therefore, it is called an artificial boundary. One applies Neumann boundary conditions for both the electrostatic potential and current, i.e.



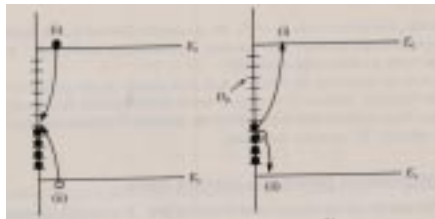
### 2.8 Generation and Recombination Model

- The **Schockley-Reed-Hall** model for the generation-recombination term due to trap levels is given by:

$$(G - R)_{SRH} = \frac{n_i^2 - np}{\tau_p \left[ n + n_i \exp\left(\frac{E_t - E_i}{k_B T}\right) \right] + \tau_n \left[ p + n_i \exp\left(\frac{E_i - E_t}{k_B T}\right) \right]}$$

where  $\tau_n$  and  $\tau_p$  typically lie in the range 100 ns to 5  $\mu$ s.

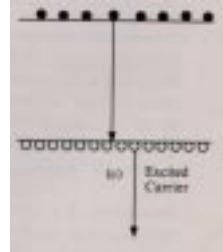
- For surface states, the carrier lifetimes are replaced with the inverse of the surface recombination velocities ( $10^{-2} \text{ m}^2\text{s}^{-1}$ )



- The **Auger recombination** is accounted for with the following expression:

$$(G - R)_{Auger} = (n_i^2 - np)(nC_n + pC_p)$$

where  $C_n$  and  $C_p$  are of the order of  $3 \times 10^{-43} \text{ m}^6 \text{ s}^{-1}$  and  $10^{-43} \text{ m}^6 \text{ s}^{-1}$



- The generation process due to **Impact Ionization** can be included using the field-dependent rate:

$$(G - R)_{II} = \frac{1}{e} \left[ A_n \exp \left[ - \left( \frac{E_n^{crit}}{E} \right)^{\beta_n} \right] |J_n| + A_p \exp \left[ - \left( \frac{E_p^{crit}}{E} \right)^{\beta_p} \right] |J_p| \right]$$

where  $A_n$  and  $A_p$  are in the range from  $10^7 \text{ m}^{-1}$  to  $2 \times 10^9 \text{ m}^{-1}$

Scenario for the impact ionization process



- Photon transition** (narrow band-gap and direct band-gap):

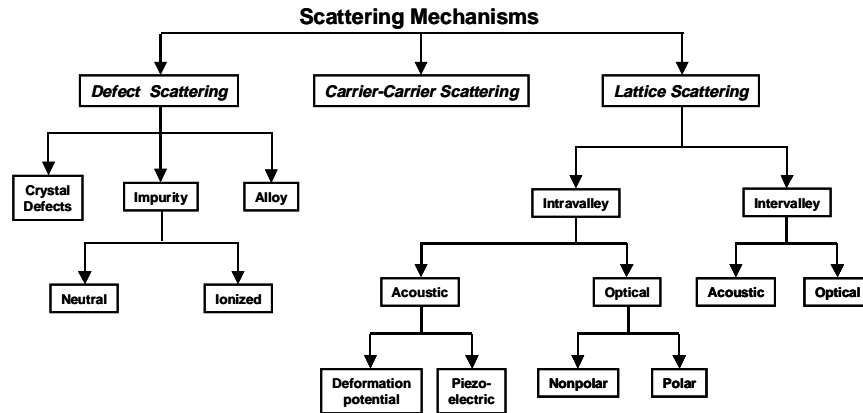
$$(G - R)_{optical} = C_{opt} (n_i^2 - np)$$

where  $C_{opt}$  is the optical capture-emission rate



## 2.9 Mobility Models

Summarized below are the various scattering mechanisms that affect the magnitude of the electron or hole mobility:



Mobility modeling is normally divided into:

- Low-field mobility models (**bulk materials** and **inversion layers**)
- High-field mobility models

- **Bulk mobility:**

- 1) Characterization of  $\mu_0$  as a function of doping and lattice scattering
- 2) Characterization of  $v_{sat}$  as a function of lattice temperature
- 3) Describing the transition between the low-field and the saturation velocity region

- **Inversion layers:**

- 1) Characterization of surface-roughness scattering
- 2) Description of the carrier-carrier scattering
- 3) Quantum-mechanical size-quantization effect

### **(A) Low-field models for bulk materials**

#### *Phonon scattering:*

- Simple power-law dependence of the temperature
- Sah et al. model:  
acoustic + optical and intervalley phonons combined via Mathiessen's rule

#### *Ionized impurity scattering:*

- Conwell-Weiskopf model
- Brooks-Herring model

#### *Combined phonon and ionized impurity scattering:*

- Dorkel and Leturg model:  
temperature-dependent phonon scattering + ionized impurity scattering + carrier-carrier interactions
- Caughey and Thomas model:  
temperature independent phonon scattering + ionized impurity scattering



- Sharfetter-Gummel model:  
phonon scattering + ionized impurity scattering (parameterized expression – does not use the Mathiessen's rule)
- Arora model:  
similar to Caughey and Thomas, but with temperature dependent phonon scattering

#### *Carrier-carrier scattering*

- modified Dorkel and Leturg model

#### *Neutral impurity scattering:*

- Li and Thurber model:  
mobility component due to neutral impurity scattering is combined with the mobility due to lattice, ionized impurity and carrier-carrier scattering via the Mathiessen's rule



### **(B) Field-dependent mobility**

The field-dependent mobility model provides smooth transition between low-field and high-field behavior

$$\mu(E) = \frac{\mu_0}{\left[1 + \left(\frac{\mu_0 E}{v_{sat}}\right)^\beta\right]^{1/\beta}} \quad \begin{array}{l} \beta = 1 \text{ for electrons} \\ \beta = 2 \text{ for holes} \end{array}$$

$v_{sat}$  is modeled as a temperature-dependent quantity:

$$v_{sat}(T) = \frac{2.4 \times 10^7}{1 + 0.8 \exp\left(\frac{T_L}{600}\right)} \text{ cm/s}$$

### **(C) Inversion layer mobility models**

- **CVT model:**
  - combines acoustic phonon, non-polar optical phonon and surface-roughness scattering (as an inverse square dependence of the perpendicular electric field) via Mathiessen's rule
- **Yamaguchi model:**
  - low-field part combines lattice, ionized impurity and surface-roughness scattering
  - there is also a parametric dependence on the in-plane field (high-field component)
- **Shirahata model:**
  - uses Klaassen's low-field mobility model
  - takes into account screening effects into the inversion layer
  - has improved perpendicular field dependence for thin gate oxides
- **Tasch model:**
  - the best model for modeling the mobility in MOS inversion layers; uses universal mobility behavior

## 2.10 Gummel's and Newton's Schemes

- There are two different methods to solve the set of coupled equations that comprise the DD-model:
  - Gummel's scheme
  - Newton's scheme

### (A) Gummel's Relaxation Method

Gummel's relaxation method, which solves the equations with the decoupled procedure, is used in the case of weak coupling:

- Low current densities (leakage currents, subthreshold regime), where the concentration dependent diffusion term in the current continuity equation is dominant
- The electric field strength is lower than the avalanche threshold, so that the generation term is independent of  $\nabla V$
- The mobility is nearly independent of  $E$

The computational cost of the Gummel's iteration is one matrix solution for each carrier type plus one iterative solution for the linearization of the Poisson Equation

The solution strategy when using Gummel's relaxation scheme is the following one:

- Find the equilibrium solution of the linearized Poisson equation

$$\frac{d^2 \delta \bar{V}}{dx^2} - \frac{n_i}{N} [\exp(\bar{V}) + \exp(-\bar{V})] \delta \bar{V} = -\frac{d^2 \bar{V}}{dx^2} + \frac{n_i}{N} \left[ \exp(\bar{V}) - \exp(-\bar{V}) + \frac{N_A - N_D}{n_i} \right]$$

- After the solution in equilibrium is obtained, the applied voltage is increased in steps  $\Delta V \leq V_T$
- Now the scaled Poisson equation becomes:

$$\frac{d^2 \bar{V}}{dx^2} = \frac{n_i}{N} \left[ \exp(-\bar{\phi}_n) \exp(\bar{V}) - \exp(\bar{\phi}_p) \exp(-\bar{V}) + \frac{N_A - N_D}{n_i} \right]$$

The 1D discretized electron current continuity equation (as long as Einstein's relations are valid) is:

$$\frac{D_{i+1/2}}{a_i} [n_{i+1}B(\bar{V}_{i+1} - \bar{V}_i) - n_i B(\bar{V}_i - \bar{V}_{i+1})] + \frac{D_{i-1/2}}{a_{i-1}} [n_{i-1}B(\bar{V}_{i-1} - \bar{V}_i) - n_i B(\bar{V}_i - \bar{V}_{i-1})] + \frac{1}{2}(a_{i-1} + a_i)(G - R)_i = 0$$

For holes, one can obtain analogous equations by substituting:

$$V \rightarrow -V, \quad n \rightarrow p$$

The *decoupled* iteration scheme goes as follows:

