Sandia
National
Laboratories

Charon User Manual: v. 2.2 (revision1)

Lawrence Musson, Gary Hennigan, Xujiao Gao, Richard Humphreys, Mihai Negoita, Andy Huang

ABSTRACT

This manual gives usage information for the Charon semiconductor device simulator. Charon was developed to meet the modeling needs of Sandia National Laboratories and to improve on the capabilities of the commercial *TCAD* simulators; in particular, the additional capabilities are running very large simulations on parallel computers and modeling displacement damage and other radiation effects in significant detail.

The parallel capabilities are based around the *MPI* interface which allows the code to be ported to a large number of parallel systems, including linux clusters and proprietary "big iron" systems found at the national laboratories and in large industrial settings.

ACKNOWLEDGMENT

Thanks to everyone that contributed to the project and this document.

Revision	Date	Author(s)	Description
2.2 Rev 1	May 2022	XG	§1 Equations and discretizations updates
2.2 Rev 1	May 2022	GH, LM	§2 Build and install updates
2.2 Rev 1	May 2022	XG	§4.1 Allow sinusoidal and trapzoidal voltages at an Ohmic contact
2.2 Rev 1	May 2022	MN	§4.2 Schottky contacts
2.2 Rev 1	May 2022	XG	§4.3 Allow linear voltage ramping at a gate contact
2.2 Rev 1	May 2022	MN	§5.2 Initial guess for insulators
2.2 Rev 1	May 2022	MN	§8.5 Dynamic traps recombination
2.2 Rev 1	May 2022	XG	§8.7 Read in multiple time-dependent optical generation files
2.2 Rev 1	May 2022	XG	§8.8 Local band-to-band tunneling models
2.2 Rev 1	May 2022	MN	§10 Mole-fraction dependent materials
2.2 Rev 1	May 2022	MN	§11.1 Density gradient quantum model
2.2 Rev 1	May 2022	AH	§13 Harmonic balance
2.2 Rev 1	May 2022	MN	§14.2.4 Kimpton total ionizing dose model
2.2 Rev 1	May 2022	XG	§16.7 Read in multiple doping files and Gauss decays for uniform doping
2.2 Rev 1	May 2022	LM	§17 Charon analysis with Dakota

Revision	Date	Author(s)	Description
2.1 Rev 1	March 2020	LM, GH, XG, MN, AH, RH	Initial version

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The Charon semiconductor device simulator is based on a multi-physics code for simulating general transport-reaction phenomena. The focus of this manual is on Charon's capabilities for modeling semiconductor devices. The semiconductor modeling capability in Charon was developed to work in a manner similar to other *TCAD* codes such as Atlas™ and Sentarus Device™. Additionally Charon supports massively parallel execution.

Charon allows to solve different sets of partial differential equations (PDEs), incuding the Poisson/Laplace equation, the drift-diffusion continuity equations for electrons/holes/ions, and the lattice temperature equation. Some of the PDEs can be solved individually or coupled together. In this chapter, we highlight the equations and relevant physics available in Charon.

1.1. Nonlinear Poisson Equation

The first important equation for semiconductor device modeling is the nonlinear Poisson (NLP) equation. Solving the NLP equation allows us to understand the electrostatic potential and field profiles in a semiconductor device under thermal equilibrium, i.e., zero current flow.

The famous electrostatic Poisson equation is given by

$$-\nabla \cdot (\varepsilon_0 \varepsilon_r \nabla \psi) = q\left(p - n + N_D^+ - N_A^-\right). \tag{1}$$

where

- ψ is the electrostatic potential.
- n is the electron concentration/density.
- p is the hole concentration/density.
- N_D^+ is the ionized donor density.
- N_A^- is the ionized acceptor density.
- ε_0 is the vacuum permittivity and $\varepsilon_0 = 8.8542 \times 10^{-14}$ C/(V.cm).
- ε_r is the relative permittivity or dielectric constant of a material.

Electron and hole densities can be computed from the electron and hole quasi-Fermi energy levels and the conduction and valence band edges. If Boltzmann statistics is assumed, the carrier densities are given as

$$n = N_C \exp\left(\frac{E_{Fn} - E_C}{k_B T}\right),$$
$$p = N_V \exp\left(\frac{E_V - E_{Fp}}{k_B T}\right), \tag{2}$$

where

- N_C is the effective density of states in the conduction band.

- N_V is the effective density of states in the valence band.

- E_C is the conduction band edge.

- E_V is the valence band edge.

- E_{Fn} is the electron quasi-Fermi energy level.

- E_{Fp} is the hole quasi-Fermi energy level.

- k_B is the Boltzmann constant and $k_B = 8.6173 \times 10^{-5}$ eV/K.

- T is the lattice temperature in Kelvin.

When Fermi-Dirac statistics is used, the carrier densities expressions become

$$n = N_C F_{1/2} \left(\frac{E_{Fn} - E_C}{k_B T} \right),$$
$$p = N_V F_{1/2} \left(\frac{E_V - E_{Fp}}{k_B T} \right), \tag{3}$$

where $F_{1/2}$ is the Fermi-Dirac integral of 1/2 order [1]. Under thermal equilibrium, we have $E_{Fn} = E_{Fp} = $ constant and we choose to set $E_{Fn} = E_{Fp} = 0$, which serves as the zero energy level. When solving the Poisson equation (1), since any potential with a constant offset satisfies the equation, we need to know what potential is solved for. Following the discussions given in Ref. [1], we have

$$E_C = -q(\psi - \psi_{ref}) + \chi_{eff},$$
$$E_V = E_C - E_{g,eff} = -q(\psi - \psi_{ref}) + \chi_{eff} - E_{g,eff}, \tag{4}$$

where

- q is the elemental charge.

- χ_{eff} is the effective electron affinity.

- $E_{g,eff}$ is the effective band gap.

- ψ_{ref} is a constant reference potential. It is chosen to be the difference between the vacumm potential and the intrinsic Fermi potential of a reference material specified in a Charon simulation.

Therefore, under thermal equilibrium, we solve the following nonlinear Poisson equation for Boltzmann statistics

$$-\nabla \cdot (\varepsilon_0 \varepsilon_r \nabla \psi) = q \left[N_V \exp \left(\frac{E_V}{k_B T} \right) - N_C \exp \left(\frac{-E_C}{k_B T} \right) + N_D^+ - N_A^- \right]. \tag{5}$$

For Fermi-Dirac statistics, we solve this equation

$$-\nabla \cdot (\varepsilon_0 \varepsilon_r \nabla \psi) = q \left[N_V F_{1/2} \left(\frac{E_V}{k_B T} \right) - N_C F_{1/2} \left(\frac{-E_C}{k_B T} \right) + N_D^+ - N_A^- \right].$$ (6)

For a dielectric material, the Laplace equation with a fixed charge profile is solved

$$-\nabla \cdot (\varepsilon_0 \varepsilon_r \nabla \psi) = \rho_{fix},$$ (7)

where ρ_{fix} is a fixed charge profile specified by a user.

1.2.   Isothermal Drift Diffusion Model

Charon supports several carrier transport models. The fundamental components of these models are the coupled Poisson and electron and hole continuity equations for semiconductor devices as given below:

$$-\nabla \cdot (\varepsilon_0 \varepsilon_r \nabla \psi) = q \left(p - n + N_D^+ - N_A^- \right),$$

$$q \frac{\partial n}{\partial t} = \nabla \cdot \mathbf{J}_n - qR,$$

$$q \frac{\partial p}{\partial t} = -\nabla \cdot \mathbf{J}_p - qR,$$ (8)

where

- R is the net recombination rate.

- $\mathbf{J}_n$ is the electron current density.

- $\mathbf{J}_p$ is the hole current density.

- t is the time in seconds.

The transport models differ in the expressions to compute the current densities. For isothermal drift-diffusion model, we follow the current density formulation given in Ref. [2],

$$\mathbf{J}_n = qn\mu_n \mathbf{F}_{n,eff} + qD_n \nabla n,$$
$$\mathbf{J}_p = qp\mu_p \mathbf{F}_{p,eff} - qD_p \nabla p,$$ (9)

where

- $\mathbf{F}_{n,eff}$ is the electron effective field.

- $\mathbf{F}_{p,eff}$ is the hole effective field.

- μ_n is the electron mobility.

- μ_p is the hole mobility.

- D_n is the electron diffusion coefficient.

- D_p is the hole diffusion coefficient.

In Charon, the effective electric fields are computed using [2]

$$\mathbf{F}_{n,eff} = \nabla \left(E_i - \frac{\Delta E_g}{2} - \frac{k_B T}{2} \ln(\gamma_n \gamma_p) \right),$$

$$\mathbf{F}_{p,eff} = \nabla \left(E_i + \frac{\Delta E_g}{2} + \frac{k_B T}{2} \ln(\gamma_n \gamma_p) \right), \tag{10}$$

where

- E_i is the intrinsic Fermi energy level.

- ΔE_g is the band gap narrowing due to high doping.

- γ_n is the electron degeneracy factor.

- γ_p is the hole degeneracy factor.

The ΔE_g term contributes only when a band gap narrowing model is specified. E_i is defined as

$$E_i = q\psi_{ref} - \chi_{eff} - q\psi - \frac{E_{g,eff}}{2} - \frac{k_B T}{2} \ln \left(\frac{N_C \gamma_n}{N_V \gamma_p} \right). \tag{11}$$

The generacy factors γ_n and γ_p are given by

$$\gamma_n = \frac{F_{1/2}(\eta_n)}{\exp(\eta_n)} \quad \text{where} \quad \eta_n = \frac{E_{Fn} - E_C}{k_B T},$$

$$\gamma_p = \frac{F_{1/2}(\eta_p)}{\exp(\eta_p)} \quad \text{where} \quad \eta_p = \frac{E_V - E_{Fp}}{k_B T}. \tag{12}$$

The degeneracy factors account for the Fermi-Diract statistics which is needed for high doping. All the terms involving the degeneracy factors have non-zero contributions only when the Fermi-Dirac statistics is enabled and a high doping is used.

1.3. Non-isothermal Drift Diffusion Model

Another transport model supported in Charon is the non-isothermal drift diffusion model that allows to simulate current induced self-heating responses in certain semiconductor devices. This non-isothermal model solves the Poisson equation, the electron and hole continuity equations, together with the lattice heat equation given below.

$$-\nabla \cdot (\varepsilon_0 \varepsilon_r \nabla \psi) = q \left(p - n + N_D^+ - N_A^- \right),$$

$$q \frac{\partial n}{\partial t} = \nabla \cdot \mathbf{J}_n - qR,$$

$$q \frac{\partial p}{\partial t} = -\nabla \cdot \mathbf{J}_p - qR,$$

$$\frac{\partial (c_L T)}{\partial t} - \nabla \cdot (\kappa_L \nabla T) = H. \tag{13}$$

18

where

- c_L is the lattice heat capacity.

- κ_L is the lattice thermal conductivity.

- H is the heat generation rate.

Within this non-isothermal transport model, the current densities are defined as

$$\begin{aligned}
\mathbf{J}_n &= qn\mu_n\mathbf{F}_{n,eff} + qD_n\nabla n + n\mu_n k_B \nabla T, \\
\mathbf{J}_p &= qp\mu_p\mathbf{F}_{p,eff} - qD_p\nabla p - p\mu_p k_B \nabla T.
\end{aligned} \tag{14}$$

The ∇T term in the current density expression accounts for the temperature gradient contribution to the current. For the heat generation rate H, there exist many different models [3] [4] [5] [6]. Charon currently supports the following simple model

$$H = \mathbf{J}_n \cdot \mathbf{F}_{n,eff} + \mathbf{J}_p \cdot \mathbf{F}_{p,eff} + R(E_{g,eff} + 3k_B T). \tag{15}$$

Here the first two terms account for the Joule heating effect and the third term includes contribution from possible electron-hole recombination processes. R denotes the net recombination rate.

The lattice heat equation in Eq. (13) can be solved independently to quickly obtain temperature spatial distribution for any given heat source. In this case, the heat generation rate H is provided by a user.

1.4.   Isothermal Memristor Model

Charon provides two types of transport models to simulate oxide memristor devices. This section highlights the isothermal memristor model. This model solves the Poisson equation, the electron and hole continuity equations, and the ion or vacancy continuity equation at a given lattice temperature. It is based on the work by Strukov *et al.* [7]. These equations are given below

$$\begin{aligned}
-\nabla \cdot (\varepsilon_0\varepsilon_r \nabla \psi) &= q\left(p - n + N_D^+ - N_A^- + z_{i/v}N_{i/v}\right), \\
q\frac{\partial n}{\partial t} &= \nabla \cdot \mathbf{J}_n - qR, \\
q\frac{\partial p}{\partial t} &= -\nabla \cdot \mathbf{J}_p - qR, \\
qz_{i/v}\frac{\partial N_{i/v}}{\partial t} &= -\nabla \cdot \mathbf{J}_{i/v}.
\end{aligned} \tag{16}$$

where

- $N_{i/v}$ is the ion or vacancy density.

- $\mathbf{J}_{i/v}$ is the ion or vacancy current density.

- $z_{i/v}$ is an integer number, which can be either positive or negative. $z_{i/v}q$ denotes the charge of ions or vacancies.

Since vacancies are mainly concerned in oxide memristor devices, we will focus discussions on vacancies in the remainder of this section and also next secton, Sec. 1.5. However, the discussions are equally applicable to ions with the subscript v replaced by i. Within this model, the current densities are given by

$$
\begin{aligned}
\mathbf{J}_n &= qn\mu_n\mathbf{F}_n + qD_n\nabla n, \\
\mathbf{J}_p &= qp\mu_p\mathbf{F}_p - qD_p\nabla p, \\
\mathbf{J}_v &= |z_v|qN_v\mu_v\mathbf{F}_v - z_vqD_v\nabla N_v.
\end{aligned}
\tag{17}
$$

where

- μ_v is the vacancy mobility.

- D_v is the vacancy diffusion coefficient. D_v and μ_v follow the Einstein relation, $\frac{D_v}{\mu_v} = \frac{k_BT}{q}$.