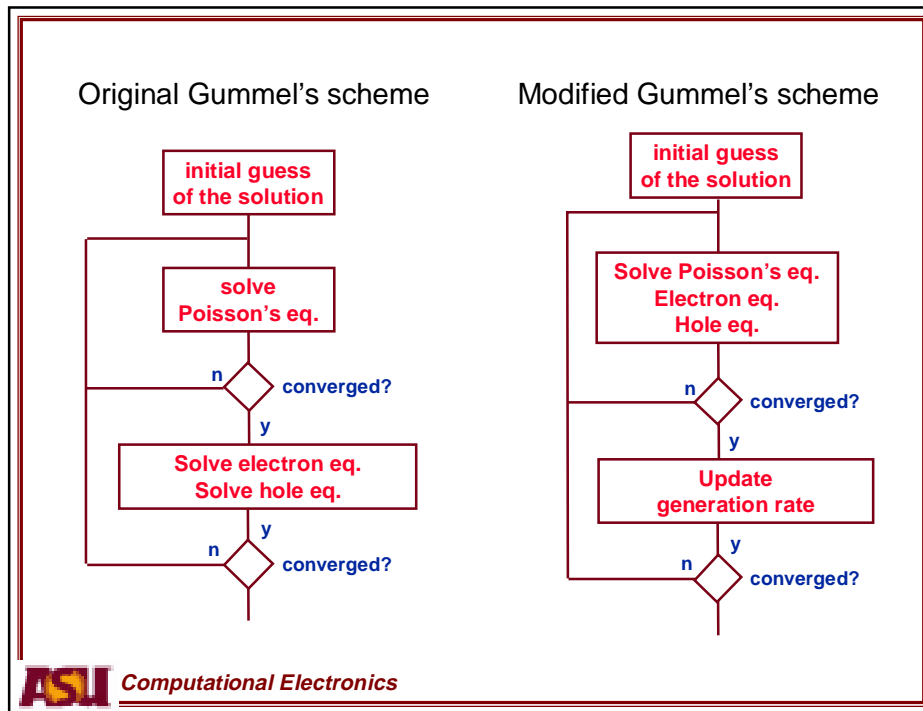
- (1) Solve the Poisson equation with a guess for the quasi-Fermi levels (use the applied voltage as initial guess)
- (2) The potential is used to update the Bernoulli functions
- (3) The above equations are solved to provide an update for the quasi-Fermi levels, that enter into the Poisson equation

The criterion for convergence is:

$$\max |V^{k+1} - V^k|, \quad \max |V_T \ln \left( \frac{n^k}{n^{k+1}} \right)|, \quad \max |V_T \ln \left( \frac{p^k}{p^{k+1}} \right)|$$

In the case of strong coupling, one can use the extended Gummel's scheme

$$\begin{aligned} V^{k+1} &= V^k + \delta V \\ n^{k+1} &= n^k \exp \left[ \frac{\delta V}{V_T} \right] \left( 1 - \delta V + V_T \ln \left[ \frac{\delta n}{n^k} \right] \right) \\ p^{k+1} &= p^k \exp \left[ \frac{\delta V}{V_T} \right] \left( 1 - \delta V + V_T \ln \left[ \frac{\delta p}{p^k} \right] \right) \end{aligned}$$



(B) Newton's method

- The three equations that constitute the DD model, written in residual form are:

$$F_V(v, n, p) = 0 \quad F_n(v, n, p) = 0 \quad F_p(v, n, p) = 0$$

- Starting from an initial guess, the corrections are calculated by solving:

$$\begin{bmatrix} \frac{\partial F_V}{\partial v} & \frac{\partial F_V}{\partial n} & \frac{\partial F_V}{\partial p} \\ \frac{\partial F_n}{\partial v} & \frac{\partial F_n}{\partial n} & \frac{\partial F_n}{\partial p} \\ \frac{\partial F_p}{\partial v} & \frac{\partial F_p}{\partial n} & \frac{\partial F_p}{\partial p} \end{bmatrix} \cdot \begin{bmatrix} \Delta v \\ \Delta n \\ \Delta p \end{bmatrix} = \begin{bmatrix} F_V \\ F_n \\ F_p \end{bmatrix} \Rightarrow \begin{aligned} V^{k+1} &= V^k + \Delta V^k \\ n^{k+1} &= n^k + \Delta n^k \\ p^{k+1} &= p^k + \Delta p^k \end{aligned}$$

**ASU** Computational Electronics

- The method can be simplified by the following iterative scheme:

$$\begin{bmatrix} \frac{\partial F_V}{\partial V} & 0 & 0 \\ \frac{\partial F_n}{\partial V} & \frac{\partial F_n}{\partial n} & 0 \\ \frac{\partial F_p}{\partial V} & \frac{\partial F_p}{\partial n} & \frac{\partial F_p}{\partial p} \end{bmatrix} \cdot \begin{bmatrix} \Delta V \\ \Delta n \\ \Delta p \end{bmatrix} = - \begin{bmatrix} F_V \\ F_n \\ F_p \end{bmatrix} - \begin{bmatrix} 0 & \frac{\partial F_V}{\partial n} & \frac{\partial F_V}{\partial p} \\ 0 & 0 & \frac{\partial F_n}{\partial p} \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta V \\ \Delta n \\ \Delta p \end{bmatrix}$$

$\downarrow$   
k+1
 $\downarrow$   
k

$$\frac{\partial F_V}{\partial V} \Delta V^{k+1} = -F_V - \frac{\partial F_V}{\partial n} \Delta n^k - \frac{\partial F_V}{\partial p} \Delta p^k$$

$$\frac{\partial F_n}{\partial V} \Delta n^{k+1} = -F_n - \frac{\partial F_n}{\partial V} \Delta V^{k+1} - \frac{\partial F_n}{\partial p} \Delta p^k$$

$$\frac{\partial F_p}{\partial p} \Delta p^{k+1} = -F_p - \frac{\partial F_p}{\partial V} \Delta V^{k+1} - \frac{\partial F_p}{\partial n} \Delta n^{k+1}$$

## 2.11 Existing Drift-Diffusion Simulators

- Silvaco ATLAS** device simulator → MOSFETs, BJT's, heterojunction devices, thin-film transistors, etc.
- FIELDAY (IBM)** → bipolar and field-effect transistors  
(SRH + Auger recombination, avalanche and photogeneration, impurity-dependent mobility, band-gap narrowing effect, Schottky contacts)
- TOPMOST (Toshiba)** → MOS transistors with small geometries
- CADDETH (Hitachi)** → MOS and bipolar transistors, memory cells (effective intrinsic carrier densities, impurity and field-dependent mobilities, SRH, Auger and surface recombination, Avalanche generation and Zener tunneling, currents generated by photons and alpha particles, polysilicon and Schottky contacts)
- MINIMOS (Technical University in Vienna)** → MOSFET's  
(most sophisticated physical models for carrier transport in MOS structures, including energy transport)
- MEDES (ETH Zurich)** → abrupt diodes, BJT's, LOCOS, trench cell structures (doping and field-dependent mobilities, band-gap narrowing, SRH and Auger recombination, global device temperature)
- HFIELDS (University of Bologna)** → narrow-channel effects in MOSFET's
- DA VINCI (TMA)** → transient and steady-state analysis of any structure  
(Concentration and field-dependent mobilities, impact ionization and radiation induced carrier generation)

### 3. Hydrodynamic Simulator

- 3.1 Derivation of the basic hydrodynamic equations
- 3.2 Ensemble relaxation rates and their calculation
- 3.3 Discretization of the balance or hydrodynamic equations

#### 3.1 Derivation of the Basic Hydrodynamic Equations

- To derive the Balance Equations, one starts with the BTE , multiplies it with an appropriate function  $\phi(\mathbf{k})$  and integrates over  $\mathbf{k}$  to get:

$$\frac{\partial \Phi}{\partial t} = -\nabla_{\mathbf{r}} \cdot \mathbf{J}_{\phi} + G_{\phi} - R_{\phi}$$

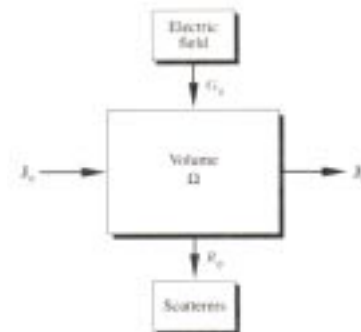
where:

$$\Phi(\mathbf{r}, t) = \int d\mathbf{p} \phi(\mathbf{p}) f(\mathbf{r}, \mathbf{p}, t)$$

$$\mathbf{J}_{\phi}(\mathbf{r}, t) = \int d\mathbf{p} \phi(\mathbf{p}) \mathbf{v}(\mathbf{p}) f(\mathbf{r}, \mathbf{p}, t)$$

$$G_{\phi}(\mathbf{r}, t) = -e\mathbf{E} \cdot \int d\mathbf{p} (\nabla_{\mathbf{p}} \phi) f(\mathbf{r}, \mathbf{p}, t)$$

$$R_{\phi}(\mathbf{r}, t) = -\int d\mathbf{p} \phi(\mathbf{p}) \left. \frac{\partial f}{\partial t} \right|_{coll}$$



- Balance equation for the carrier density is obtained by assuming that  $\phi(\mathbf{p})=1$ :

$$\Phi(\mathbf{r}, t) = n(\mathbf{r}, t)$$

$$\mathbf{J}_\phi(\mathbf{r}, t) = \int d\mathbf{p} \mathbf{v}(\mathbf{p}) f(\mathbf{r}, \mathbf{p}, t) = -\frac{J_n}{e}$$

$$G_\phi(\mathbf{r}, t) = 0$$

$$R_\phi(\mathbf{r}, t) = -\left(\frac{\partial n}{\partial t}\right)_{coll}$$

- Using  $\mathbf{v}=\mathbf{v}_d+\mathbf{c}$ , one gets the final form of the balance equation for the carrier density:

$$\frac{\partial n}{\partial t} = -\nabla \cdot (n\mathbf{v}_d) + \left(\frac{\partial n}{\partial t}\right)_{coll}$$

- Momentum Balance equation is obtained by assuming that  $\phi(\mathbf{p})=\mathbf{p}$ :

$$\Phi(\mathbf{r}, t) = \mathbf{P}_d = n\mathbf{p}_d$$

$$\mathbf{J}_\phi(\mathbf{r}, t) = \mathbf{J}_p(\mathbf{r}, t) = \int d\mathbf{p} \mathbf{v}(\mathbf{p}) \mathbf{p} f(\mathbf{r}, \mathbf{p}, t)$$

$$G_\phi(\mathbf{r}, t) = -en\mathbf{E}$$

$$R_\phi(\mathbf{r}, t) = -\left(\frac{\partial(n\mathbf{p}_d)}{\partial t}\right)_{coll}$$

- This leads to the following momentum balance equation:

$$\frac{\partial(n\mathbf{p}_d)}{\partial t} = -\nabla \cdot \mathbf{J}_p - en\mathbf{E} + \left(\frac{\partial(n\mathbf{p}_d)}{\partial t}\right)_{coll}$$

- For the flow of the momentum component  $p_x$  we have that:

$$\nabla \cdot \mathbf{J}_{p_x} = \nabla \cdot \int d\mathbf{p} \mathbf{v} p_x f = \frac{\partial J_{p,xx}}{\partial x} + \frac{\partial J_{p,yx}}{\partial y} + \frac{\partial J_{p,zx}}{\partial z}$$

- Using again  $\mathbf{v} = \mathbf{v}_d + \mathbf{c}$ , one gets:

$$J_{p,ij} = \int d\mathbf{p} v_i p_j f = nm * v_{di} v_{dj} + nm * \langle c_i c_j \rangle$$

- Assuming diagonal temperature tensor, the above equation simplifies to:

$$\nabla \cdot \mathbf{J}_{p_x} = \frac{\partial (nv_{dx} p_{dx})}{\partial x} + \frac{\partial (nv_{dy} p_{dx})}{\partial y} + \frac{\partial (nv_{dz} p_{dx})}{\partial z} + \frac{\partial (nk_B T)}{\partial z}$$

- The final form of the momentum balance equation is:

$$\frac{\partial (n\mathbf{p}_d)}{\partial t} = -\nabla \cdot (n\mathbf{v}_d \mathbf{p}_d) - \nabla (nk_B T) - en\mathbf{E} + \left( \frac{\partial (n\mathbf{p}_d)}{\partial t} \right)_{coll}$$

- The Energy Balance equation is obtained by assuming that  $\phi(\mathbf{p}) = E(\mathbf{p})$ :

$$\Phi(\mathbf{r}, t) = W = nw$$

$$\mathbf{J}_\phi(\mathbf{r}, t) = \mathbf{J}_w(\mathbf{r}, t) = \int d\mathbf{p} \mathbf{v}(\mathbf{p}) E(\mathbf{p}) f(\mathbf{r}, \mathbf{p}, t)$$

$$= nw\mathbf{v}_d + nk_B T \mathbf{v}_d + n\mathbf{q}$$

$$G_\phi(\mathbf{r}, t) = -e\mathbf{E} \cdot (n\mathbf{v}_d)$$

$$R_\phi(\mathbf{r}, t) = -\left( \frac{\partial (nw)}{\partial t} \right)_{coll}$$

- The energy balance equation is then of the form:

$$\frac{\partial (nw)}{\partial t} = -\nabla \cdot (nw\mathbf{v}_d + nk_B T \mathbf{v}_d + n\mathbf{q}) - en\mathbf{E} \cdot \mathbf{v}_d + \left( \frac{\partial (nw)}{\partial t} \right)_{coll}$$

- To have a closed set of equations, one either:
  - (a) ignores the heat flux altogether
  - (b) uses a simple recipe for the calculation of the heat flux:

$$n\mathbf{q} = -\kappa\nabla T, \quad \kappa = \frac{5k_B^2 n T}{2m^* v(w)}$$

- Substituting  $T$  with the density of the carrier energy, the momentum and energy balance equations become:

$$\begin{aligned} \frac{\partial(n\mathbf{p}_d)}{\partial t} &= -\nabla \cdot (n\mathbf{v}_d \mathbf{p}_d) - \frac{2}{3} \nabla \left( nw - \frac{1}{2} nm^* v_d^2 \right) - en\mathbf{E} + \left( \frac{\partial(n\mathbf{p}_d)}{\partial t} \right)_{coll} \\ \frac{\partial(nw)}{\partial t} &= -\nabla \cdot \left[ n\mathbf{v}_d w + \frac{2}{3} \nabla \left( n\mathbf{v}_d - \frac{\kappa}{k_B} \nabla \right) \left( w - \frac{1}{2} m^* v_d^2 \right) \right] \\ &\quad - en\mathbf{E} \cdot \mathbf{v}_d + \left( \frac{\partial(nw)}{\partial t} \right)_{coll} \end{aligned}$$

- More convenient set of balance equations is in terms of  $n$ ,  $\mathbf{v}_d$  and  $w$ :

$$\begin{aligned} \frac{\partial(n)}{\partial t} &= -\nabla \cdot (n\mathbf{v}_d) + \left( \frac{\partial(n)}{\partial t} \right)_{coll} \\ \frac{\partial(\mathbf{v}_d)}{\partial t} &= -\frac{\mathbf{v}_d}{m^*} \cdot \nabla (m^* \mathbf{v}_d) - \frac{2}{3nm^*} \nabla \left( nw - \frac{1}{2} nm^* v_d^2 \right) \\ &\quad - \frac{e\mathbf{E}}{m^*} + \left( \frac{\partial(\mathbf{v}_d)}{\partial t} \right)_{coll} \\ \frac{\partial(w)}{\partial t} &= -\mathbf{v}_d \cdot \nabla w - \frac{2}{3n} \nabla \cdot \left[ \left( n\mathbf{v}_d - \frac{\kappa}{k_B} \nabla \right) \left( w - \frac{m^* v_d^2}{2} \right) \right] \\ &\quad - e\mathbf{E} \cdot \mathbf{v}_d + \left( \frac{\partial(w)}{\partial t} \right)_{coll} \end{aligned}$$

(A) Extensions of the model  
beyond parabolic band approximation

- The non-parabolicity of the conduction band can be included by using the energy-dependent effective masses. If one uses a hyperbolic band model, then:

$$E(\mathbf{k}) = \frac{\sqrt{1+4\alpha\gamma(\mathbf{k})}-1}{2\alpha}, \quad \gamma(\mathbf{k}) = \frac{\hbar^2 k^2}{2m^*}$$

- The energy dependent effective mass is then given by:

$$\frac{1}{m^{**}} = \frac{1}{\hbar^2} \frac{\partial^2 E_k}{\partial k^2} = \frac{1}{m^*} \frac{1}{[1+4\alpha\gamma(\mathbf{k})]^{3/2}}$$

(B) Reduction to the drift-diffusion model

- The DD model is obtained by simplifying the momentum balance equation, which in 1D is of the form:

$$\begin{aligned} \frac{\partial(P_{dz})}{\partial t} &= -\frac{\partial}{\partial z}(nv_d p_d) - \frac{\partial}{\partial z}(nk_B T) - enE + \left(\frac{\partial(P_z)}{\partial t}\right)_{coll} \\ &= -\frac{\partial}{\partial z}(2nw) - enE - \left\langle \frac{1}{\tau_m} \right\rangle P_z \end{aligned}$$

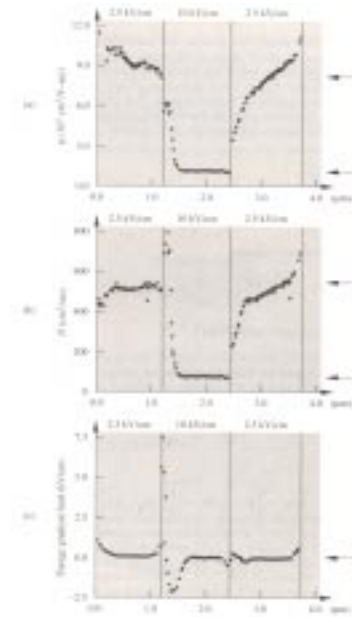
- In steady-state, one gets for the momentum balance equation that:

$$J_z = ne\mu_n [E + E'] + eD_n \frac{\partial n}{\partial z} \quad \text{where:} \quad E' = \frac{2}{e} \frac{\partial w}{\partial z}$$

$$D_n = \frac{2\mu_n w}{e}$$

The DD model is then obtained by making the following assumptions:

- The distribution function is close to the equilibrium value
- The energy gradient field is zero.
- In the extended DD model  $\mu(\mathbf{E})$  and  $D(\mathbf{E})$  are assumed to depend upon the local field values only.