The electric fields in this model are simply defined as

$$
\begin{aligned}
F_n &= -\nabla\psi - \frac{1}{2}\nabla(\Delta E_g), \\
F_p &= -\nabla\psi + \frac{1}{2}\nabla(\Delta E_g), \\
F_v &= -\nabla\psi
\end{aligned}
\tag{18}
$$

1.5.　　　Non-isothermal Memristor Model

The non-isothermal memristor model in Charon solves the Poisson equation, the electron and hole continuity equations, the ion or vacancy continuity equation, together with the lattice heat equations. Some details of the model and its application are described by Gao *et al.* [8] [9]. The equations that are solved for this model are given by

$$
\begin{aligned}
-\nabla\cdot(\varepsilon_0\varepsilon_r\nabla\psi) &= q\left(p - n + N_D^+ - N_A^- + z_vN_v\right), \\
q\frac{\partial n}{\partial t} &= \nabla\cdot\mathbf{J}_n - qR, \\
q\frac{\partial p}{\partial t} &= -\nabla\cdot\mathbf{J}_p - qR, \\
qz_v\frac{\partial N_v}{\partial t} &= -\nabla\cdot\mathbf{J}_v, \\
\frac{\partial(c_LT)}{\partial t} - \nabla\cdot(\kappa_L\nabla T) &= H.
\end{aligned}
\tag{19}
$$

where

- N_v is the vacancy density.

- $\mathbf{J}_v$ is the vacancy current density.

- z_v is an integer number, which can be either positive or negative.

Note the Poisson equation given here is somewhat different from the one in Ref. ([8]). This is because the Poisson equation for the non-isothermal memristor model in Charon was later reformulated to be consistent with the isothermal drift-diffusion model in Sec. 1.2. The current densities are defined as [8]

$$
\begin{aligned}
\mathbf{J}_n &= qn\mu_n\mathbf{F}_n + qD_n\nabla n + n\mu_n k_B\nabla T, \\
\mathbf{J}_p &= qp\mu_p\mathbf{F}_p - qD_p\nabla p - p\mu_p k_B\nabla T, \\
\mathbf{J}_v &= |z_v|qN_v\mathbf{v}_v - z_v qD_v\nabla N_v - z_v qD_v S_v N_v\nabla T.
\end{aligned}
\tag{20}
$$

Here

- $\mathbf{v}_v$ is the vacancy velocity.

- S_v is the vacancy Soret coefficient. More information on $\mathbf{v_v}$ and S_v will be discussed in Sec. 7.

The heat generation in Eq. (19) can be modeled at different levels of sophistication. Since it is generally accepted that Joule heating is the most dominant heat generation source in memristive devices, Charon models the heat generation as

$$
H = \mathbf{J}_n\cdot\mathbf{F}_n + \mathbf{J}_p\cdot\mathbf{F}_p + \mathbf{J}_v\cdot\mathbf{F}_v + R(E_{g,eff} + 3k_BT).
\tag{21}
$$

This expression includes Joule heating due to all charge carriers and contribution from possible electron-hole recombination processes.

1.6.   Discretization Methods

To numerically solve the PDEs described in the above sections, we need to discretize the equations. Charon supports several discretization methods. The basic discretization scheme is the finite element method (FEM) with the Streamline Upwinding Petrov-Galerkin (SUPG) stabilization [10] denoted as FEM-SUPG. With FEM-SUPG, the current densities are not modified and the stabilization terms are added to the Galerkin residuals. The most popular discretization method for solving the semiconductor PDEs is the Scharfetter-Gummel (SG) approach. Charon implements the generalized SG discretization by Bochev *et al.* [11] denoted as CVFEM-SG. With CVFEM-SG, the current densities are modified according to the SG approach, and there is no stabilization term in the equation residuals. The third discretization method implemented in Charon is the Exponentially Fitted Flux Petrov-Galerkin (EFFPG) approach [12] within the FEM framework denoted as FEM-EFFPG. The FEM-EFFPG discretization is a method between FEM-SUPG and CVFEM-SG, which implements the SG current density expressions in the FEM framework. This eliminates the need of finding optimal stabilization parameters that are often needed for FEM-SUPG to work well. It also allows the code implementation to be fully consistent with the FEM framework. The three discretization methods are not implemented for all

Table 1-1 Available discretization methods for transport models.

Transport Model	Available Methods	Preferred Method
Laplace equation, Sec. 1.1	FEM-SUPG, CVFEM-SG	FEM-SUPG or CVFEM-SG
Nonlinear Poisson equation, Sec. 1.1	FEM-SUPG, CVFEM-SG	FEM-SUPG or CVFEM-SG
Isothermal drift diffusion model, Sec. 1.2	FEM-SUPG, CVFEM-SG, FEM-EFFPG	CVFEM-SG
Non-isothermal drift diffusion model, Sec. 1.3	FEM-SUPG, FEM-EFFPG	FEM-EFFPG
Lattice heat model, Sec. 1.3	FEM-SUPG	FEM-SUPG
Isothermal memristor model, Sec. 1.4	FEM-SUPG	FEM-SUPG
Non-isothermal memristor model, Sec. 1.5	FEM-SUPG, FEM-EFFPG	FEM-EFFPG

the transport models. The transport models and corresponding implemented discretization methods are listed in Table 1-1, where the preferred method is also listed.

The transport model and corresponding discretization method in a Charon simulation can be specified via the `standard discretization type` line and the `discretization method` line in a physics block as shown below.

```
start phyiscs block {physicsBlockName}
      standard discretization type is {discType}
      discretization method is {discMethod}
      {other physics descriptions}
end physics block {physicsBlockName}
```

Available values for `discType` and `disMethod` are listed in Table 1-2 together with the corresponding transport models and discretization methods.

Table 1-2 Available *discType* and *discMethod* values and corresponding transport models and discretization methods

discType Value	*discMethod* Value	Transport Model	Discretization Method
laplace gfem	N/A	Laplace equation, Sec. 1.1	FEM-SUPG
laplace cvfem	N/A	Laplace equation, Sec. 1.1	CVFEM-SG
nlp or nlp gfem	N/A	Nonlinear Poisson equation, Sec. 1.1	FEM-SUPG
nlp cvfem	N/A	Nonlinear Poisson equation, Sec. 1.1	CVFEM-SG
drift diffusion gfem	N/A	Isothermal drift diffusion model, Sec. 1.2	FEM-SUPG
drift diffusion effpg	N/A	Isothermal drift diffusion model, Sec. 1.2	FEM-EFFPG
drift diffusion cvfem	N/A	Isothermal drift diffusion model, Sec. 1.2	CVFEM-SG
lattice gfem	N/A	Lattice heat model, Sec. 1.3	FEM-SUPG
ddion	N/A	Isothermal memristor model, Sec. 1.4	FEM-SUPG
ddlattice	FEM-SUPG	Non-isothermal drift diffusion model, Sec. 1.3	FEM-SUPG
ddlattice	FEM-EFFPG	Non-isothermal drift diffusion model, Sec. 1.3	FEM-EFFPG
ddionlattice	FEM-SUPG	Non-isothermal memristor model, Sec. 1.5	FEM-SUPG
ddionlattice	FEM-EFFPG	Non-isothermal memristor model, Sec. 1.5	FEM-EFFPG

2. INSTALLATION

Charon is currently in use on many unix and unix-like systems at Sandia National Laboratories, including multiple Linux distributions and custom operating systems on large parallel systems. Instructions for installing it on representative systems follow. Note that each system is often unique and this section is not meant to be exhaustive, only to give you the background knowledge necessary to successfully install the code. Contact the authors if you are having trouble installing on a specific system.

2.1. Prerequisites

Charon requires a modern compiler suite, including a C++ compiler that is compliant with the C++-14 standard. The GNU 7.x and Intel 19 compiler suites are two examples that have been successfully utilized to build Charon. Requirements of third party libraries are given in Table 2-1. Additionally, several utilities are required by either Charon itself or the Trilinos framework. Those are given in Table 2-2. Note that earlier or later versions of the packages and utilities listed in Tables 2-1 and 2-2 may work but there those versions were not tested with Charon. Also note that if at all possible the libraries and utilities in Table 2-1 and Table 2-2 should be installed via your OS. For example, on Ubuntu Linux `apt install netcdf` would install netcdf on your system alleviating the need to install it yourself from source.

Table 2-1 Packages and libraries required by Charon. Installation should be performed in the order as shown in the table.

Package Name	URL for Package	Minimum Version
OpenMPI	https://www.open-mpi.org	4.0.x
HDF5	https://www.hdfgroup.org/HDF5	1.10
boost	https://www.boost.org	1.71
netCDF	https://www.unidata.ucar.edu/software/netcdf/	4.7.x
TriBITS	https://tribits.org/	769f615fafb
Trilinos	https://trilinos.org	81e9581a3c5

To clone TriBITS, for example, do:

```
git clone https://github.com/TriBITSPub/TriBITS.git
git checkout 769f615fafb
```

Note that after TriBITS is cloned you must set the environemnt variable `TRIBITS_BASE_DIR` to point to the location into which it was cloned prior to invoking the Charon build script, `build_charon.py`.

Also note that Charon is tightly dependent on the Trilinos framework. At the present time the build of Charon is tightly integrated with the build of Trilinos and the two cannot be separated. It is therefore required that you obtain a copy of the *develop* branch of the Trilinos distribution.

Table 2-2 Utilities required to install and run Charon.

Utility Name	URL for Utility	Minimum Version
cmake	http://www.cmake.org	3.23.1
python	http://www.python.org	3.x
git	http://www.git-scm.com	2.36.1
cubit	http://cubit.sandia.gov	15.5

Once the prerequisites are available, the next step is to unpack the tarball that you obtained from the Charon team. On a Linux system a typical command to do this would be

```
tar xzvf charon-distrib.tar.gz
```

Next change to the `tcad-charon` directory and clone a copy of the Trilinos repository. Instructions for cloning a *development* version of Trilinos can be found at the URL referenced in Table 2-1. On a typical Linux system the commands to accomplish this are given by

```
cd tcad-charon
git clone https://github.com/Trilinos/Trilinos.git Trilinos
cd Trilinos
git checkout 81e9581a3c5
```

Note that the *git checkout* step above requires that you use the version that corresponds to the distribution of charon you obtained as specified in Table 2-1.

Once the distribution is unpacked and a clone of Trilinos has been performed the resulting directory structure should resemble the one shown in Figure 2-1.

```
tcad-charon
├── cmake
├── docs
├── scripts
│   └── build
├── src
├── test
│   ├── nightlyTests
│   └── nightlyTestsOUO
└── Trilinos
```
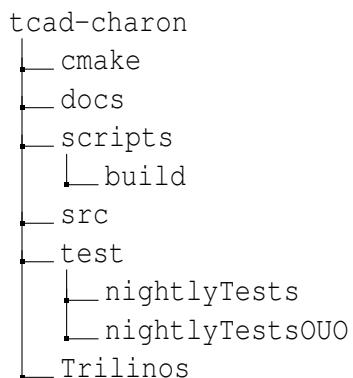
Figure 2-1 Partial directory tree resulting from unpacking of Charon tarball and subsequent cloning within that tree of the Trilinos repository.

2.2. Building Charon

2.2.1. *Using the Python Build Script*

The `scripts/build` subdirectory within `tcad-charon` contains utilities and data necessary to build Charon on various platforms. The python script `scripts/build/all/build_charon.py` is the preferred method for starting a Charon build. The purpose of the script is to generate an appropriate *cmake* command line and execute it. The script has a built-in help facility which should serve as a starting point for perfomring the installation. To see the built-in documentation invoke the script like

```
build_charon.py --help
```

The `--manual-page` argument will output detailed help for the script similar to what you would see in a Unix *man* page. The build script should be invoked within a dedicated build directory for the code, preferably a previously empty directory.

At a minimum you will need to create a `*.opts` file in

```
tcad-charon/scripts/build/<username>,
```

where `<username>` is replaced with your unix login ID. That file should be the name of your system, as returned by the *hostname* command, and contain the location of Boost and NetCDF on your system. A good example of the contents of that file can be found in

```
tcad-charon/scripts/build/all/attaway.opts.
```

Once the `build_charon.py` script has been successfully executed a simple invocation of *make* should compile the code, including the necessary Trilinos packages, and build a Charon executable. The resulting executable can be tested via the *cmake* utility *ctest*. Invoking the *ctest* command within the build directory will run a set of tests used internally as nightly regression tests and should give you some assurance that the resulting executable is working correctly.

Should the MPICH2 or IntelMPI *MPI* implementation be required then include the following CXX flag in the build script.

```
-DMPICH_IGNORE_CXX_SEEK
```

The Intel Math Kernel Library is a highly optimized math library, if Intel compilers are being used and the Intel Math Kernel Library is available then it can be linked in the build script using the following:

```
  TPL_ENABLE_MKL=ON
  MKL_LIBRARY_DIRS:FILEPATH="${MKLROOT}/lib/intel64"
  MKL_LIBRARY_NAMES:STRING="mkl_rt"
  TPL_ENABLE_BLAS=ON
  BLAS_LIBRARY_DIRS:FILEPATH="${MKLROOT}/lib/intel64"
  BLAS_LIBRARY_NAMES:STRING="mkl_rt"
  TPL_ENABLE_LAPACK=ON
```

```
LAPACK_LIBRARY_DIRS:FILEPATH="${MKLROOT}/lib/intel64"
LAPACK_LIBRARY_NAMES:STRING="mkl_rt"
```

If Sandia's optimization software, Dakota, is to be used in conjunction with Charon, support scripts may be built with Charon to ease the use of Dakota. See Chapter 17. To incorporate those scripts, add the following line to the opts file.

```
tcad-charon_ENABLE_DAKOTA_DRIVERS:BOOL=ON
```

3.   RUNNING CHARON

Once the code is compiled, and the resulting executable has been tested, the next step is running a simulation. There are numerous examples, including regression tests, that are included with the distribution. The recommended approach for a new user would be to start with one of the example problems and modify it to suit their particular problem.

3.1.   General Information

One thing to note about Charon is that it follows the example of many other finite-element like codes in that the mesh generation phase and the analysis phase are separate entities. Specifically Charon relies on a mesh file generated elsewhere. At Sandia this generally means that a program called *Cubit* (https://cubit.sandia.gov) is used to generate the mesh and output a file in *Exodus* format for input into Charon. Otherwise, any mesh generator that can produce an exodus formatted mesh should be usable by Charon.

3.2.   Conventions

Electric current is referenced as positive if it's leaving a device contact and negative if it's entering.

3.3.   The Charon Interpreter

All simulations in Charon require an input file to configure the run. The input file that Charon natively reads is a parameter list that contains information that completely configures the simulation including names of initial and output state files, physics, material models, solver configuration, etc. These parameter lists are highly structured, but are lengthy, complex and therefore fragile even for the simplest of simulations and require an experienced, knowledgeable user to create and maintain. The format of the input files is either XML or YAML which incurs additional challenges in visual acuity.