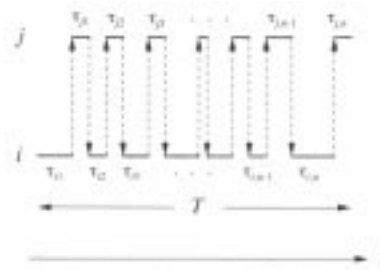


### 3.2 Ensemble Relaxation Rates

- The ensemble relaxation rate, which appears in the carrier density balance equation, is related to the intervalley transfer in many-valley semiconductors. It equals to:

$$\left(\frac{\partial n}{\partial t}\right)_{coll} = -v_n(n - n_0)$$

One usually calculates this ensemble relaxation rate by using Monte Carlo simulation:



$$v_{n,ij} = \frac{N_{i \rightarrow j}}{T_i}$$

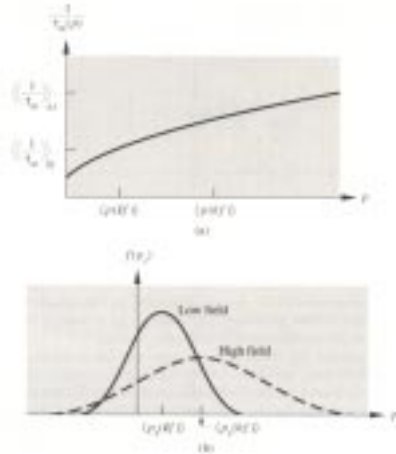
$$v_{n,ji} = \frac{N_{j \rightarrow i}}{T_j}$$

- To see how one can calculate the **momentum and the energy ensemble relaxation rates**, it is necessary to go back to the definition of  $R_\phi$ :

$$\begin{aligned}
 R_\phi &= v_\phi (\Phi - \Phi_0) \\
 &= -\int d\mathbf{p} \phi(\mathbf{p}) \left. \frac{\partial f}{\partial t} \right|_{coll} \\
 &= \int d\mathbf{p} \frac{\phi(\mathbf{p}) f(\mathbf{r}, \mathbf{p}, t)}{\tau_\phi(\mathbf{p})}
 \end{aligned}$$

where the relaxation time  $\tau_\phi$  is:

$$\frac{1}{\tau_\phi(\mathbf{p})} = \sum_{\mathbf{p}'} \left[ 1 - \frac{\phi(\mathbf{p}')}{\phi(\mathbf{p})} \right] S(\mathbf{p}, \mathbf{p}')$$

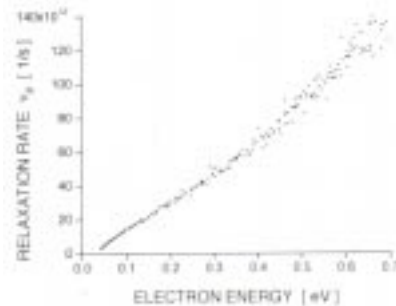


### (A) Momentum relaxation rate

- The **momentum rate** is determined by a steady-state MC calculation in a bulk semiconductor under a uniform bias electric field, for which:

$$\frac{\partial \mathbf{v}_d}{\partial t} = -\frac{e\mathbf{E}}{m^*} + \left( \frac{\partial \mathbf{v}_d}{\partial t} \right)_{coll} = -\frac{e\mathbf{E}}{m^*} - v_p(w) \mathbf{v}_d = 0$$

$$v_p(w) = \frac{eE}{m^* v_d}$$

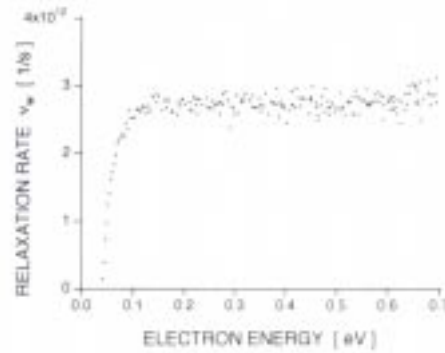


### (B) Energy relaxation rate

- The ensemble **energy relaxation rate** is also determined by a steady-state MC calculation in a bulk semiconductor under a uniform bias electric field, for which:

$$\frac{\partial w}{\partial t} = -e\mathbf{E} \cdot \mathbf{v}_d + \left( \frac{\partial w}{\partial t} \right)_{coll} = -e\mathbf{E} \cdot \mathbf{v}_d - v_w(w - w_0) = 0$$

$$v_w(w) = -\frac{e\mathbf{E} \cdot \mathbf{v}_d}{w - w_0}$$



### **3.3 Discretization of the Balance Equations**

- For simplicity, the equations will be discretized on an equally spaced meshes. In 1D and in finite-difference operator notation, the RHS's of the equations that need to be solved are:

$$G_{n,i} = -n_i L_x(v_i) - v_i L_x(n_i)$$

$$G_{v,i} = -v_i L_x(v_i) - \frac{2}{3n_i m^*} L_x \left( n_i w_i - \frac{1}{2} n_i m^* v_i^2 \right) - \frac{eE_i}{m^*} - v_p(w_i) v_i$$

$$G_{w,i} = -v_i L_x(w_i) - \frac{2}{3n_i} L_x \left[ n_i v_i \left( w_i - \frac{m^* v_i^2}{2} \right) \right] + \frac{2\kappa}{3n_i k_B} L_{xx} \left( w_i - \frac{m^* v_i^2}{2} \right) - eE_i v_i - v_w(w_i)(w_i - w_0)$$

$$G_{\phi,i} = L_{xx}(\phi_i) + \frac{e}{\epsilon} (N_{D,i} - n_i)$$

- The various explicit schemes that one can use are listed below:

(a) Forward-time centered-space scheme (FTCS):

$$L_x[f_i] = \frac{f_{i+1} - f_{i-1}}{2\Delta x}, \quad L_{xx}[f_i] = \frac{f_{i-1} - 2f_i + f_{i+1}}{\Delta x^2}, \quad \left. \frac{\partial f}{\partial t} \right|_i \approx \frac{f_i^f - f_i}{\Delta t}$$

(b) Upwind scheme:

$$v_i L_x^{uw}[f_i] = \begin{cases} v_i \frac{f_i - f_{i-1}}{\Delta x}, & v_i \geq 0 \\ v_i \frac{f_{i+1} - f_i}{\Delta x}, & v_i < 0 \end{cases}$$

(c) Lax-Wendroff scheme:

$$v_i L_x^{LW}[f_i] = v_i L_x[f_i] - v_i^2 \frac{\Delta t}{2} L_{xx}[f_i]$$

(d) DuFort-Frankel scheme:

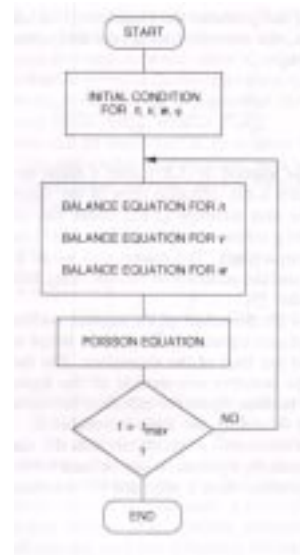
$$L_{xx}[f_i] = \frac{f_{i-1}^f - f_i^f - f_i^p + f_{i+1}^f}{\Delta x^2}$$

$$\left. \frac{\partial f}{\partial t} \right|_i \approx \frac{f_i^f - f_i^p}{2\Delta t}$$

(e) Leapfrog scheme:

$$L_x^{lf}[f_i] = \frac{f_{i+1} - f_{i-1}}{2\Delta x}$$

$$\left. \frac{\partial f}{\partial t} \right|_i \approx \frac{f_i^f - f_i^p}{2\Delta t}$$

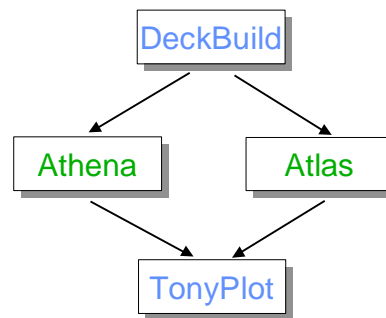


## 4. Introduction to the Silvaco ATLAS Tool

- 4.1 Some general comments
- 4.2 Deckbuild overview
- 4.3 ATLAS syntax
  - (A) Structure specification
  - (B) Materials models specification
  - (C) Numerical method selection
  - (D) Solution specification
  - (E) Results analysis
- 4.4 ATLAS Extract description
- 4.5 Examples
  - (A) Diode example
  - (B) Bipolar junction transistor simulation
  - (C) MOSFET simulation

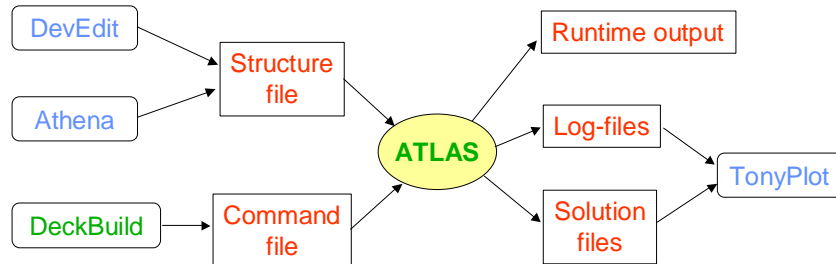
### 4.1 Some general comments

- ⊙ **The VWF** (Virtual Wafer Fab) Framework consists of two different sets of tools:
  - ➔ core tools
  - ➔ auxiliary tools
- ⊙ **ATHENA** - process simulation tool
  - predicts the physical structure that results from the processing steps
  - treats process simulation as a serial flow of events
- ⊙ **ATLAS** - device simulation tool
  - performs physically-based 2D/3D device simulations
  - predicts the electrical behavior of specified semiconductor structures and provides insight into the internal physical mechanisms associated with the device operation
  - various tools that comprise ATLAS include: S-PISCES, BLAZE, GIGA, TFT, LUMINOUS, LASER, MIXEMODE, DEVICE3D, INTERCONNECT3D, THERMAL3D



⊙ **ATLAS Inputs and Outputs:**

- Most ATLAS simulations use two types of inputs: **text files** and **structure files**
- There are three types of outputs produced by ATLAS:
  - 1) **Runtime output** - guide to the progress of simulation that is running
  - 2) **Log files** - summaries of the electrical output information
  - 3) **Solution files** - store 2D and 3D data relating to the values of the solution variables



⊙ **Modes of operation:**

There are three different modes of operation of ATLAS:

**1) Interactive mode with DeckBuild**

```
deckbuild -as <input_filename>
```

**2) Batch mode with DeckBuild**

With X-Windows operation:

```
deckbuild -run -as <input_filename> -outfile <output_filename>
```

Without X-Windows operation:

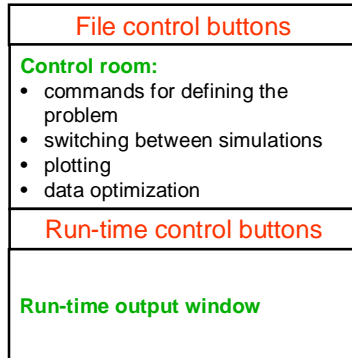
```
deckbuild -run -ascii -as <input_filename> -outfile  
<output_filename>
```

**3) Batch mode without DeckBuild**

```
atlas <input_filename> -logfile <output_filename>
```

## 4.2 DeckBuild Overview

- ⊙ To start DeckBuild, one needs to type: `deckbuild &`
- ⊙ When DeckBuild starts, the following application window pops up:



### Used for:

- importing previously saved ASCII files describing a structure of interest
- **Main control** button contains: **Optimizer** and **Examples**

### Used for controlling the way the simulator is run:

- **next** - sends current line to simulator
- **run** - runs deck from top to bottom
- **quit** - sends a quit statement to the simulator
- **restart** - restarts the current simulator
- **kill** - kills the simulator



## 4.3 ATLAS Syntax

- ⊙ The form of the input file statements is:

`<STATEMENT> <PARAMETER> = <VALUE>`

The parameter can be: real, integer, character and logical.

- ⊙ The order in which the ATLAS commands occur is the following:

**A) Structure specification:** MESH, REGION, ELECTRODE, DOPING

**B) Material models specification:** MATERIAL, MODELS, CONTACT, INTERFACE

**C) Numerical method selection:** METHOD

**D) Solution specification:** LOG, SOLVE, LOAD, SAVE

**E) Results analysis:** EXTRACT, TONYPLOT

- ⊙ The input file can be created using the DeckBuild Command Menu:

**Commands/Command Menu**



### (A) STRUCTURE SPECIFICATION

#### ⊙ MESH statement specification

**INFILE, OUTFILE** → file with previously saved mesh, new file  
**SPACE.MULT** → scale factor applied to all specified grid spacing  
**CYLINDRICAL, RECTANGULAR** → describes mesh symmetry  
**NX, NY** → number of nodes along the x- and y-direction  
`mesh nx=36 ny=30`

#### ⊙ X.MESH, Y.MESH statements - Specify the location of grid lines along the x- and y-axes

**NODE** → specifies mesh line index  
**LOCATION** → specifies the location of the grid line  
**RATIO** → ratio to be used when interpolating grid lines between given locations  
**SPACING** → specifies mesh spacing at a given location  
`x.mesh loc = 0.0 spacing = 0.2`  
`x.mesh loc = 0.85 spacing = 0.01`  
`x.mesh loc = 2 spacing = 0.3`

#### ⊙ ELIMINATE statement

Eliminates every second mesh point in a rectangular grid specified by **X.MIN**, **X.MAX**, **Y.MIN** and **Y.MAX**

**COLUMNS, ROWS** → columns, rows elimination  
`eliminate x.min=0 x.max=4 y.min=0 y.max=3`

#### ⊙ REGION statement - Specifies regions and materials

**NUMBER** → denotes region number  
**material** → can be **SILICON**, **OXIDE**  
**position** → defines the location of the region in terms of (1) actual position and (2) grid nodes  
`region num=1 ix.lo=1 ix.hi=25 iy.lo=1 iy.hi=20 silicon`  
`region num=1 y.max=0 oxide`  
`region num=2 y.min=0 silicon`

#### ⊙ ELECTRODE statement - must specify at least one electrode within the simulation domain

**NAME** - defines the name of the electrode: **SOURCE**, **DRAIN**, **GATE**  
**position parameter** - **BOTTOM**, **LEFT**, **RIGHT**, **TOP**, **SUBSTRATE**,  
**IX.LOW**, **IX.HIGH**, **X.MIN**, **X.MAX**, **LENGTH**

⊙ **DOPING** statement

Can be used to set the doping profile analytically. Analytical doping profiles can be defined with the following parameters:

**distribution type** → UNIFORM, GAUSSIAN

**doping type** → N.TYPE, P.TYPE

**CONCENTRATION** → peak concentration specification for Gaussian profiles

**CHARACTERISTIC** → principal characteristic length of the implant (standard deviation). One can specify junction depth instead.

**PEAK** → specifies the location of a peak of a Gaussian profile

**position** → X.LEFT, X.RIGHT, REGION

```
doping uniform concentration=1E16 n.type region=1
```

```
doping gaussian concentration=1E18 characteristic=0.05 \  
p.type x.left=0 x.right=1.0 peak=0.1
```

The doping profile can also be imported from SSUPREM3. One must use the **MASTER** parameter in the doping statement combined with the **INFILE** parameter to be able to properly import the doping profile.

⊙ **COMMENTS ON THE MESH SET-UP**

(1) Defining a good mesh is a crucial issue in device simulations. There are several factors that need to be taken into account when setting the mesh:

**ACCURACY** - fine mesh is needed to properly resolve the structure

**EFFICIENCY** - for the simulation to finish in a reasonable time, fewer grid points must be used

(2) Critical areas where fine mesh is needed include

**depletion regions:** high-field regions

**Si/SiO<sub>2</sub> interface:** high transverse electric field region

**emitter/base junction of a BJT:** recombination is important

**impact ionization areas**

⊙ **REGRID** statement allows fine mesh generation in critical device areas. This statement is used after the MESH, REGION, MATERIAL, ELECTRODE, and DOPING statements. There are two ways in which regridting can be done:

regrid on **DOPING**

regrid using **SOLUTION VARIABLES**

## **(B) MATERIAL MODELS SPECIFICATION**

### ⊙ **CONTACT** statement

**NAME** → specifies the name of the contact: GATE, DRAIN, ANODE

**WORKFUNCTION** → specifies workfunction of a metal, or if specifies N.POLYSILICON, then it implicitly assumes one

**type** → specifies the type of a contact: CURRENT, VOLTAGE, FLOATING

**CONTACT IMPEDANCE** → uses RESISTANCE, CAPACITANCE, INDUCTANCE, CON.RESISTANCE

(used

for distributed contact resistance

specification)

**Schottky barrier** → BARRIER (turns on barrier lowering mechanism), ALPHA (specification of the barrier lowering)

```
contact name=gate workfunction=4.8
```

```
contact name=gate n.polysilicon
```

```
contact name=drain current
```

```
contact name=drain resistance=40.0 \
```

```
capacitance=20.E-12 inductance=1.E-6
```



Computational Electronics

### ⊙ **MATERIAL** statement

Atlas also supplies a default list of parameters for the properties of the material used in the simulation. The parameters specified in the MATERIAL statement include, for example: *electron affinity, energy bandgap, density of states function, saturation velocities, minority carrier lifetimes, Auger and impact ionization coefficients, etc.*

**REGION** → specifies the region number to which the above-described parameters apply

**parameters** → Some of the most commonly used parameters include: AFFINITY, EG300, MUN, MUP, NC300, NV300, PERMITTIVITY, TAUN0, TAUP0, VSATN, VSATP

```
material taun0=5.0E-6 taup0=5.0E-6 mun=3000 \
mup=500 region=2
```

```
material material=silicon eg300=1.2 mun=1100
```

### ⊙ **INTERFACE** statement – Specifies interface charge density and surface recombination velocity.

**QF, S.N, S.P** → amount of interface charge density, surface recombination velocity for electrons and holes

```
interface qf=3E10 x.min=1. x.max=2. y.min=0.
y.max=0.5
```

```
interface y.min=0 s.n=1E4 s.p=1E4
```



Computational Electronics

⊙ **MODELS** and **IMPACT** statements

The physical models that are specified with the MODELS and IMPACT statements include:

**mobility model** → CONMOB, ANALYTIC, ARORA, FLDMOB, TASCH, etc.

**recombination models** → SRH, CONSRH, AUGER, OPTR

**carrier statistics** → BOLTZMANN, FERMI, INCOMPLETE, IONIZ, BGN

**impact ionization** → CROWELL, SELB

**tunneling model** → FNORD, BBT.STD (band to band - direct transitions), BBT.KL (direct and indirect transitions), HEI and HHI (hot electron and hot hole injection)

`models conmob fldmob srh fermidirac  
impact selb`

Additional important parameters that can be specified within the MODELS statement include:

**NUMCARR** → specifies number of carriers, and is followed by a carrier type specification (ELECTRONS or HOLES or both)

**MOS, BIPOLAR** → standard models used for MOSFET and BIPOLARs

`models MOS numcarr=1 holes  
models BIP print`

**(C) NUMERICAL METHOD SELECTION**

⊙ **METHOD** statement – allows for several different choices of numerical method selection. The numerical methods that can be specified within the METHOD statement include

**GUMMEL** → De-coupled Gummel scheme which solves the necessary equations sequentially, providing linear convergence. Useful when there is weak coupling between the resultant equations.