The Charon interpreter is a tool that the user can employ to make configuration of a charon simulation much simpler. The user-supplied input files to the interpreter are composed of simpler, more straight-forward language that is easier for the user to comprehend. Ultimately, the interpreter will map the interpreter input file into a parameter list that Charon understands.

3.3.1.   Invoking the Interpreter

The charon interpreter is invoked with arguments from the command line. To see the list of available arguments, invoke the interpreter with the –help argument,

```
charonInterpreter.py --help
```

When the interpreter is invoked with the -i or –input option plus a file name, the interpreter will create the parameter list in an XML format and terminate execution. For example, if the user has created an interpreter input called diode.inp and invokes

```
charonInterpreter.py -i diode.inp
```

the interpreter will generate the equivalent parameter list in an XML format and store it in a file called diode.inp.xml. Advanced users can do this if they wish to see, for example, the fine details of how the solvers are configured for their simulation. Otherwise, if the user simply wishes to run the simulation, the interpreter may be invoked as

```
charonInterpreter.py -i diode.inp --run
```

and this will generate the parameter list and execute Charon with it. Often times, TCAD simulations can be high in computational effort and it may be desirable to run Charon in parallel. Domain decomposition must be completed prior to executing Charon with the scripts that are included with the Trilinos libraries. In that instance, the interpreter may be used to execute Charon on, say, 4 processors via

```
charonInterpreter.py -i diode.inp --np 4 --run
```

The syntax of the input file is available in this user manual. However, for a quick reference, syntax can be obtained through the interpreter itself and will always be current with the installed version of the interpreter. The –syntax (-s) and –longsyntax (-S) arguments will return all available input lines the interpreter knows about. The latter will return a slightly more verbose description of each individual line. There is presently no way to use the interpreter to search for a specific syntax. There are easy ways to search the output, however. The output of the syntax help is plain text. The output may be redirected to a text file and opened in any editor. Another convenient way is to pipe the output through less in any Linux system as follows

```
charonInterpreter.py -S | less -i
```

This will allow scrolling through the syntax help. The keys /<searchTerm> will allow case insensitive searching through the entire syntax help.

When Charon is executed in parallel, the state file must be decomposed into a number of parts equal to the desired number of processors used to execute the run. The tools decomp (domain decomposition) and epu (reunification of decomposed state files) may be used manually to process state files into the desired composition. The decomposition may also be done through the interpreter so that the decomposition is done before Charon executes. For example, if the interpreter is executed as

```
charonInterpreter.py diode.inp --np 4 --run --decomp
```

the state file specified by the input file to be imported will be decomposed into 4 parts prior to the execution of Charon. If a decomposition of that size already exists, the interpreter will produce an error and stop with a warning. This is to safeguard against accidentally overwriting states already computed at that decomposition.

If it is desired to change the size of the decomposition of a previously computed run, this may be done through the interpreter by

```
charonInterpreter.py diode.inp --np 8 --run --resize_from_nprocs 4
```

This will take the 4 processor decomposition and redistribute it to 8 parts prior to executing the next run on 8 processors. This will likewise cause the interpreter to shut down with an error if a decomposition of size 8 already exists.

3.3.2.   *Configuring the Interpreter*

When Charon is installed in common locations for executable commands such as /usr/bin or any other location pointed to by the PATH variable, the interpreter will be able to find and use Charon without intervention. It is very common, however, that Charon is installed in a non-standard location. Less frequently, it is installed with something other than its default name (charon_mp.exe). Location and executable name may both be configured in the interpreter with environment variables. The environment variable, CHARON_EXECUTABLE_PATH set to the directory where the executable is installed will tell the interpreter where to find it. The variable CHARON_EXECUTABLE can be set if the name of the executable is different than charon_mp.exe.

3.3.3.   *Essential Parts of the Interpreter Input*

Examples of interpreter input files will be given in later sections of this manual. There are several hundred possible lines of input that can go into an input file. In practice, only a very small fraction of those will ever be included. And there is a small amount of information that is required of any Charon simulation no matter how small. The following shows essential, generic boilerplate of an interpreter input file.

```
import state file <stateFile.exo>

start output parameters
     output state file <stateFileOutput.exo>
end output parameters


start Physics Block Semiconductor
     geometry block is <regionName>
     standard discretization type is <equations to solve>
     material model is <materialModelName>
end Physics Block Semiconductor
```

```
start Material Block <materialBlockName>
     material name is <material>
     <Mobility Model>
     <Doping Parameters>
end Material Block siliconParameter

BC is ohmic for <contact name> on <region> fixed at <voltage>

initial conditions for <Field> in <region> is <IC>

start solver block
     <Solver configuration>
     <termination criteria>
end solver block
```

Each of these items will be covered in considerably more detail later in this manual. But the essential items are an input exodus file, an output exodus file, a block which describes the physics to be simulated in the regions of the model (there may be more than one), the material properties and models for each region, boundary conditions, initial conditions and a solver configuration.

Some general notes on the input file format:

- Comments are delineated with the "#" symbol.

- The keywords that make up the input file are not case senstive, but most values are case sensitve. A good example of this would be the name of the input file. It is case senstive.

- Other than enhancing readability, indentation is not significant.

- Blocks are always delineated via *start* and *end* statements.

- Text immediately following an *end* statement, on the same line, is not significant but can enhance readability.

3.3.4.   *Use of /include in the Interpreter Input File*

In nearly every application of Charon, multiple input files, with subtle but important differences between each, are required to arrive at the desired analysis. For example, a nonlinear Poisson solver must be completed before a drift-diffusion solve can be attempted. Or, if a MOSFET model is employed, one typically must sweep drain voltage up to a certain value before a meaningful gate sweep solver can be completed...or vice versa. A great deal of information can be identical between each solve, e.g. the doping and the mobility models. This can be particularly fragile during the calibration stage of assembling a predictive model because identical components must be changed identically in each of the input files.

Modifying all input files at once is most easily accomplished through the use of include files. One file may be created that contains mobility models or doping specifications that every input file can use. This is accomplished with the */include* directive in the input file. For example, if one creates the file *doping.inp*

```
        <Doping Parameters>
```

and the file *mobility.inp*

```
        <Mobility Model>
```

then the input file as demonstrated in the previous section can be change to read

```
import state file <stateFile.exo>

start output parameters
     output state file <stateFileOutput.exo>
end output parameters


start Physics Block Semiconductor
     geometry block is <regionName>
     standard discretization type is <equations to solve>
     material model is <materialModelName>
end Physics Block Semiconductor

start Material Block <materialBlockName>
     material name is <material>

     /inlcude mobility.inp

     /include doping.inp

end Material Block siliconParameter

BC is ohmic for <contact name> on <region> fixed at <voltage>

initial conditions for <Field> in <region> is <IC>

start solver block
     <Solver configuration>
     <termination criteria>
end solver block
```

and every input file in that same problem directory with the appropriate /inlcude directives will use the same specifications.

The use of /include is unlimited. Any line or block of text that is contiguous can be placed in a text file to be included in another file. In other words, it can likewise be used for an entire solver block, or some subsection thereof.

What is not allowed is nesting includes. So, for example, one cannot include a file in another include file. They are allowed only one deep to avoid making things overly confusing.

3.4. Example Problem: Nonlinear Poisson Simulation for a Pseudo One Dimensional Silicon PN Diode

A simple example of simulating a nonlinear-Poisson problem is given for a pseudo-1D PN diode with a step junction. This includes the mesh generation for the problem within cubit. An illustration of the geometry for this problem is given in Figure 3-1.

Charon can operate on either two- or three-dimensional geometries. It cannot be used for a true one-dimensional geometry. In this example the problem is one-dimensional because there is no variation in the geometry or the physics in the vertical direction. In a real problem with this one-dimensional nature you could minimize the problem size by keeping the vertical dimension one "element" thick. In this case however the vertical dimension has been extended to better illustrate the problem.
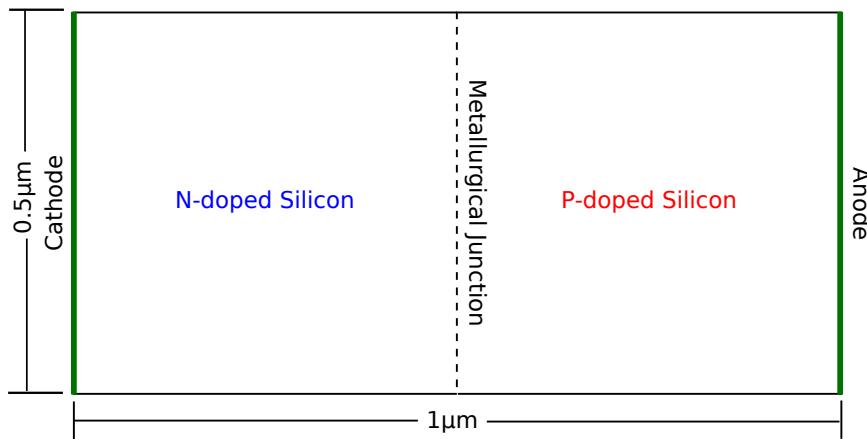
Figure 3-1 Geometry for a simple pseudo one-dimensional PN diode with a step junction. The metallurgical junction is at $0.5\mu m$**.**

3.4.1. Mesh Generation Using Cubit

You can utilize Cubit using two methods: the first is via the graphical-user interface (GUI) and the second is via journal commands. The second method will be utilized here. Read the Cubit documentation for further information. Note that the journal commands can be entered from within the GUI.

The contents of a Cubit journal file used to generate a mesh are given in Figure 3-2 and the resulting mesh is shown in Figure 3-3. If you save the contents of Figure 3-2 as a file you can generate the mesh using cubit via the command line with:

```
cubit -batch -nojou <filename>
```

where `<filename>` is the name of the file in which the journal commands are saved.

The file is heavily commented and should be self explanatory.Further information can be found on the Cubit website (see table 2-2 for the URL) which has documentation available directly on

```
1    graphics mode wireframe
2
3    # Create a three-dimensional volume. The two-dimensional diode will be
4    # created on a surface of this three-dimensional volume. By default
5    # Charon assumes the dimensions are in microns
6    create brick x 1.0 y 0.5 z 0.1
7
8    # This makes the coordinates of the resulting mesh all positive. This
9    # isn't required but can be useful for post-processing
10   move vertex 4 location 0 0 0
11
12   # These will be the contacts, anode and cathode. The names are used in
13   # the input file to distinguish them
14   sideset 1 curve 3
15   sideset 1 name "anode"
16
17   sideset 2 curve 1
18   sideset 2 name "cathode"
19
20   # "blocks" are typically regions of different materials or distinct
21   # regions of the device. For this simple problem we only have one
22   # region
23   block 1 surface 1
24   block 1 name "si"
25
26   # Quads are currently the preferred element type for Charon
27   # simulations.
28   block 1 element type quad4
29
30
31   # The interval specifications set how dense or coarse the
32   # discretization is
33
34   ## Long side
35   curve 2 4 interval 100
36
37   ## Short side (contacts)
38   curve 3 1 interval 10
39
40   # Generate, or "mesh", the problem geometry
41   mesh surface 1
42
43   # Create the output "exodus" file, or overwrite it if it already
44   # exists.
45   export mesh "pndiode.exo" dimension 2 overwrite
```

Figure 3-2 Cubit journal commands for the PN diode geometry illustrated in Figure 3-1.

the web page. The last statement in Figure 3-2, at line 45, instructs Cubit to output an Exodus mesh file named *pndiode.exo*. That file will be used as the input mesh file for the subsequent Charon simulation.

3.4.2. *Charon Input Deck for a NLP Simulation*

In most Charon drift-diffusion device simulations it is necessary to first solve a simpler problem, the results of which are then used as an initial guess for the more complex simulation. The nonlinear Poisson, or NLP, simulation is used for that purpose and is mathematically described in section 1.1.

34
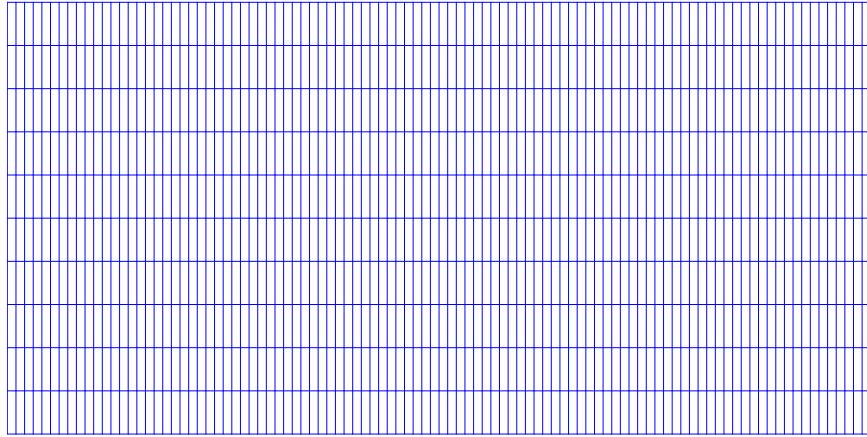
Figure 3-3 Mesh for PN diode generated with Cubit.

The Charon input file for the NLP simulation is shown in Listing 1 within Appendix D.1. Lines from that file will be discussed here in detail. More specifics about the syntax available via the input file can be found in Section 16.

The first block in the file is used to specify the input mesh.

```
1   # Name of the input exodus file, geometry only
2   import state file pndiode.exo
```

Line 1 contains a comment. Comments can increase the readability of the input files and should be generously utilized. Line 2 specifies that the file to be imported by Charon is named *pndiode.exo*. This file must exist and contain, at a minimum, the input mesh.

The next section specifies where and what output to perform.

```
5   start output parameters
6      output state file pndiode.nlp.exo
7   end output parameters
```

Line 5 illustrates the use of an input-file block. Such blocks encapsulate units of related specifications, in this case a block of text to specify `output parameters`, or how Charon will perform file output. A block begins with the `start` keyword, followed by the name of the input-file block, in this case `output parameters`, and ends with the `end` keyword. The `output parameters` text immediately following the `end` keyword is not required but can add to the readability of input-file blocks, particularly longer blocks containing a large number of lines.

Line 6 contains the name of the Exodus file to which output will be performed, *pndiode.nlp.exo*. If that file does not exist then it will be created. If the file exists then results from the simulation will be added to the existing file.