**NEWTON** → Provides quadratic convergence, and needs to be used for the case of strong coupling between the resultant equations.

**BLOCK NEWTON** → more efficient than NEWTON method

`method gummel block newton  
method carriers=0`

One can also alter the parameters relevant for the numerical solution procedure:

**CLIMIT.DD** → Specifies minimum value of the concentration to be resolved by the solver.

**DVMAX** → Maximum potential update per iteration. Default value is 1V.

#### (D) SOLUTION SPECIFICATION

ATLAS allows for four different types of solutions to be calculated: **DC**, **AC**, **small signal** and **transient** solutions. The previously set bias at a given electrode is remembered and does not need to be set again.

##### ① DC solution procedures and statements:

- A stable DC solution is obtained with the following two-step procedure:
  - Find good initial guess by solving equilibrium case (initial guess is found based on the local doping density)  
`solve init`
  - Step the voltage on a given electrode for a convergent solution:  
`solve vcollector=2.0`  
`solve vbase=0.0 vstep=0.05 vfinal=1.0 name=base`
- To overcome the problems with poor initial guess, one can use the **TRAP** statement, where **MAXTRAPS** is the maximum allowed number of trials (default value is 4)  
`method trap`  
`solve init`  
`solve vdrain=2.0`



- To generate a family of curves, use the following set of commands:

```
solve vgate=1.0 outf=solve_vgate1
solve vgate=2.0 outf=solve_vgate2
load infile=solve_vgate1 log outfile=mos_drain_sweep1 \
  solve name=drain vdrain=0 vfinal=3.3 vstep=0.3
load infile=solve_vgate2 log outfile=mos_drain_sweep2 \
  solve name=drain vdrain=0 vfinal=3.3 vstep=0.3
```

The log statement is used to save the  $I_d/V_{ds}$  curve from each gate voltage to separate file.

##### ② AC solution procedures and statements:

The AC simulation is simply an extension to the DC simulation procedure. The final result of this analysis is the conductance and capacitance between each pair of electrodes. The two types of simulations are:

- Single frequency solution during a DC Ramp  
`solve vbase=0. vstep=0.05 vfinal=1 name=base AC freq=1e6`
- Ramped frequency at a single bias  
`solve vbase=0.7 ac freq=1e9 fstep=1e9 nsteps=10`  
`solve vbase=0.7 ac freq=1e6 fstep=2 mult.f nsteps=10`



③ Transient solution procedures and statements:

For transient solutions, one needs to use piecewise-linear, exponential and sinusoidal bias functions. For a linear ramp, one needs to specify the following parameters: **TSTART**, **TSTOP**, **TSTEP** and **RAMPTIME**.

```
solve vgate=1.0 ramptime=1e-9 tstop=10e-9 tstep=1e-11
```

④ Advanced solution procedures:

- Obtaining solutions around a breakdown point – uses MAXTRAPS
- Using current boundary conditions

Instead of voltage, one can also specify current boundary conditions. This is important, for example, when simulating BJTs:

```
solve ibase=1e-6  
solve ibase=1e-6 istep=1e-6 ifinal=5e-6 name=base
```

- The compliance parameter

This parameter is used to stop simulation when appropriate current level is reached.

```
solve vgate=1.0  
solve name=drain vdrain=0 vfinal=2 vstep=0.2 \  
compl=1e-6 cname=drain
```

- The curve trace capability – enables tracing out of complex IV curves



**(E) RESULTS ANALYSIS**

Three types of outputs are produced by the ATLAS tool: *run-time outputs*, *log files* and *solution files*.

① Run-time outputs:

The various parameters displayed during the SOLVE statement are listed below:

**proj** → initial guess methodology used (previous, local or init)

**i, j, m** → iteration numbers of the solution and the solution method

**i** = outer loop iteration number

**j** = inner loop number for decoupled solutions

**m** = solution method used: G=Gummel, B=Block, N=Newton

**x, rhs** → norms of the equations being solved

**(\*)** → the error measure has met its tolerance

② Log files:

The **LOG** parameter is used to store the device characteristics calculated using ATLAS:

```
log outfile=<file_name>
```



③ Solution files:

The syntax to produce the solution files that can be used in conjunction with TonyPlot is:

```
save outfile=<file_name>
solve . . . . outfile=<file_name>.sta master [onefileonly]
```

④ Invoking TonyPlot

→ To create overlaid plots with TonyPlot, one needs to use the following command:

```
tonyplot -overlay file1.log file2.log
```

→ To load structure files, containing mesh, doping profile information, etc., one can use the following statement:

```
tonyplot file.str -set mx.set iv.data
```

This command allows loading of the file called “*file.str*” and sets its display to a previous setup stored in the “*mx.set*” file, and then loads the file containing the *IV*-data.



The parameters extraction can be accomplished in two different ways:

1) Using the **EXTRACT** command that operates on previously solved curve or structure file:

→ To override the default of using open log file, the name of the file that needs to be used is specified in the following manner:

```
extract init infile="<file_name>"
```

→ Parameters that can be extracted using this EXTRACT statement include: threshold voltage, cutoff frequency, etc. The extraction of the threshold voltage is accomplished with the following statement:

```
extract name="nvt" xintercept(maxslope(curve (v."gate", \
(i."drain")))) -(ave(v."drain"))/2.0
```

→ Default file for saving results is results.final . The results can be stored in other file using the following options:

```
extract ... . Datafile="<file_name>"
```

2) Using the **Functions Menu** in TonyPlot that allows one to use saved data for post-computation

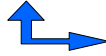
3) Using the LOG statement for AC parameter extraction



## 4.4 ATLAS Extract Description

(1) The extract statement can be used in conjunction with:

- ⊙ **Process extraction**, after running Silvaco ATHENA simulator
- ⊙ **Device extraction**, after obtaining the electrical characteristics of the device structure being simulated



- **Log-files**: contain the electrical information, more precisely, the *IV*-data obtained via the ATLAS simulation process
- **Structure files**: contain the additional electrical information, such as electric field, electrostatic potential, etc.

(2) One can construct a *curve* using separate X and Y-axes. For each of the electrodes, one can choose one of the following: **Voltage** (*v*), **Current** (*i*), **Capacitance** (*c*), **Conductance** (*g*), **Transient time** for AC simulations (*time*), **Frequency** for AC simulations (*frequency*), **Temperature** (*temperature*), etc.

(3) More in-depth description of the use of the EXTRACT statement:

- ⊙ **Curve**, basic element in the extract statement. The syntax is as follows:

```
extract name="curve_name" curve(v."name", i."name")
```

"*curve\_name*" = name of the curve to which one can refer to in later post-processing steps

- ⊙ **Axes manipulation:**

- algebra with a constant (multiplication, division)
- operators application (*abs*, *log*, *log10*, *sqrt*)

- ⊙ **Curve manipulation primitives:**

*min*, *max*, *ave*, *minslope*, *maxslope*, *slope*, *xintercept*, *yintercept*, *x.val* from curve where *y.val=Y* (*val.oceno=1*, would mean first occurrence of the preset condition)

- ⊙ **Example:** Find  $\max \beta = I_C / I_B$  vs.  $I_C$

```
extract "maxbeta" max(curve(i."colector", i."colector"/i."base"))
```

(\*) Additional set of examples for the EXTRACT statement can be found in the Silvaco ATLAS manual: *VWF Interactive Tools – part I*

## 4.5 Examples

### (A) Diode example

```
go atlas

# MESH SPECIFICATION PART
mesh space.mult=1.0
#
x.mesh loc=0.00 spac=0.5
x.mesh loc=3.00 spac=0.2
x.mesh loc=5.00 spac=0.25
x.mesh loc=7.00 spac=0.25
x.mesh loc=9.00 spac=0.2
x.mesh loc=12.00 spac=0.5
#
y.mesh loc=0.00 spac=0.1
y.mesh loc=1.00 spac=0.1
y.mesh loc=2.00 spac=0.2
y.mesh loc=5.00 spac=0.4

# REGIONS AND ELECTRODES SPECIFICATION
region num=1 silicon

electr name=anode x.min=5 length=2
electr name=cathode bot
```