The next block is used to describe what physics is going to be solved during the simulation and how to associate material parameters with that physics.

```
9   start physics block semiconductor
10
11     # geometry block name IS case sensitive, note however that Cubit
```

35

```
12      # often downcases names prior to output so it is recommended
13      # that in Cubit you use all lower case for naming entities to
14      # avoid confusion.
15      geometry block is si
16
17      # The type of physics to be solved, in this case a nonlinear
18      # Poisson, or nlp, simulation will be performed
19      standard discretization type is nlp
20
21      # The name of the material model IS case sensitive. The name is
```

Line 9 is the start of the block, indicated with `start physics block`, followed by the name of the block, in this case `semiconductor`. Line 15 specifies the geometry block associated with this physics block. This name must correspond to an element block in the input Exodus file. Line 17 is the type of physics to simulate, in this case a `nlp`, or nonlinear Poisson, problem. Line 20 is the name of the material model for this physics. The name will be used to lookup the specifications for that material. Line 21 ends this input-file block. Note again that the only required keyword here is `end`, the name of the block is not required but can add to the readability of the input file.

The next block is a `material block` specification. The name of this block, `siliconParameter`, should be referenced in a physics block or it won't be utilized. In this case the reference is on line 20. Note that the material block name **is** case sensitive.

```
23      # in this input file.
24      material model is siliconParameter
25
26   end physics block semiconductor
27
28   # The material block where most material parameters for this
29   # simulation are set. It is specified in the physics block by it's
30   # name, siliconParameter.
31   start material block siliconParameter
32
33      # Material name IS case sensitive
34      material name is Silicon
35
36      # Simple, scalar, material property
37      relative permittivity = 11.9
```

The first non-comment line in this block, line 26, is the name of the material as used within Charon. This serves as a keyword by the code to look up default properties for that specific material, in this case silicon. The name **is** case sensitive and must match what is used within Charon for the specific material. Line 27 specifies the value of the `relative permittivity` material property for `Silicon`. Materials generally have default values for common material properties, such as relative permittivity, but those can be overridden via the input file, as in this case.

Lines 29–35 contain the doping specification for this problem. In this case the doping is a step junction with symmetric doping of donors and acceptors at $1.0 \times 10^{16}/cm^3$, the junction located at $x = 0.5 \mu m$ with the donor dopant on the left and the acceptor on the right in the x direction.

Line 37 ends the `material block` specification using the `end` keyword along with the optional `material block siliconParameter`.

The boundary conditions are specified on the following lines of the input file

```
39     # The doping for the diode
40     start step junction doping
```

Note that even though this example problem is a nonlinear Poisson simulation, an equilibrium problem, boundary conditions must be specified in the input file with the value of zero.

The specification of boundary conditions start with the `bc` keyword followed by the type of contact, `ohmic` in this case. Next the name of the relevant contact is given. Recall from Figure 3-1 that there were two contacts specified for the problem and they were named `anode` and `cathode`. These names were user specified during the construction of the mesh and completely arbitrary, although in this case they are physically significant.

Next, the name of `geometry block` which is adjacent to the boundary is given, in this case `si`. And finally the value of the actual boundary condition is given. In this case the specification is stated as a fixed, applied potential of zero volts. Since this is a NLP simulation and the only degree of freedom is electric potential, and there are only two contacts, no further boundary conditions are needed. Any boundary not specified, for example the top and bottom of the diode in this case, have the default zero-flux boundary condition applied to all degrees of freedom.

The initial conditions are set next via the line

```
42        donor concentration = 1e16
```

In this case the stanza simply says that `initial conditions` for the `ELECTRIC_POTENTIAL`, the only degree of freedom in this simulation, will be set using the an approximation for the equilibrium potential.

The last section in the input file deals with techniques for solving the sets of equations resulting from the simulation

```
44        dopant order is PN
45        direction is x
46     end step junction doping
```

Most settings are consolidated into a `solver pack`, in this case `solver pack 1`. A solver pack does have quantities that the user can override but it is generally best to make use of the default settings within a particular solver pack when possible. Other available solver packs, which simulation types they're relevant for, and how to override their settings will be covered in a subsequent section.

3.4.3. *Command Line Serial Execution for a NLP simulation*

The two items necessary to perform a Charon simulation are now in place, the input mesh file generated with Cubit, *pndiode.exo*, and an input file, *pndiode.nlp.inp*. The next step is to invoke the interpreter and have it perform the simulation via the Charon executable.

The basics of using *charonInterpreter.py* were covered in section 3.3.1. For the specific example here the invocation is

```
charonInterpreter.py -i pndiode.nlp.inp --run
```

If the example has never been executed before then a new Exodus output file named

```
pndiode.nlp-result.exo
```

will be created. If the file already exists then any results contained in the file will be overwritten.

This file contains a copy of the mesh as it existed in the input Exodus file as well as the results of the simulation, which in this case means the ELECTRIC_POTENTIAL, which was the degree of freedom in the NLP simulation. You can use a post-processing program such as Paraview (http://www.paraview.org) to visualize and post-process the results.

3.5. **Example Problem: IV Sweep for a Pseudo One Dimensional Silicon PN Diode**

This example problem builds on the example problem given in Section 3.4. The geometry is identical and given in Figure 3-1. The results of the NLP simulation described in that section will be used here as the initial guess for a full IV (current versus voltage) sweep of a drift diffusion simulation for the same diode. During an IV sweep the potential at one contact is incremented over a specified range while the potential at the other contact(s) is fixed at a specified value. At each bias point a simulation is performed and a scalar electric current is obtained. The voltage sweeping capability in Charon utilizes LOCA (https://trilinos.github.io/nox_and_loca.html) to perform the sweep.

3.5.1. *Charon Input Deck for an IV Sweep*

The input file for the problem will be described in detail here. Note that this section will primarily cover the differences between the input file used for the IV sweep. The detail on parameters common to this IV sweep problem and the NLP simulation in the previous section were covered in Section 3.4.2.

The Charon input file for the IV simulation described here is given in Listing 2 within Appendix D.2. The first block of lines in that file to be discussed are

```
1  # Name of the input exodus file, which includes the results for
2  # ELECTRIC_POTENTIAL as obtained from a previous NLP simulation
3  import state file pndiode.nlp.exo at index 1
```

As stated in the comments on lines 1 and 2, we want to use the ELECTRIC_POTENTIAL as obtained from the NLP simulation as an initial guess for the IV sweep. The results from the NLP simulation were output to the file named pndiode.nlp.exo at the first index in that file. That file will be used for both the input geometry as well as the ELECTRIC_POTENTIAL at the first results index in that file.

Next, the output will go to a new file created by Charon. That is specified via input-file block

```
5  # Output exodus file for results of this simulation
6  start output parameters
7    output state file pndiode.dd.iv.exo
8  end output parameters
```

Since no special output is specified the default output, which is generally outputs all the degrees of freedom, will be output at each voltage step in the IV sweep.

The physics block for this simulation is similar to that for the NLP with the exception of lines 20 and 28.

```
10  start physics block Semiconductor
11
12    # geometry block name IS case sensitive, note however that Cubit
13    # often downcases names prior to output so it is recommended that
14    # in Cubit you use all lower case for naming entities to avoid
15    # confusion.
16    geometry block is si
17
18    # The type of physics to be solved, in this case a nonlinear
19    # Poisson, or nlp, simulation will be performed
20    standard discretization type is drift diffusion gfem
21
22    # The name of the material model IS case sensitive. The name is
23    # used as a key for the associated material block, also contained
24    # in this input file.
25    material model is siliconParameter
26
27    # Turn on Schokley-Reed-Hall recombination
28    srh recombination is on
29
30  end physics block
```

Line 20 tells the code what type of physics simulation we will be performing and what discretization we are utilizing for that simulation. In this case we are doing a full drift-diffusion simulation using the Galerkin finite-element discretization, or gfem. Line 28 specifies that SRH, or Shockley-Read-Hall, recombination should be enabled.

This simulation is for the same device as that in the example NLP simulation in the previous section, however, we are solving a different set of physics here, drift-diffusion in this case, NLP in

the previous case. As such, the material properties specified in the material block starting on line 32 and ending on line 49 is identical to that of the NLP problem covered in the previous section.

```
32  # The material block where most material parameters for this
33  # simulation are set. It is specified in the physics block by it's
34  # name, siliconParameter.
35  start material block siliconParameter
36
37     # Material name IS case sensitive
38     material name is Silicon
39     relative permittivity = 11.9
40
41     start step junction doping
42        acceptor concentration =1.0e16
43        donor concentration =1.0e16
44        junction location = 0.5
45        dopant order is PN
46        direction is x
47        end step junction doping
48
49  end material block siliconParameter
```

Next the initial conditions are specified in lines 51 through 61.

```
51  # This is taken from the NLP file. Note that it is read from "index 1"
52  # as specified in the input file specification. In this case there is
53  # only a single result in that file.
54  initial conditions for ELECTRIC_POTENTIAL in si is exodus file
55
56  # The NLP simulation does not include a solution for the carrier
57  # densities, therefore some other type of estimate, in this case an
58  # equilibrium calculation, is used to obtain an initial guess for
59  # the carrier densities.
60  initial conditions for ELECTRON_DENSITY in si is equilibrium density
61  initial conditions for HOLE_DENSITY in si is equilibrium density
```

The first thing to note is that as specified in line 54 the initial condition for ELECTRIC_POTENTIAL is going to be read in from the exodus file that was specified as the input file in line 3. Additionally line 3 specifies that the result in variable index 1 in that file is to be used. In this case, since that file is the result of a NLP simulation, there is only a single variable index contained in the file, but in more complex cases you can specify the relevant variable index. In transient simulations a variable index corresponds to a time index. In an IV sweep simulation the variable index corresponds to the index associated with a particular value of the voltage during the sweep.

For the remaining degrees of freedom in the problem, ELECTRON_DENSITY and HOLE_DENSITY, an approximation will be used to generate an initial guess internally since those variables were not part of the NLP simulation. In this case lines 60 and 61 specify that the equilibrium density approximation be used for the initial guess.

The boundary conditions are set for the problem in lines 64 and 65 Next the initial conditions are specified in lines 51 through 61.

```
63  # Boundary condtions at the contacts.
64  bc is ohmic for cathode on si fixed at 0.0
65  bc is ohmic for anode on si swept from 0.0 to 1.0
```

The boundary condition for the cathode is specified in line 64 as an `ohmic` type contact and is going to have a fixed bias of 0 volts applied to it for the entirety of the sweep.

The boundary condition for the anode is specified in line 65 as the same type, `ohmic`, but in this case also specifies that the value of the bias to be applied is going to be swept from 0 to 1 volt.

How the voltage on the anode is going to be swept is controlled by lines 68 through 72 of the input file.

```
67  # Sweep parameter controls
68  start sweep options
69     initial step size = 0.02
70     minimum step size = 0.02
71     maximum step size = 0.02
72  end sweep options
```

The LOCA package which is used by Charon to perform parameter sweeps, like this voltage sweep, is capable of very sophisticated step control. However, in this case only a simple constant voltage step of 0.02*V* will be used.

As in the previous NLP example, the last section of the code deals with solver settings.

```
74  # Use a straightforward solver pack for this simulation.
75  start solver block
76    use solver pack 1
77  end solver block
```

Most settings are consolidated into a `solver pack`, in this case `solver pack 1`. A solver pack does have quantities that the user can override but it is generally best to make use of the default settings within a particular solver pack when possible. Other available solver packs, which simulation types they're relevant for, and how to override their settings will be covered in a subsequent section.

3.5.2.   *Command Line Serial Execution for an IV sweep*

The items necessary to perform the IV sweep are now in place, the file containing the initial guess and input mesh, `pndiode.nlp.exo` and the input file, `pndiode.iv.inp`. In order to run the problem simply invoke the interpreter like

```
charonInterpreter.py -i pndiode.iv.inp --run
```

An initial run, without having previously executed this command, a couple of results files will be generated

```
pndiode.dd.iv.exo
currents-loca.dat
```

The first file contains the results of the simulation for each voltage step. This includes the degrees of freedom at each voltage step, ELECTRIC_POTENTIAL, ELECTRON_DENSITY and HOLE_DENSITY, as well as the scalar electric current at the contacts. For convenience Charon also outputs scalar electric currents to a regular text file at each of the contacts for each of the voltage steps taken during the sweep.

For post-processing a visualization program such as Paraview (http://www.paraview.org) can be used to examine the exodus file. If only the currents are of interest a standard plotting package, such as gnuplot, can be used to plot the results. For this case the results are shown in Figure 3-4.
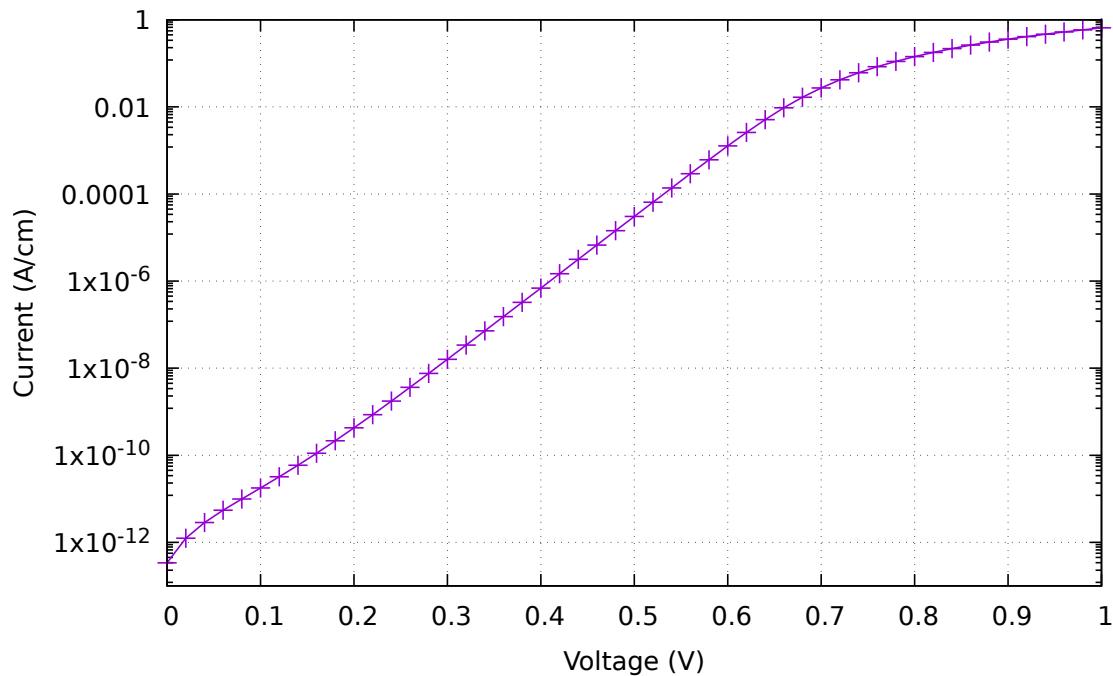
Figure 3-4 Results of IV sweep

4.   BOUNDARY CONDITIONS

4.1.   Ohmic Contacts

Boundary conditions for Ohmic contacts are derived assuming charge neutrality and equilibrium conditions and in relation to carrier statistics and dopant ionization. By imposing charge neutrality and equilibrium conditions:

$$
\begin{aligned}
n_0 - p_0 &= N_D^+ - N_A^- = C, \\
n_0 p_0 &= n_i^2
\end{aligned}
$$

(22)

the carrier densities and contact potential assuming fully ionized dopants and Maxwell-Boltzmann statistics can be solved for at the Ohmic contacts. For a n-type semiconductor they are given by:

$$
\begin{aligned}
n_0 &= \frac{C}{2} + \sqrt{\left(\frac{C}{2}\right)^2 + n_i^2}, \\
p_0 &= \frac{n_i^2}{n_0}, \\
\phi &= \frac{k_B T}{q} \ln(y) + \frac{1}{q}(E_{ref} - \chi) + V, \\
y &= \frac{C}{2N_C} + \sqrt{\left(\frac{C}{2N_c}\right)^2 + \frac{N_V}{N_C} \exp\left(-\frac{E_G}{k_B T}\right)}
\end{aligned}
$$

(23)

and for a p-type semiconductor by:

$$
\begin{aligned}
p_0 &= \frac{-C}{2} + \sqrt{\left(\frac{C}{2}\right)^2 + n_i^2}, \\
n_0 &= \frac{n_i^2}{p_0} \\
\phi &= \frac{-k_B T}{q} \ln(y) + \frac{1}{q}(E_{ref} - \chi - E_G) + V, \\
y &= -\frac{C}{2N_V} + \sqrt{\left(\frac{C}{2N_V}\right)^2 + \frac{N_C}{N_V} \exp\left(-\frac{E_G}{k_B T}\right)}
\end{aligned}
$$

(24)

where V is the voltage applied at the Ohmic contact and E_{ref} is the Fermi energy of the reference material and χ is the electron affinity in eV.

For Fermi Dirac statistics the second equilibrium condition in Eq.(22) rewrites as $n_0 p_0 = \gamma_n \gamma_p n_i^2$ where γ_n, γ_p are the electron and hole degeneracy factors. If the dopants are fully ionized, the

43

carrier densities and contact potential are given by

$$n_0 = C,$$

$$p_0 = N_V exp\left(-\mathcal{F}_{1/2}^{-1}\left(\frac{n_0}{N_C}\right) - \frac{E_G}{k_B T}\right),$$

$$\phi = \frac{1}{q}(E_{ref} - \chi) + \frac{k_B T}{q}\mathcal{F}_{1/2}^{-1}\left(\frac{n_0}{N_C}\right) + V \tag{25}$$

for a n-type semiconductor and by

$$p_0 = C,$$

$$n_0 = N_C exp\left(-\mathcal{F}_{1/2}^{-1}\left(\frac{p_0}{N_V}\right) - \frac{E_G}{k_B T}\right),$$

$$\phi = \frac{1}{q}(E_{ref} - \chi - E_G) - \frac{k_B T}{q}\mathcal{F}_{1/2}^{-1}\left(\frac{p_0}{N_V}\right) + V \tag{26}$$

for a p-type semiconductor.

When incomplete ionization models are used for dopants (see §9.1), implicit algebraic equations are used to solve for carrier densities instead of the explicit relations in Eq.(23), Eq.(24), Eq.(25) and Eq.(26). The carrier densities are computed using one of the system of algebric equations, selected by the user input:

$$n_0 - p_0 = N_D^+ - N_A^- = \frac{N_D}{1 + g_D\frac{n_0}{n_1}} - N_A,$$

$$n_0 p_0 = \gamma_n \gamma_p n_i^2 \tag{27}$$

$$n_0 - p_0 = N_D^+ - N_A^- = \frac{N_D}{1 + g_D\frac{n_0}{n_1}},$$

$$n_0 p_0 = \gamma_n \gamma_p n_i^2 \tag{28}$$

$$n_0 - p_0 = N_D^+ - N_A^- = \frac{N_D}{1 + g_D\frac{n_0}{n_1}} - \frac{N_A}{1 + g_A\frac{n_i^2}{n_1 n_0}},$$

$$n_0 p_0 = \gamma_n \gamma_p n_i^2 \tag{29}$$

for a n-type semiconductor and

$$n_0 - p_0 = N_D^+ - N_A^- = N_D - \frac{N_A}{1 + g_A\frac{p_0}{p_1}},$$

$$n_0 p_0 = \gamma_n \gamma_p n_i^2 \tag{30}$$

$$n_0 - p_0 = N_D^+ - N_A^- = -\frac{N_A}{1 + g_A\frac{p_0}{p_1}},$$

$$n_0 p_0 = \gamma_n \gamma_p n_i^2 \tag{31}$$

44

$$n_0 - p_0 \;=\; N_D^+ - N_A^- = \frac{N_D}{1 + g_D \frac{n_i^2}{n_1 p_0}} - \frac{N_A}{1 + g_A \frac{p_0}{p_1}},$$

$$n_0 p_0 \;=\; \gamma_n \gamma_p n_i^2 \tag{32}$$

for p-type semiconductor. For Maxwell-Boltzmann statistics the electron and hole degeneracy coefficients γ_n and γ_p become both 1.

To enable/use an Ohmic contact, a user can use one of the following methods to specify a voltage source. The first one is to use a constant voltage value, which is done using

```
BC is Ohmic for {contactName} on {geometryBlock} fixed at {voltage}
```

The second one is to sweep a voltage source from one voltage to another voltage using the syntax below

```
BC is Ohmic for {contactName} on {geometryBlock} swept from {voltage1} to
    {voltage2}
```

where `contactName` is a valid sideset name to be used as an Ohmic contact, `geometryBlock` is a valid geometry block name the Ohmic contact is connected to, `voltage` is a fixed voltage applied at the Ohmic contact, `voltage1` and `voltage2` are the start and end voltages at the Ohmic contact in the case of a voltage sweep.

The third one is to specify one sinusoidal voltage source using

```
BC is sinusoidal for {contactName} on {geometryBlock} with ampl = {amplitude} freq = {
    frequency} phase = {phase} and dc offset = {dcoffset}
```

or two sinusoidal voltage sources using

```
BC is sinusoidal for {contactName} on {geometryBlock} with ampl = {amplitude} freq = {
    frequency} phase = {phase} plus ampl2 = {amplitude2} freq2 = {frequency2} phase2 = {
    phase2} and dc offset = {dcoffset}
```

Here `amplitude` is the sinusoidal voltage amplitude in Volts, `frequency` is the frequency in Hertz, `phase` is the phase in a percentage of 2π and so its value is within $[0, 1]$, and `dcoffset` is a DC voltage applied. `amplitude2`, `frequency2`, `phase2` are used to specify a second sinusoidal voltage which is added to the first one.

The fourth method is to specify a trapezoid voltage source using the syntax given below

```
BC is trapezoid for {contactName} on {geometryBlock} with amplitude = {amplitude}
    period = {period} rise time = {risetime} fall time = {falltime} duty cycle = {dutycycle
    } dc offset = {DCoffset} delay = {delay} repeat = {repeat}
```

where `amplitude` is the peak voltage amplitude, `period` is the period of the pulse in seconds, `risetime` is the time to reach peak voltage from DC offset in seconds and `falltime` is the time to go from peak voltage to DC offset in seconds. The optional parameters are `dutycycle` which is the ratio of the pulse width to the period of the waveform with a value between $[0,1]$, `DCoffset` is the DC voltage applied, `delay` is the time in seconds before the first period is started and `repeat` is the number of periods as an integer.

4.2.　　　Schottky Contacts

Boundary conditions at Schottky contacts are based on the thermionic emission theory developed by Bethe [13]. The thermionic emission theory assumes that at the metal/semiconductor interface where the boundary conditions are derived, the barrier height is much larger than $k_B T$, thermal equilibrium is valid at the plane that determines emission and the net current flow through the interface does not effect the equilibrium. The energy band diagrams of the n-type and p-type Schottky interfaces between metal and semiconductor are shown in Figure 4-1, with the thick line indicating the interface where the boundary conditions are imposed. Carriers at Schottky interface are assumed to follow Maxwell-Boltzmann statistics. Under these assumptions, the boundary conditions for a n-type or p-type Schottky contact are:

$$
\begin{aligned}
q\phi &= E_{ref} - W_F + qV_{app}, \\
J_{n,norm} &= \frac{A_n^* T^2}{N_C}\left(n - n_0^B\right), \\
J_{p,norm} &= -\frac{A_p^* T^2}{N_V}\left(p - p_0^B\right)
\end{aligned}
\tag{33}
$$

where E_{ref} is the intrinsic Fermi energy level of the reference material from vacuum, A_n^* and A_p^* are electron and hole Richardson constants, $n_0^B = N_C exp\left(-\frac{q\phi_B}{k_B T}\right)$ and $p_0^B = N_V exp\left(\frac{-E_G + q\phi_B}{k_B T}\right)$ are the electron and hole equilibrium densities, $J_{n,norm}$ and $J_{p,norm}$ are the electron and hole currents normal to the metal/semiconductor interface, W_F is the metal workfunction and V_{app} is the applied voltage at the Schottky contact.

(a) n-type　　　　　　　　　　　　**(b)** p-type
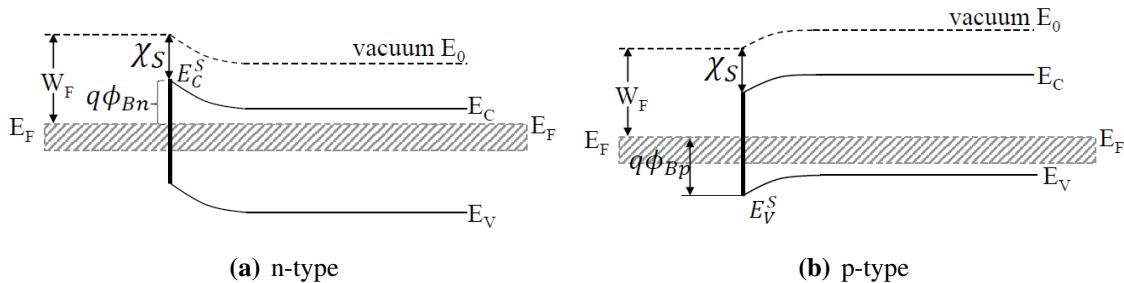
Figure 4-1 Band diagram of Schottky contacts

To enable/use a Schottky contact the user must specify the following blocks at the root level of the interpreter input file:

```
start Schottky BC for {contactName} on {materialName}
  voltage is fixed at {voltage}
  type is electron/hole
  (one parameter per line in the form):
  {parameter name} = {parameter value}
  (see table 4-1 for available parameters)
end
```

or

```
start Schottky BC for {contactName} on {materialName}
  voltage is swept from {voltage} to {voltage2}
  type is electron/hole
  (one parameter per line in the form):
  {parameter name} = {parameter value}
  (see table 4-1 for available parameters)
end
```

Table 4-1 Syntax and parameters for the Schottky contact.

Input file	Corresponding variable in (33)	Description	Units
work function	W_F	contact metal work function	eV
electron richardson constant	A_n^*	electron Richardson constant	$\frac{A}{K^2 cm^2}$
hole richardson constant	A_p^*	hole Richardson constant	$\frac{A}{K^2 cm^2}$

4.2.1.   *Schottky Contacts Barrier Lowering*

The barrier lowering model implemented by Charon accounts for two different physical mechanisms at the Schottky contacts: image force potential and dipole effect. The barrier height change is modeled as

$$\Delta\phi_B = \alpha\sqrt{\frac{qE}{4\pi\varepsilon_{semic}}} + \beta E^\gamma \tag{34}$$

where the first term is due the image force potential and the second term due to dipole effect. The default values for the parameters in Eq.(34) are $\alpha = 1.0$, $\beta = 0.0$ and $\gamma = 1.0$. When barrier lowering model is used, the Schottky boundary conditions in Eq.(33) remains unchanged but equilibrium electron and hole densities at the contact are modified to account for the lowering of the barrier. For a n-type contact equilibrium densities change to:

$$
\begin{aligned}
n_0^B &= N_C exp\left(-\frac{q(\phi_B - \Delta\phi_B(E))}{k_B T}\right), \\
p_0^B &= N_V exp\left(\frac{-E_G + q(\phi_B - \Delta\phi_B(E))}{k_B T}\right)
\end{aligned} \tag{35}
$$

and for a p-type contact to

$$
\begin{aligned}
n_0^B &= N_C exp\left(-\frac{q(\phi_B + \Delta\phi_B(E))}{k_B T}\right), \\
p_0^B &= N_V exp\left(\frac{-E_G + q(\phi_B + \Delta\phi_B(E))}{k_B T}\right)
\end{aligned} \tag{36}
$$

Poisson equation is solved consistently with the real charge seeing a full barrier at the Schottky contact, while the electron and hole continuity equations see a combined Poisson and image-force potential.

To activate the barrier lowering model at a Schottky contact the user must specify the following block inside the Schottky contact block

```
start Schottky BC for {contactName} on {materialName}
  ...
  start barrier lowering parameters
   (one parameter per line in the form):
   /*!{parameter name} = {parameter value}
   (see table 4-2 for available parameters)
  end
  ...
end
```

Table 4-2 Syntax and parameters for the Schottky Contact Barrier Lowering.