Computational Electronics

```
# DOPING SPECIFICATION
#... N-epi doping
doping n.type conc=5.e16 uniform

#... Guardring doping
doping p.type conc=1e19 x.min=0 x.max=3 junc=1 rat=0.6 gauss
doping p.type conc=1e19 x.min=9 x.max=12 junc=1 rat=0.6 gauss

#... N+ doping
doping n.type conc=1e20 x.min=0 x.max=12 y.top=2 y.bottom=5 uniform

# SAVING THE MESH
save outf=diodeex01_0.str
tonyplot diodeex01_0.str -set diodeex01_0.set

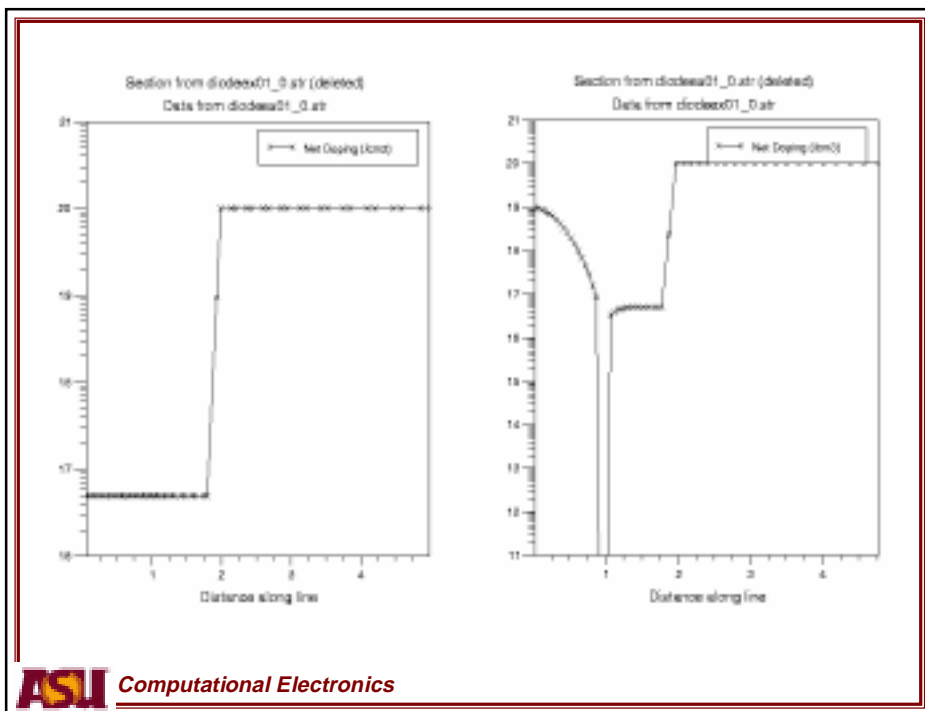
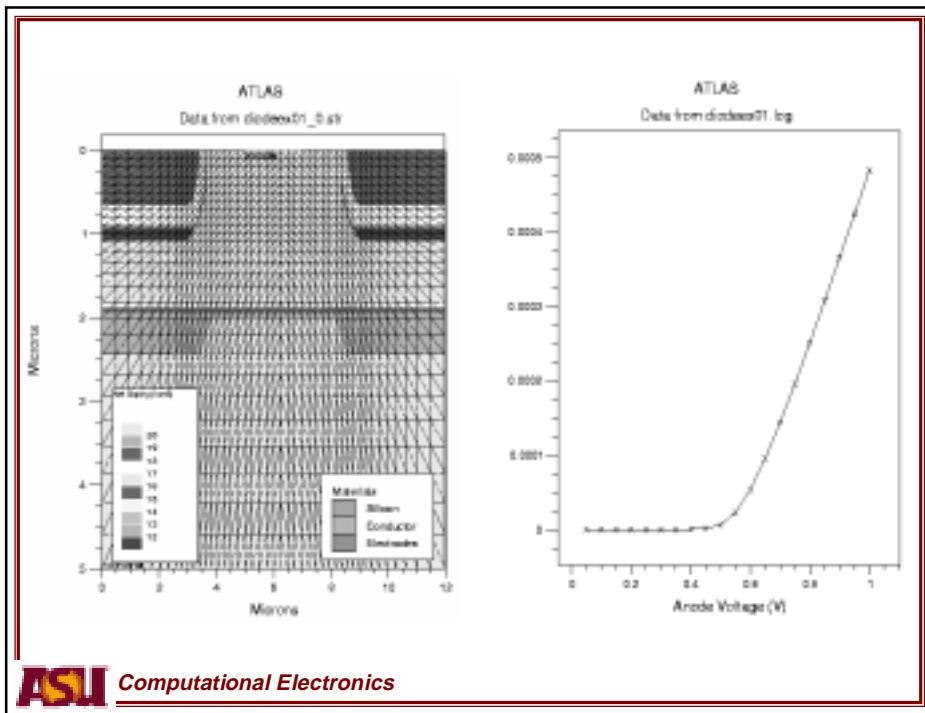
# MODELS SPECIFICATION
model conmob fldmob srh auger bgn
contact name=anode workf=4.97

# SOLUTION PART
#... Initial solution part
solve init
method newton

#... Stepping the anode voltage and saving the data
log outfile=diodeex01.log
Solve vanode=0.05 vstep=0.05 vfinal=1 name=anode
tonyplot diodeex01.log -set diodeex01_log.set
quit
```



Computational Electronics



## (B) Bipolar Junction Transistor Simulation

go atlas

TITLE Bipolar Gummel plot and IC/VCE with constant IB  
# Silvaco International 1992, 1993, 1994

### # STRUCTURE SPECIFICATION PART:

mesh

x.m l=0 spacing=0.15  
x.m l=0.8 spacing=0.15  
x.m l=1.0 spacing=0.03  
x.m l=1.5 spacing=0.12  
x.m l=2.0 spacing=0.15

y.m l=0.0 spacing=0.006  
y.m l=0.04 spacing=0.006  
y.m l=0.06 spacing=0.005  
y.m l=0.15 spacing=0.02  
y.m l=0.30 spacing=0.02  
y.m l=1.0 spacing=0.12

region num=1 silicon

electrode num=1 name=emitter left length=0.8  
electrode num=2 name=base right length=0.5 y.max=0  
electrode num=3 name=collector bottom



Computational Electronics

doping reg=1 uniform n.type conc=5e15  
doping reg=1 gauss n.type conc=1e18 peak=1.0 char=0.2  
doping reg=1 gauss p.type conc=1e18 peak=0.05 junct=0.15  
doping reg=1 gauss n.type conc=5e19 peak=0.0 junct=0.05 x.right=0.8  
doping reg=1 gauss p.type conc=5e19 peak=0.0 char=0.08 x.left=1.5

### # MATERIALS MODELS SPECIFICATION

# set bipolar models  
models conmob fldmob consrh auger print  
contact name=emitter n.poly surf.rec

### # NUMERICAL SOLUTION PART – SOLUTION SPECIFICATION

#### # Initial solution part

solve init  
save outf=bjtex04\_0.str  
tonyplot bjtex04\_0.str -set bjtex04\_0.set

#### # Gummel plot

method newton autonr trap  
solve vcollector=0.025  
solve vcollector=0.1  
solve vcollector=0.25 vstep=0.25 vfinal=2 name=collector  
solve vbase=0.025  
solve vbase=0.1  
solve vbase=0.2  
log outf=bjtex04\_0.log  
solve vbase=0.3 vstep=0.05 vfinal=1 name=base  
tonyplot bjtex04\_0.log -set bjtex04\_0\_log.set



Computational Electronics

```

#C/VCE with constant IB
#Ramp Vb
log off
solve init
solve vbase=0.025
solve vbase=0.05
solve vbase=0.1 vstep=0.1 vfinal=0.7 name=base

#Switch to current boundary conditions
contact name=base current

#Ramp IB and save solutions
solve ibase=1.e-6
save outf=bjtex04_1.str master
solve ibase=2.e-6
save outf=bjtex04_2.str master
solve ibase=3.e-6
save outf=bjtex04_3.str master
solve ibase=4.e-6
save outf=bjtex04_4.str master
solve ibase=5.e-6
save outf=bjtex04_5.str master

#Load in each initial guess file and ramp VCE
load inf=bjtex04_1.str master
log outf=bjtex04_1.log
solve vcollector=0.0 vstep=0.25 vfinal=5.0 name=collector

load inf=bjtex04_2.str master
log outf=bjtex04_2.log
solve vcollector=0.0 vstep=0.25 vfinal=5.0 name=collector