Input file	Corresponding variable in (34)	Description	Units	Default
alpha	α	specifies parameter alpha	*none*	1.0
beta	β	specifies parameter beta	*none*	0.0
gamma	γ	specifies parameter gamma	*none*	1.0

4.2.2. *Schottky Contacts Barrier Tunneling*

Effects of tunneling at the Schottky contacts can be taken into account by enabling tunneling. Charon uses a tunneling model based on Tsu-Esaki formula (see [14]) to simulate the effects of tunneling currents through the Schottky barrier. The tunneling currents at the Schottky contact are given by:

$$
\begin{aligned}
J_e &= \frac{4\pi q k_B T m^e_{eff}}{h^3} \int_{E_{F,M}+qV_a}^{E_{F,M}+q\phi_{Bn}} TC_e(\xi) N_e(\xi)\,\mathrm{d}\xi, \\
J_h &= \frac{4\pi q k_B T m^h_{eff}}{h^3} \int_{E_{F,M}+qV_a}^{E_{F,M}+q\phi_{Bp}} TC_h(\xi) N_h(\xi)\,\mathrm{d}\xi
\end{aligned}
\tag{37}
$$

where

$$
\begin{aligned}
TC_e(\xi) &= \exp\left(-\frac{8\pi\sqrt{2m^e_{tunnel}m_0}}{3qhE}(q\phi_{Bn}-\xi)^{3/2}\right), \\
TC_h(\xi) &= \exp\left(-\frac{8\pi\sqrt{2m^h_{tunnel}m_0}}{3qhE}(q\phi_{Bp}-\xi)^{3/2}\right)
\end{aligned}
\tag{38}
$$

48

are the transmission coefficients of the Schottky barrier and

$$
\begin{aligned}
N_e(\xi) &= \ln\left(\frac{1+\exp\left(-\frac{\xi-E_{F,M}}{k_BT}\right)}{1+\exp\left(-\frac{\xi-(E_{F,M}+qV_a)}{k_BT}\right)}\right), \\
N_h(\xi) &= \ln\left(\frac{1+\exp\left(-\frac{E_{F,M}-\xi}{k_BT}\right)}{1+\exp\left(-\frac{(E_{F,M}+qV_a)-\xi}{k_BT}\right)}\right)
\end{aligned}
\tag{39}
$$

are the carrier supply functions.

The transmission coefficients $TC_e(\xi)$ and $TC_h(\xi)$ are a measure of the Schottky barrier transparency and they are functions of the electric field at the contact on the semiconductor side. The supply functions $N_e(\xi)$ and $N_h(\xi)$ are the differences in the supply of carriers at the interface and they are derived assuming Fermi-Dirac statistics for carriers. When the tunnneling model is enabled, the tunneling currents in Eq.(37) are added to the thermionic emission currents in Eq.(33) as a total current at the Schottky contact.

To activate the tunneling model at the Schottky contact the user must specify the following block inside the Schottky contact block

```
start Schottky BC for {contactName} on {materialName}
  ...
  start barrier tunneling parameters
   (one parameter per line in the form):
  {parameter name} = {parameter value}
   (see table 4-3 for available parameters)
  end
  ...
end
```

Table 4-3 Syntax and parameters for the Schottky Contact Tunneling.

Input file	Corresponding variable in (37) and (38)	Description	Units
mass	m^e_{tunnel} or m^h_{tunnel}	specifies carrier relative tunneling mass	*none*

4.3. Gate Contacts

Boundary conditions at gate contacts or contacts on insulators consist of giving the electrostatic potential as a function of the contact workfunction and applied voltage at the contact:

$$
q\phi = E_{ref} - W_F + qV_{app}
\tag{40}
$$

where E_{ref} is the intrinsic Fermi energy level of the reference material from vacuum, W_F is the metal workfunction and V_{app} is the applied voltage at the gate contact.

To enable/use a gate contact the user must specify the following lines at the root level of the interpreter input file, depending on the type of biasing used:

```
BC is contact on insulator for {contactName} on {MaterialName} with
  work function {WF} fixed at {voltage}
```

or

```
BC is contact on insulator for {contactName} on {MaterialName} with
  work function {WF} swept from {voltage1} to {voltage2}
```

or

```
BC is contact on insulator for {contactName} on {MaterialName} with
  work function {WF} linearly ramped from {time0} to {time1} and
  voltage from {voltage0} to {voltage1}
```

where *contactName* is a valid sideset name to be used as a gate contact, *materialName* is a valid insulator material name the gate contact is connected to, *WF* is the workfunction of the gate, *voltage* is a fixed voltage applied at the gate contact, *voltage*1 and *voltage*2 are the start and end voltages at the gate contact in the case of a voltage sweep and *voltage*0 and *voltage*1 are the applied voltages at *time*0 and respectively *time*1 for the linear time ramp.

4.4.   Constraint Boundary Conditions

There are two types of constraints supported at device contacts:

Constant Current This constraint corresponds to the user attaching a current source to one terminal of the device. The user therefore specifies that current value in amps. There can be at most one of these constraints on the device.

Resistor Contact This constraint corresponds to the user attaching a resistor to one terminal of the device, with a voltage source on the far side of the resistor. The user therefore specifies that applied voltage in volts, along with the resistance in ohms.

All constraints have the following common attributes:

- the sideset ID denoting the terminal to which the constraint is applied;

- the initial value given to the voltage parameter corresponding to this constraint; and

- the element block ID corresponding to the sideset ID.

There can be at most one constraint on any given terminal of the device.

4.4.1. Device Contact Dimensions

In addition to the common attributes listed above, all constraints can have two other common attributes that are only applicable in a two-dimensional simulation:

- the length of the contact in the simulation in μm; and

- the actual device contact area in μm^2.

Internally Charon will deal with currents in A/cm for 2-D simulations, so these values are used to scale the currents to Amps if you would like to compare to experimental results. The first is the length of the sideset to which the constraint is applied. You should know this value from the mesh you created in Cubit. The second is the surface area of the contact on the actual device you're simulating. Note that the units here are μm and μm^2, respectively. These two quantities must be specified together; you cannot specify one without the other.

If you choose to omit these quantities for a 2-D simulation, there will be no scaling of the currents that takes place. If you accidentally specify these quantities for a 3-D simulation, they will simply be ignored.

4.4.2. Constant Current

Now let's take a look at how you would specify one of these constraints in an input deck.

```
bc is current for anode on silicon fixed at -3.9e-7 with initial voltage 0.1
```

The `bc is current` indicates to Charon that this contact has a constant-current applied to it for the simulation. The `for anode on silicon` tells Charon what contact the constant-current will be applied to, in this case the contact named `anode`, and what geometry block that contact is associated with, `silicon` in this case.

At present `fixed` is the only valid keyword when an electric current is applied to a contact and it is followed by the value of the fixed current, `at -3.9e-7`, or $-0.39\mu A$. As stated in Section 3.2, the minus sign on the value indicates the current is flowing into the specified contact.

Finally the applied bias to use as the starting point on the contact is given, `with initial voltage 0.1`. This is just a guess of the bias at the contact that would yield the given current. Generally you can get a good estimate of this by looking at IV curves associated with the device and picking off the voltage at the specified current. The better this estimate the more rapidly the simulation will complete. If the value of initial voltage is too far off the actual value then Charon may have trouble converging. You can also think of the initial voltage as the parameter that will be iteratively solved for as the simulation proceeds, thus the need for a good initial guess.

The full syntax for this boundary condition is given in Table 4-4.

Table 4-4 Syntax for specifying constant current at a contact

BC is current for {*contactName*} on {*materialName*} fixed at {*current*} with initial voltage {*voltage*} **[** with area {*contact area*} **[** and length {*contact length*} **[** with base doping {*base doping type*} **]** **]** **]**		
Option	Description	Required
fixed at	Current is fixed at this contact	Yes
with initial voltage	Voltage to start simulation with. This can be considered an initial guess for the voltage.	Yes
with area	The contact area (cm^2)	No
and length	The contact length (*cm*)	No
with base doping	When utilized in a pseudo-1D BJT simulation, the doping at the base contact	No
Variables	Description	Default
contactName	Specifies a valid sideset name to be used as an Ohmic contact	*none*
materialName	Specifies a valid material name the Ohmic contact is connected to.	*none*
current	Fixed current applied at the contact, in amps	*none*
voltage	Approximate starting voltage at the contact. This can be considered an initial guess for the voltage at the contact.	*none*
contact area	Area of the contact (cm^2)	1.0
contact length	Length of the contact (*cm*)	1.0
base doping type	Type of doping at the base of a pseudo-1D BJT (*acceptor* or *donor*)	*none*

4.4.3.   *Resistor Contact*

The syntax used to perform a simulation with a resistor attached to a contact is similar to that used for a constant-current constraint.

```
bc is resistor for cathode on silicon with resistor 1e3 fixed at -4e-3 and
   initial voltage 0.1
```

Note that the input above would be on a single line in the input file but was broken here to avoid running past the margins.

The `is resistor` indicates the boundary-condition to be used here utilizes a resistor attached to the contact that is specified via `for cathode on silicon`. The resistance of the attached resistor is specified with `with resistor 1e3`. The value is always in ohms so in this example the resistance is $1k\Omega$.

The `fixed at -4e3` specifies the value of the voltage on the terminal that is not attached to the device. The `initial voltage` is the same as that specified in the constant-current simulation and can be considered an initial guess for the voltage at the device terminal to which the resistor is attached. An illustration is shown in Figure 4-2.
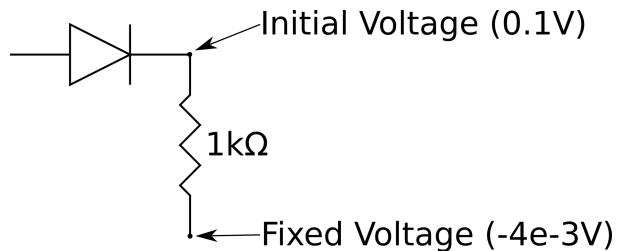
Figure 4-2 Diagram for illustration of a resistor attached to a device contact.

The full syntax for this boundary condition is given in Table 4-5.

Table 4-5 Syntax for specifying resistor attached to a contact

BC is resistor for {*contactName*} on {*materialName*} with resistor {*resistance*} fixed at {*voltage1*} and initial voltage {*voltage2*} **[** with area {*contact area*} **[** and length {*contact length*} **[** with base doping {*base doping type*} **]]]**		
Option	Description	Required
with resistor	The resistance value of the attached resistor	Yes
fixed at	Voltage is fixed at the resistor contact NOT connected to the device. See Figure 4-2.	Yes
with initial voltage	Voltage to start simulation with. This can be considered an initial guess for the voltage.	Yes
with area	The contact area (cm^2)	No
and length	The contact length (cm)	No
with base doping	When utilized in a pseudo-1D BJT simulation, the doping at the base contact	No
Variables	Description	Default
contactName	Specifies a valid sideset name to be used as an Ohmic contact	*none*
materialName	Specifies a valid material name the Ohmic contact is connected to.	*none*
resistance	The resistance value of the attached resistor (ohms).	*none*
voltage1	Voltage applied at the resistor contact (volts).	*none*
voltage2	Approximate starting voltage at the device contact where the resistor is attached. This can be considered an initial guess for the voltage at the contact.	*none*
contact area	Area of the contact (cm^2)	1.0
contact length	Length of the contact (cm)	1.0
base doping type	Type of doping at the base of a pseudo-1D BJT (*acceptor* or *donor*)	*none*

4.4.4. Constant Current on the Base Contact of a Pseudo 1D BJT

In order to perform a simulation of a pseudo one-dimensional *BJT* a special boundary condition must be utilized for the base contact due to the fact that such a contact is not physical in a pseudo 1D simulation of a *BJT*. When using this boundary condition often you might want to apply a constant current to the base contact. This is possible via the input line

```
bc is current for base on silicon fixed at 1.5e-6 and initial voltage 0.0 with base
    doping acceptor
```

Note that the example above was wrapped to avoid clipping. It would comprise a single line in an actual input file.

The only difference to note between this case and the one given in Section 4.4.2 is the addition of the `with base doping acceptor`. This tells Charon that the contact is a base contact in an pseudo-1D, NPN *BJT*.

An illustration of a pseudo-1D *BJT* is given in Figure 4-3. The information for this type of boundary condition can be found in Table 4-4.

Figure 4-3 Pseudo one-dimensional *BJT*.

4.4.5. Solver Specifications for Constrained Problems

Constrained problems generally require slightly different solver settings than typical steady-state problems. At the present time those settings are encapsulated in Charon's `solver pack 3`. An example of a solver specification for a simulation with a constraint would be

```
start solver block
    use solver pack 3
end solver block
```

The solver settings are constantly being refined for specific problem types and so it is generally beneficial to examine the latest version of this manual, or contact the developers directly, to get recommendations on solver settings for specific problem types, particularly when convergence problems arise for your simulation.

5. INITIAL GUESS

The initial guess is the starting solution for the nonlinear solver. The rate of convergence and whatever the convergence is obtained or not are strongly dependent on how close the initial guess is to the real solution. Charon can compute/estimate or read the initial guess for electrostatic potential in semiconductor or insulator blocks and for carrier densities in semiconductor blocks. The initial guess for lattice temperature can be set to a constant or read from a previous simulation for all material blocks.

5.1. Initial Guess in Semiconductors

To activate the initial guess in semiconductor blocks, the user must specify one or more of the following line at the root level of the interpreter input file

```
initial conditions for {DOF} in {BlockName} is {option}
```

where the command and its options are described in Table 5-1.

Table 5-1 Syntax for Initial Conditions.

<table>
<tr><td colspan="3">initial conditions for {DOF} in {BlockName} is [
exodus file [
{compType} [
constant = {value} [
uninitialized]]]]</td></tr>
<tr><td>Option</td><td>Description</td><td>Required</td></tr>
<tr><td></td><td>None</td><td></td></tr>
<tr><td>Variables</td><td>Description</td><td>Default</td></tr>
<tr><td>DOF</td><td>Choose one of: <code>ELECTRIC_POTENTIAL</code>, <code>ELECTRON_DENSITY</code>, <code>HOLE_DENSITY</code>, <code>LATTICE_TEMPERATURE</code></td><td>None</td></tr>
<tr><td>BlockName</td><td>Specify the block in which this initial condition is to be specified.</td><td>None</td></tr>
<tr><td>exodus file</td><td>This will read in the degree of freedom from the input exodus file.</td><td>None</td></tr>
<tr><td>compType</td><td>Choose equilibrium potential or equilibrium density.</td><td>None</td></tr>
<tr><td>constant</td><td>This specifies that a constant value will be set for this degree of freedom.</td><td>None</td></tr>
<tr><td>value</td><td>This will set the constant value.</td><td>None</td></tr>
<tr><td>uninitialized</td><td>This leaves the degree of freedom uninitialized (zero).</td><td>None</td></tr>
</table>

To initialize the initial guess to a constant value throughout a semiconductor block the user must use:

```
initial conditions for {DOF} in {BlockName} is constant = {value}
```

where *value* is a user specified constant. When the *DOF* is zero, the command becomes

```
initial conditions for {DOF} in {BlockName} is uninitialized
```

To let Charon compute/estimate the initial guess in equilibrium, the user must use

```
initial conditions for ELECTRIC_POTENTIAL in {BlockName} is equilibrium potential
```

for electrostatic potential and

```
initial conditions for ELECTRON_DENSITY in {BlockName} is equilibrium density
```

```
initial conditions for HOLE_DENSITY in {BlockName} is equilibrium density
```

for carrier densities. In this case the electrostatic potential and carrier densities are computed depending on doping type of the block. For isothermal simulations the electrostatic potential and carrier densities are given by

$$
\begin{aligned}
y &= \frac{N_D - N_A}{2N_C} + \sqrt{\left(\frac{N_D - N_A}{2N_C}\right)^2 + \frac{N_V}{N_C} exp\left(-\frac{E_{G,\text{eff}}}{k_B T}\right)}, \\
\phi &= \frac{E_{ref}}{q} - \frac{\chi_{\text{eff}}}{q} + \frac{k_B T ln(y)}{q}, \\
n &= N_C exp\left(-\frac{E_C}{k_B T}\right), \\
p &= N_V exp\left(\frac{E_V}{k_B T}\right)
\end{aligned}
\tag{41}
$$

for a n-type semiconductor block and as

$$
\begin{aligned}
y &= -\frac{N_D - N_A}{2N_V} + \sqrt{\left(\frac{N_D - N_A}{2N_V}\right)^2 + \frac{N_C}{N_V} exp\left(-\frac{E_{G,\text{eff}}}{k_B T}\right)}, \\
\phi &= \frac{E_{ref}}{q} - \frac{\chi_{\text{eff}}}{q} + \frac{E_{G,\text{eff}}}{q} - \frac{k_B T ln(y)}{q}, \\
n &= N_C exp\left(-\frac{E_C}{k_B T}\right), \\
p &= N_V exp\left(\frac{E_V}{k_B T}\right)
\end{aligned}
\tag{42}
$$

for a p-type block. For non-isothermal simulations the initial guess is computed as

$$
\begin{aligned}
\theta(T) &= \frac{\chi_{\text{eff}}}{q} + \frac{E_{G,\text{eff}}}{2q} + \frac{k_B T}{2q} ln\left(\frac{N_C}{N_V}\right), \\
y &= \frac{N_D - N_A}{2N_C} + \sqrt{\left(\frac{N_D - N_A}{2N_C}\right)^2 + \frac{N_V}{N_C} exp\left(-\frac{E_{G,\text{eff}}}{k_B T}\right)}, \\
\phi &= \theta(T) - \frac{\chi_{\text{eff}}}{q} + \frac{k_B T ln(y)}{q}, \\
n &= N_C exp\left(-\frac{E_C}{k_B T}\right), \\
p &= N_V exp\left(\frac{E_V}{k_B T}\right)
\end{aligned}
\tag{43}
$$

for n-type semiconductors blocks and as

$$
\begin{aligned}
\theta(T) &= \frac{\chi_{\text{eff}}}{q} + \frac{E_{\text{G,eff}}}{2q} + \frac{k_B T}{2q} ln\left(\frac{N_C}{N_V}\right), \\
y &= -\frac{N_D - N_A}{2N_V} + \sqrt{\left(\frac{N_D - N_A}{2N_V}\right)^2 + \frac{N_C}{N_V} exp\left(-\frac{E_{\text{G,eff}}}{k_B T}\right)}, \\
\phi &= \theta(T) - \frac{\chi_{\text{eff}}}{q} + \frac{E_{\text{G,eff}}}{q} - \frac{k_B T ln(y)}{q}, \\
n &= N_C exp\left(-\frac{E_C}{k_B T}\right), \\
p &= N_V exp\left(\frac{E_V}{k_B T}\right)
\end{aligned}
\tag{44}
$$

for p-type blocks. If the semiconductor blocks are connected to Schottky contacts, then the potential on the Schottky contact nodes is computed as $\phi = \frac{E_{ref}}{q} - \frac{W_F}{q} + V_{app}$ for isothermal simulations and $\phi = \theta - \frac{W_F}{q} + V_{app}$ for non-isothermal simulations.

To use a previously computed solution and use it as an initial guess for the current simulation, the user must specify

```
initial conditions for {DOF} in {BlockName} is exodus file
```

In this case the solution found in the input state file is used as initial guess.

Building a good initial guess requires a combination of the options described above. For instance, to build a good initial guess for a drift diffusion simulation, the user must first solve the nonlinear Poisson equation with the initial guess for electrostatic potential computed

```
initial conditions for ELECTRIC_POTENTIAL in {BlockName} is equilibrium potential
```

followed by solving drift diffusion equations with the initial guess for electrostatic potential loaded from the previously solved nonlinear Poisson equation and the initial guess for carrier densities computed by Charon

```
initial conditions for ELECTRIC_POTENTIAL in {BlockName} is exodus file

initial conditions for ELECTRON_DENSITY in {BlockName} is equilibrium density

initial conditions for HOLE_DENSITY in {BlockName} is equilibrium density
```

5.2.   Initial Guess in Insulators

To activate the initial guess in insulator blocks, the user must specify the following line at the root level of the interpreter input file

```
initial conditions for {DOF} in {BlockName} is {option}
```

where the command and its options are described in Table 5-2. The *exodus file*, *constant = {value}* and *uninitialized* options are similar to those in semiconductor blocks and described in Section 5.1.

When *equilibrium potential* or *equilibrium potential with semblocks = ({semBlockIds})* options are used, Charon computes the potential on the contact gates connected to the insulator block as $\phi^{\text{gate}} = V_{\text{app}}^{\text{gate}} - W_{\text{F}}^{\text{gate}} + E_{\text{ref}}/q$ where E_{ref} is the intrinsic Fermi energy level of the reference material from vacuum, $W_{\text{F}}^{\text{gate}}$ is the metal workfunction and $V_{\text{app}}^{\text{gate}}$ is the applied voltage at the gate contact. The potential on the other nodes of the insulator block is initialized to zero.

For *equilibrium potential with semblocks = ({semBlockIds})* option, in addition to the computation of the potential on the gates as described above, Charon also computes the potential at insulator/semiconductor interfaces defined by the insulator block and the semiconductor blocks whose block ids are specified by the *{semBlockIds}* list. On these interfaces, the potential is calculated similar to the one in semiconductors (see Eq (41) for a n-type block or Eq (42) for a p-type block) where N_C and N_V have the default values at 300K for the semiconductor materials defined by *{semBlockIds}* list and N_A or N_D are given by the user as doping in the insulator material parameter section, chosen to be close to the doping in the corresponding semiconductor side of the interface. When this option is used, defining the doping in the insulator material parameter section is mandatory.

Table 5-2 Syntax for Initial Conditions in Insulators.

initial conditions for {*DOF*} in {*BlockName*} is [
exodus file [
equilibrium potential [
equilibrium potential with semblocks = ({*semBlockIds*}) [
constant = {*value*} [
uninitialized]]]]]

Option	Description	Required
	None	

Variables	Description	Default
DOF	Choose one of: `ELECTRIC_POTENTIAL`, `LATTICE_TEMPERATURE`	*None*
BlockName	Specify the block in which this initial condition is to be specified.	*None*
exodus file	This will read in the degree of freedom from the input exodus file.	*None*
equilibrium potential	equilibrium potential for `ELECTRIC_POTENTIAL`	*None*
equilibrium potential with semblocks = (*semBlockIds*)	equilibrium potential with potential calculation at insulator semiconductor interfaces; *semBlockIds* is a list of valid semiconductor blocks having an interface with the insulator block defined by *BlockName*	*None*
constant	This specifies that a constant value will be set for this degree of freedom.	*None*
value	This will set the constant value.	*None*
uninitialized	This leaves the degree of freedom uninitialized (zero).	*None*

6.   BAND STRUCTURE

6.1.   Intrinsic Density

Effective intrinsic density in Charon is given by

$$n_{ie}(T) = \sqrt{N_C(T)N_V(T)}exp\left(\frac{-E_G(T)}{2k_BT}\right)exp\left(\frac{\Delta E_G}{2k_BT}\right) \tag{45}$$

where N_C, N_V are the conduction and valence band effective density of states, E_G is the bandgap energy and ΔE_G is the bandgap narrowing.

By default no bandgap narrowing is taken into account ($\Delta E_G = 0.0$). Four bandgap narrowing models are available in Charon on user request: Old Slotboom [15], Slotboom [16], Harmon [17] and Persson [18]. Four intrinsic models with the same names of the bandgap narrowing models are available.

Old Slotboom model computes bandgap narrowing due to the heavy doping as a function of the total doping [15]

$$\Delta E_G = V_{0,\text{bgn}}\left(ln\left(\frac{N_{tot}}{N_{0,\text{bgn}}}\right) + \sqrt{\left(ln\left(\frac{N_{tot}}{N_{0,\text{bgn}}}\right)\right)^2 + C_{bgn}}\right) \quad if \quad N_{tot} \geqslant 10^{10}cm^{-3},$$

$$\Delta E_G = 0 \quad if \quad N_{tot} < 10^{10}cm^{-3} \tag{46}$$

where N_{tot} is the total doping and parameters $V_{0,\text{bgn}}$, $N_{0,\text{bgn}}$ and C_{bgn} and their description are listed in table 6-1.

To enable Old Slotboom intrinsic density with associated bandgap narrowing model, the user must enable bandgap narrowing in the appropriate physics section

```
start Physics Block {physicsBlockName}
  ...
  band gap narrowing is on
  ...
end
```

and specify the Old Slotboom intrinsic concentration model with its parameters in the material block section

```
start Material Block {materialBlockName}
  start intrinsic concentration
    model is old slotboom
    start oldslotboom parameters
      (one parameter per line in the form):
      {parameter name} = {parameter value}
      (see table 6-1 for available parameters)
    end
  end
end
```

Table 6-1 Syntax and parameters for Old Slotboom model.

Input file	Corresponding variable in (46)	Description	units
V0_BGN	$V_{0,\mathrm{bgn}}$	$V_{0,\mathrm{bgn}}$ coefficient	V
N0_BGN	$N_{0,\mathrm{bgn}}$	$N_{0,\mathrm{bgn}}$ coefficient	$\frac{1}{cm^3}$
CON_BGN	C_{bgn}	C_{bgn} coefficient	$none$

If no parameters are specified, default parameters for the corresponding material are used.

Slotboom model [16] is similar to Old Slotboom model and it uses the same formula to compute the bandgap narrowing, but it has different default parameters for $V_{0,\mathrm{bgn}}$, $N_{0,\mathrm{bgn}}$ (see table 6-2) and a different N_{tot} cutoff

$$\Delta E_G = V_{0,\mathrm{bgn}} \left(ln \left(\frac{N_{tot}}{N_{0,\mathrm{bgn}}} \right) + \sqrt{ \left(ln \left(\frac{N_{tot}}{N_{0,\mathrm{bgn}}} \right) \right)^2 + C_{bgn} } \right) \quad if \quad N_{tot} \geqslant 10^{15} cm^{-3},$$

$$\Delta E_G = 0 \quad if \quad N_{tot} < 10^{15} cm^{-3} \tag{47}$$

To use Slotboom intrinsic density with associated bandgap narrowing model, the user must enable bandgap narrowing in the appropriate physics section

```
start Physics Block {physicsBlockName}
  ...
  band gap narrowing is on
  ...
end
```

and specify Slotboom intrinsic concentration model with its parameters in the material block section

```
start Material Block {materialBlockName}
  start intrinsic concentration
    model is slotboom
    start slotboom parameters
      (one parameter per line in the form):
      {parameter name} = {parameter value}
      (see table 6-2 for available parameters)
    end
  end
end
```

If no parameters are specified, default parameters for the corresponding material are used.

Harmon model [17] calculates the bandgap narrowing as

$$\Delta E_G = A_n (N_D - N_A)^{1/3} + \Delta E_{G,\mathrm{ncor}} \quad if \quad N_D - N_A \geqslant 0,$$

$$\Delta E_G = A_p (N_A - N_D)^{1/3} + \Delta E_{G,\mathrm{pcor}} \quad if \quad N_D - N_A < 0 \tag{48}$$

Table 6-2 Syntax and parameters for Slotboom model.

Input file	Corresponding variable in (47)	Description	units
V0_BGN	$V_{0,\mathrm{bgn}}$	$V_{0,\mathrm{bgn}}$ coefficient	V
N0_BGN	$N_{0,\mathrm{bgn}}$	$N_{0,\mathrm{bgn}}$ coefficient	$\frac{1}{cm^3}$
CON_BGN	C_{bgn}	C_{bgn} coefficient	*none*

where material parameters A_n and A_p are described in table 6-3. $\Delta E_{\mathrm{G,ncor}}$ and $\Delta E_{\mathrm{G,pcor}}$ are Fermi-Dirac correction terms which are activated when Fermi Dirac statistics for carriers is used. For Boltzmann statistics, the default statistics in Charon, the corrections terms become zero. The doping-dependent Fermi-Dirac corrections terms are given by

$$
\begin{aligned}
\Delta E_{\mathrm{G,ncor}} &= k_B T \left(ln\left(\frac{N_D - N_A}{N_C}\right) - \mathcal{F}_{1/2}^{-1}\left(\frac{N_D - N_A}{N_C}\right) \right) \quad if \quad \frac{N_D - N_A}{N_C} \geqslant 10^{-4}, \\
\Delta E_{\mathrm{G,ncor}} &= 0.0 \quad if \quad \frac{N_D - N_A}{N_C} < 10^{-4}
\end{aligned}
\tag{49}
$$

where $N_D - N_A \geqslant 0$ (n-type) and

$$
\begin{aligned}
\Delta E_{\mathrm{G,ncor}} &= k_B T \left(ln\left(\frac{N_D - N_A}{N_C}\right) - \mathcal{F}_{1/2}^{-1}\left(\frac{N_D - N_A}{N_C}\right) \right) \quad if \quad \frac{N_D - N_A}{N_C} \geqslant 10^{-4}, \\
\Delta E_{\mathrm{G,pcor}} &= k_B T \left(ln\left(\frac{N_A - N_D}{N_V}\right) - \mathcal{F}_{1/2}^{-1}\left(\frac{N_A - N_D}{N_V}\right) \right) \quad if \quad \frac{N_A - N_D}{N_V} \geqslant 10^{-4}, \\
\Delta E_{\mathrm{G,pcor}} &= 0.0 \quad if \quad \frac{N_A - N_D}{N_V} < 10^{-4}
\end{aligned}
\tag{50}
$$

where $N_A - N_D \geqslant 0$ (p-type).