```

```

load inf=bjtex04_3.str master
log outf=bjtex04_3.log
solve vcollector=0.0 vstep=0.25 vfinal=5.0 name=collector

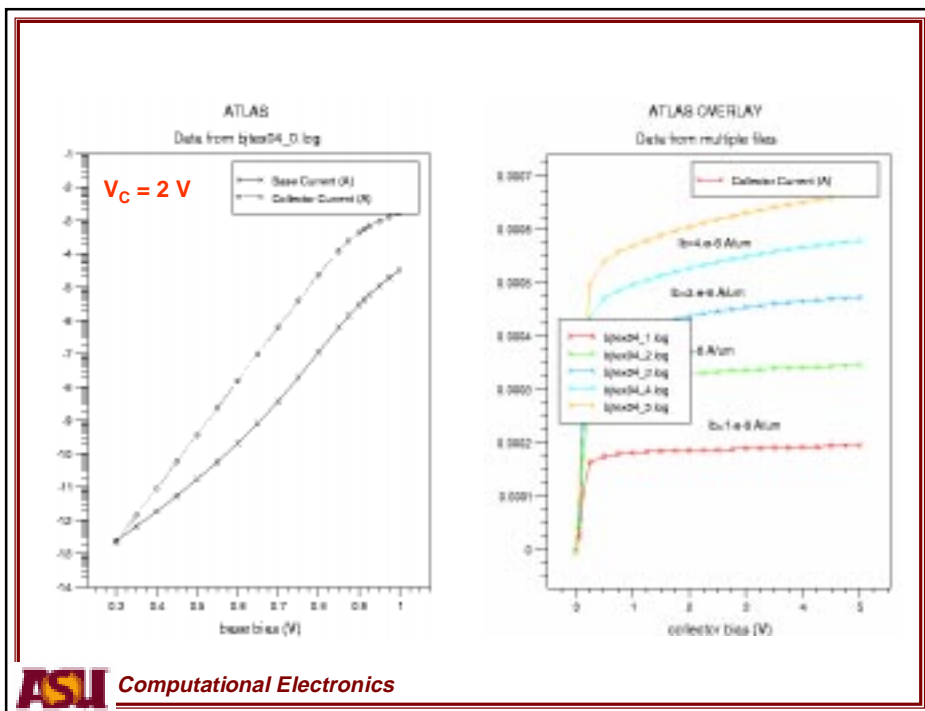
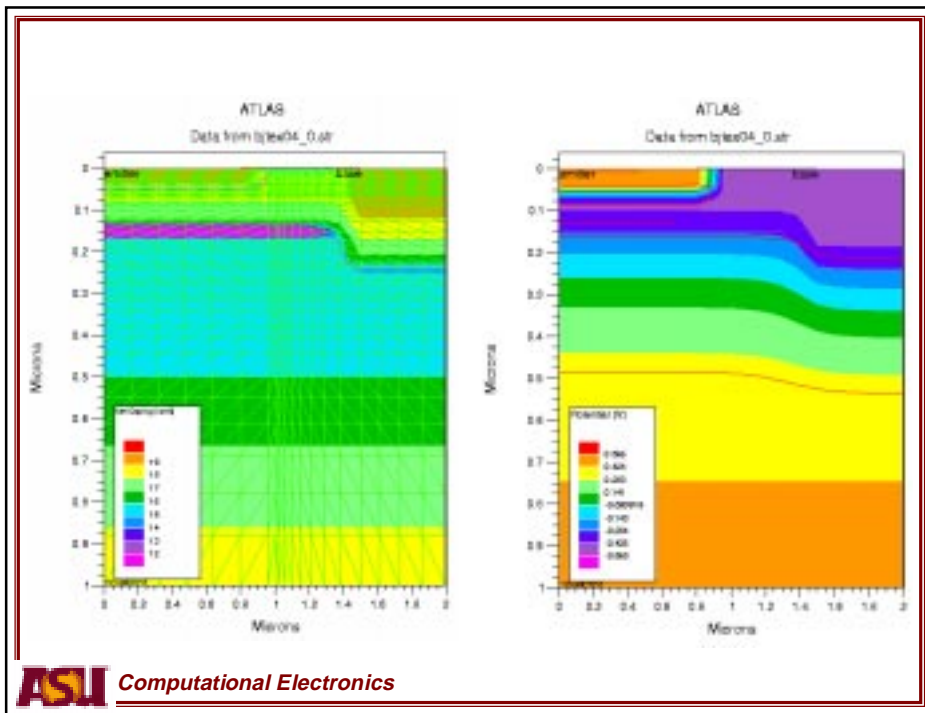
load inf=bjtex04_4.str master
log outf=bjtex04_4.log
solve vcollector=0.0 vstep=0.25 vfinal=5.0 name=collector

load inf=bjtex04_5.str master
log outf=bjtex04_5.log
solve vcollector=0.0 vstep=0.25 vfinal=5.0 name=collector

# RESULTS DISPLAY USING TONYPLOT
# plot results
tonyplot -overlay bjtex04_1.log bjtex04_2.log bjtex04_3.log bjtex04_4.log
bjtex04_5.log -set bjtex04_1_log.set

# PROGRAM TERMINATION
quit

```



### (C) MOSFET Simulation

#### **# Setting ATHENA as a default simulator**

```
go athena
#
# there is a part of the program that is missing at the moment.
# It includes a set of commands for the actual process
# simulation that leads to the obtained doping profiles
#
# the stretch command is used to redefine the drawn
# length of the poly using the cd value ....
#
```

```
set cd=0.8
```

#### **# Mesh specification part**

```
line x loc=-0.5 spac=0.1
line x loc=-0.4 spac=0.05
line x loc=-0.2 spac=0.01
line x loc=0.0 spac=0.1
line x loc=0.2 spac=0.01
line x loc=0.4 spac=0.05
line x loc=0.5 spac=0.1
#
line y loc=0.0 spac=0.001
line y loc=0.2 spac=0.005
line y loc=0.5 spac=0.05
line y loc=0.8 spac=0.15
line y loc=5 spac=1
line y loc=10 spac=5
#
```



Computational Electronics

#### **# Process simulation part**

```
# (part that is missing at the moment)
```

```
#
```

#### **# A set of EXTRACT statements that allow extraction of various device design parameters**

```
# Extract the gate oxide thickness
extract name="gateox" thickness oxide mat.ocno=1 x.val=0.05
# extract final S/D Xj...
extract name="nxj" xj silicon mat.ocno=1 x.val=(-("$cd"/2+0.2) junc.ocno=1
# extract the long chan Vt...
extract name="n1dvt" 1dvt ntype vb=0.0 qss=1e10 x.val=0.0
# extract a curve of conductance versus bias...
extract start material="Polysilicon" mat.ocno=1 \
    bias=0.0 bias.step=0.2 bias.stop=2 x.val=0.0
extract done name="sheet cond v bias" \
    curve(bias,1dn.conduct material="Silicon" mat.ocno=1 region.ocno=1)\
    outfile="extract.dat"
# extract the N++ regions sheet resistance...
extract name="n++ sheet rho" sheet.res material="Silicon" mat.ocno=1 x.val=(-("$cd"/2+0.3)
region.ocno=1
# extract the sheet rho under the spacer, of the LDD region...
extract name="ldd sheet rho" sheet.res material="Silicon" \
    mat.ocno=1 x.val=(-("$cd"/2+0.05) region.ocno=1
# extract the surface conc under the channel...
extract name="chan surf conc" surf.conc impurity="Net Doping" \
    material="Silicon" mat.ocno=1 x.val=0.0
```

#### **# Electrodes definition part**

```
#
electrode name=gate x=0.0 y=0.1
electrode name=source x=(-("$cd"/2+0.2)
electrode name=drain x=("$cd"/2+0.2)
electrode name=substrate backside
```



Computational Electronics

```

# Saving and plotting the structure
structure outfile=mos1ex15_0.str
tonyplot mos1ex15_0.str -set mos1ex15_0.set

# Setting ATLAS as default simulator
##### Vt Test : Returns Vt, Beta and Theta #####
go atlas

# set material models
models cvt srh print
contact name=gate n.poly
interface qf=3e10

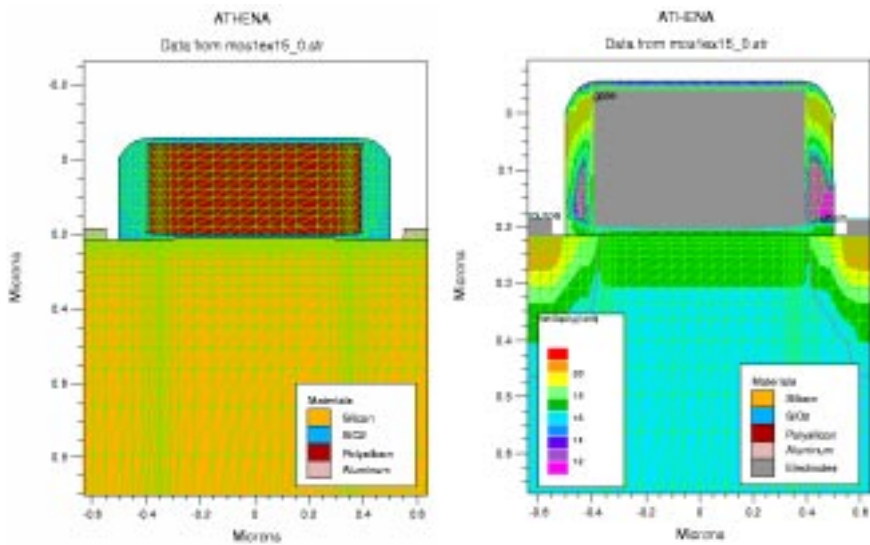
# Bias the drain a bit...
solve vdrain=0.0 vstep=0.025 vfinal=0.1 name=drain

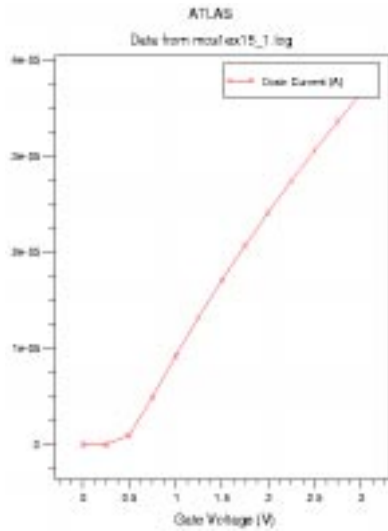
# Ramp the gate
log out=mos1ex15_1.log master
solve vgate=0 vstep=0.25 vfinal=3.0 name=gate

# plot results
tonyplot mos1ex15_1.log

# extract device parameters
extract init inf="mos1ex15_1.log"
extract name="nvt" (xintercept(maxslope(curve(abs(v."gate"),abs(i."drain")))) \
- abs(ave(v."drain"))/2.0)
extract name="nbeta" slope(maxslope(curve(abs(v."gate"),abs(i."drain")))) \
* (1.0/abs(ave(v."drain")))
extract name="ntheta" ((max(abs(v."drain")) * $"nbeta")/max(abs(i."drain"))) \
- (1.0 / (max(abs(v."gate")) - ($"nvt")))
quit

```





**Parameters stored in the *results.final* file:**

**gateox**=99.9226 angstroms (0.00999226 um)  
X.val=0.05

**nxj**=0.245228 um from top of first Silicon layer  
X.val=-0.6

**n1 dvt**=0.472837 V X.val=0

WARNING: x value is out of bounds of structure  
(min=-0.65, max=0.65)

defaulting to left hand side @ x=-0.585

**n++ sheet rho**=28.8262 ohm/square X.val=-0.585

**idd sheet rho**=1734.4 ohm/square X.val=-0.45

**chan surf conc**=3.14746e+16 atoms/cm<sup>3</sup>  
X.val=0

**nvt**=0.411988

**nbeta**=0.000170464

**ntheta**=0.0808717