To enable Harmon intrinsic density with associated bandgap narrowing model, the user must enable bandgap narrowing in the appropriate physics section

```
start Physics Block {physicsBlockName}
  ...
  band gap narrowing is on
  ...
end
```

and specify Harmon intrinsic concentration model with its parameters in the material block section

```
start Material Block {materialBlockName}
  start intrinsic concentration
    model is harmon
    start harmon parameters
      enable fermi = {fermiCorrection}
```

```
        (one parameter per line in the form):
        {parameter name} = {parameter value}
        (see table 6-3 for available parameters)
      end
    end
  end
```

where *fermiCorrection* flag must be true to enable Fermi-Dirac corrections in (49) and (50) or false or not specified if corrections are not used.

Harmon model has the option to calculate the bandgap narrowing directly based on a file provided by the user with lines specifying changes in conduction and valence band due to narrowing as a function of doping with the following format:

$$
\begin{array}{ccc}
\text{Doping}_1 & \Delta E_{C,1} & \Delta E_{V,1} \\
\text{Doping}_2 & \Delta E_{C,2} & \Delta E_{V,2} \\
& . & \\
& . & \\
& . & \\
\text{Doping}_n & \Delta E_{C,n} & \Delta E_{V,n}
\end{array}
$$

In this case $\Delta E_G = \Delta E_C + \Delta E_V$ and formulas (48), (49) and (50) are not used.

To use Harmon intrinsic density with bandgap narrowing computed from the file, the user must enable bandgap narrowing in the appropriate physics section

```
start Physics Block {physicsBlockName}
  ...
  band gap narrowing is on
  ...
end
```

and specify bgn file option in the Harmon intrinsic model parameter section

```
start Material Block {materialBlockName}
  start intrinsic concentration
    model is harmon
    start harmon parameters
      bgn file = {fileName}
    end
  end
end
```

with *fileName* the name of the file containing the bandgap narrowing data.

Persson bandgap narrowing model [18] computes doping-induced energy shifts in the conduction and valence bands. The model takes into account the exchange energy for majority carriers, correlation energy for minority carriers, and impurity interaction between the conduction and

Table 6-3 Syntax and parameters for Harmon model.

Input file	Corresponding variable in (48)	Description	units
an	A_n	A_n coefficient	$eV \cdot cm$
ap	A_p	A_p coefficient	$eV \cdot cm$

valence bands. The band shifts of the conduction and valence band edges for n-type materials are given by

$$
\begin{aligned}
\Delta E_{C,n} &= A_{\text{n,C}} \left(\frac{N_D - N_A}{10^{18}} \right)^{1/3} + B_{\text{n,C}} \left(\frac{N_D - N_A}{10^{18}} \right)^{1/4} + C_{\text{n,C}} \left(\frac{N_D - N_A}{10^{18}} \right)^{1/2}, \\
\Delta E_{V,n} &= A_{\text{n,V}} \left(\frac{N_D - N_A}{10^{18}} \right)^{1/3} + B_{\text{n,V}} \left(\frac{N_D - N_A}{10^{18}} \right)^{1/4} + C_{\text{n,V}} \left(\frac{N_D - N_A}{10^{18}} \right)^{1/2}
\end{aligned}
\tag{51}
$$

and for p-type materials by

$$
\begin{aligned}
\Delta E_{C,p} &= A_{\text{p,C}} \left(\frac{N_A - N_D}{10^{18}} \right)^{1/3} + B_{\text{p,C}} \left(\frac{N_A - N_D}{10^{18}} \right)^{1/4} + C_{\text{p,C}} \left(\frac{N_D - N_A}{10^{18}} \right)^{1/2}, \\
\Delta E_{V,p} &= A_{\text{p,V}} \left(\frac{N_A - N_D}{10^{18}} \right)^{1/3} + B_{\text{p,V}} \left(\frac{N_A - N_D}{10^{18}} \right)^{1/4} + C_{\text{p,V}} \left(\frac{N_A - N_D}{10^{18}} \right)^{1/2}
\end{aligned}
\tag{52}
$$

with the parameters $A_{\text{n,C}}$, $B_{\text{n,C}}$, $C_{\text{n,C}}$, $A_{\text{n,V}}$, $B_{\text{n,V}}$, $C_{\text{n,V}}$, $A_{\text{p,C}}$, $B_{\text{p,C}}$, $C_{\text{p,C}}$, $A_{\text{p,V}}$, $B_{\text{p,V}}$ and $C_{\text{p,V}}$ described in table 6-4.

Then, the total bandgap narrowing given by the Persson model for a n-type material can be written as

$$
\Delta E_G = \Delta E_{V,n} - \Delta E_{C,n}
\tag{53}
$$

and for a p-type material as

$$
\Delta E_G = \Delta E_{V,p} - \Delta E_{C,p}
\tag{54}
$$

To use Persson intrinsic density with associated bandgap narrowing model, the user must enable bandgap narrowing in the appropriate physics section

```
start Physics Block {physicsBlockName}
  ...
  band gap narrowing is on
  ...
end
```

and specify Persson intrinsic concentration model with its parameters in the material block section

```
start Material Block {materialBlockName}
  start intrinsic concentration
    model is persson
    start persson parameters
```

```
      (one parameter per line in the form):
      {parameter name} = {parameter value}
      (see table 6-4 for available parameters)
    end
  end
end
```

If no parameters are specified, default parameters for the corresponding material are used.

Table 6-4 Syntax and parameters for Persson model.

Input file	Corresponding variable in (51), (52)	Description	units
ANC_BGN	$A_{n,C}$	$A_{n,C}$ coefficient	eV
BNC_BGN	$B_{n,C}$	$B_{n,C}$ coefficient	eV
CNC_BGN	$C_{n,C}$	$C_{n,C}$ coefficient	eV
ANV_BGN	$A_{n,V}$	$A_{n,V}$ coefficient	eV
BNV_BGN	$B_{n,V}$	$B_{n,V}$ coefficient	eV
CNV_BGN	$C_{n,V}$	$C_{n,V}$ coefficient	eV

6.2. Band gap and Electron Affinity

Charon computes a bandgap E_G and electron affinity χ and additionally an effective bandgap $E_{G,eff}$ and an effective electron affinity χ_{eff}. The bandgap E_G and the electron affinity χ can be material constants or temperature-dependent functions and they do not include any narrowing effects. The effective bandgap $E_{G,eff}$ and effective electron affinity use the bandgap E_G and electron affinity χ and apply bandgap narrowing effects.

To specify a constant bandgap, the user must enter in the corresponding material block section

```
start Material Block {materialBlockName}
  ...
  constant bandgap = {value}
  ...
end
```

where *value* is the bandgap value for the material in *eV*. If no bandgap is specified, Charon automatically uses the default value for the corresponding material.

To specify a constant electron affinity, the user must enter in the corresponding material block section

```
start Material Block {materialBlockName}
  ...
  electron affinity = {value}
  ...
end
```

where *value* is the electron affinity value for the material in *eV*. If no electron affinity is specified, Charon automatically uses the default value for the corresponding material.

A temperature-dependent bandgap and corresponding electron affinity are also available as

$$E_G(T) = E_G(300) + \alpha \left(\frac{300^2}{300 + \beta} - \frac{T^2}{T + \beta} \right),$$

$$\chi(T) = \chi(300) - \alpha \left(\frac{300^2}{2(300 + \beta)} - \frac{T^2}{2(T + \beta)} \right) \tag{55}$$

where $E_G(300)$, $\chi(300)$, α and β parameters are described in table 6-5.

To activate the temperature-dependent bandgap model described by (55), the user must specify the model in the corresponding material block section

```
start Material Block {materialBlockName}
  start band gap
    temperature dependence is on

    (one parameter per line in the form):
    {parameter name} = {parameter value}
    (see table 6-5 for available parameters)
  end
end
```

If a constant electron affinity has already been specified in the material block section as described above, then the constant electron affinity is used and the temperature-dependent affinity computation in (55) is neglected.

Table 6-5 Syntax and parameters for temperature-dependent bandgap.

Input file	Corresponding variable in (55)	Description	units
Eg300	$E_G(300)$	Band Gap at 300 K	eV
Chi300	$\chi(300)$	Electron Affinity at 300 K	eV
alpha	α	α coefficient	$\frac{eV}{K}$
beta	β	β coefficient	K

Internally, Charon uses the effective bandgap $E_{G,\text{eff}}$ and effective electron affinity χ_{eff} in all calculations. When bandgap narrowing is not used or turned off, $E_{G,\text{eff}} = E_G$ and $\chi_{\text{eff}} = \chi$. When bandgap narrowing is turned on in physics section block

```
start Physics Block {physicsBlockName}
  ...
  band gap narrowing is on
  ...
end
```

then narrowing effects are applied depending on the narrowing model used in the intrinsic density section (see Sec. 6.1). In this case

$$
\begin{aligned}
E_{\mathrm{G,eff}}(T) &= E_G - \Delta E_G, \\
\chi_{\mathrm{eff}}(T) &= \chi + 0.5\Delta E_G
\end{aligned}
\tag{56}
$$

for the Old Slotboom, Slotboom and Harmon bandgap narrowing models with ΔE_G being defined in (46), (47) or (48).

For Persson model

$$
\begin{aligned}
E_{\mathrm{G,eff}}(T) &= E_G - \Delta E_G, \\
\chi_{\mathrm{eff}}(T) &= \chi - \Delta E_C
\end{aligned}
\tag{57}
$$

where ΔE_C is defined in (51) or (52) and ΔE_G in (53) or (54).

7.   MOBILITY MODELS

Various models for carrier mobility, electrons and holes, are available and described here. The mobility model is specified within the specific material contained in the *Material Block* section of the input file. A synopsis of the general usage of each mobility model, along with parameters associated with a particular mobility model, are documented in subsequent sections.

7.1.   Arora Model

The Arora mobility is given by [19]

$$\mu = \mu_{\min}\left(\frac{T}{300}\right)^{\exp 1} + \frac{\mu_{\max}\left(\frac{T}{300}\right)^{\exp 2}}{1 + \left(\frac{N_D + N_A}{N_{\mathrm{ref}}\left(\frac{T}{300}\right)^{\exp 3}}\right)^{A\left(\frac{T}{300}\right)^{\exp 4}}} \tag{58}$$

for each of the carriers, holes and electrons, where T is the lattice temperature and N_D and N_A are the donor and acceptor doping levels, respectively. To specify that the Arora mobility model is utilized use

```
start Material Block {materialBlockName}
  start arora mobility
    start electron parameters
      (one parameter per line in the form):
      {parameter name} = {parameter value}
      (see table 7-1 for available parameters)
    end
    start hole parameters
      (same parameters as electron above)
    end
  end
end
```

Each of the parameters in (58) has a default value for a given material which can be overridden for each carrier via the input file. The variables and the syntax to set them in the input file is given in table 7-1 and the default values for those parameters for various materials is given in table 7-2.

67

Table 7-1 Syntax and parameters for the Arora mobility model. Note setting parameter values is optional, Charon provides defaults if a particular parameter is not specified. See (58) for parameter meanings.

Input file	Corresponding variable in (58)	Description	units
mumax	$\mu_{\max}$	reference mobility parameter	$\frac{cm^2}{Vs}$
mumin	$\mu_{\min}$	reference mobility parameter	$\frac{cm^2}{Vs}$
exp1	exp1	exponent on temperature ratio 1	*none*
exp2	exp2	exponent on temperature ratio 2	*none*
exp3	exp3	exponent on temperature ratio 3	*none*
exp4	exp4	exponent on temperature ratio 4	*none*
nref	N_{ref}	reference impurity concentration	cm^{-3}
nrefexp	A	reference impurity concentration exponent	*none*

Table 7-2 Default Arora mobility model parameter values for supported materials.

	Si	GaAs	AlGaAs	InGaAs	AlInAs	GaAsP	InGaP
Electron							
$\mu_{\max}$	1252.0	8500.0	9890.0	2.73×10^4	2.41×10^4	200.0	200.0
$\mu_{\min}$	88.0	0.0	0.0	0.0	0.0	0.0	0.0
N_{ref}	1.26×10^{17}	1.26×10^{17}	10^{20}	10^{20}	10^{20}	10^{20}	10^{20}
A	0.88	1.0	1.0	1.0	1.0	1.0	1.0
exp1	-0.57	-0.57	0.0	0.0	0.0	0.0	0.0
exp2	-2.33	0.0	0.0	0.0	0.0	0.0	0.0
exp3	2.4	0.0	0.0	0.0	0.0	0.0	0.0
exp4	-0.146	0.0	0.0	0.0	0.0	0.0	0.0
Hole							
$\mu_{\max}$	407.0	400.0	400.0	480.0	480.0	150.0	150.0
$\mu_{\min}$	54.3	0.0	0.0	0.0	0.0	0.0	0.0
N_{ref}	2.35×10^{17}	2.35×10^{17}	10^{20}	10^{20}	10^{20}	10^{20}	10^{20}
A	0.88	1.0	1.0	1.0	1.0	1.0	1.0
exp1	-0.57	0.0	0.0	0.0	0.0	0.0	0.0
exp2	-2.23	0.0	0.0	0.0	0.0	0.0	0.0
exp3	2.4	0.0	0.0	0.0	0.0	0.0	0.0
exp4	-0.146	0.0	0.0	0.0	0.0	0.0	0.0

7.2.   Albrecht Mobility Model

Monte Carlo simulations were performed by Albrecht *et al.* [20] of electron transport based upon an analytical representation of the lowest conduction band of bulk, wurtzite phase *GaN* to develop a set of transport parameters for devices with electron conduction in *GaN*. The analytic form of

the parameters determined from the Monte Carlo results are given by:

$$\frac{1}{\mu} = a\left(\frac{N_T}{N_0}\right)\left(\frac{T}{T_0}\right)^{-3/2} \ln\left(1+\beta_{CW}^2\right)$$
$$+ b\left(\frac{T}{T_0}\right)^{3/2} + \frac{c}{e^{(T_1/T)}-1}, \tag{59}$$

where

$$\beta_{CW}^2 = 3.0\left(\frac{T}{T_0}\right)^2\left(\frac{N_T}{N_0}\right)^{-2/3},$$
$$N_T = N_D + N_A.$$

For electrons, typical values of the constants in (7.2) are

$$T_1 = \frac{\hbar\omega_{LO}}{k_B} = 1065\,\text{K},$$
$$a = 2.61\times 10^{-4}\text{V\,s\,cm}^{-2},$$
$$b = 2.90\times 10^{-4}\text{V\,s\,cm}^{-2},$$

and

$$c = 1.70\times 10^{-2}\text{V\,s\,cm}^{-2}.$$

where N_D and N_A are the ionized donor concentration and acceptor concentrations in cm^{-3} and T is the lattice temperature in Kelvin.

The Albrecht mobility model is calibrated for electrons. The model should only be used for holes when a real set of parameters are defined.

The syntax for utilizing this model is

```
start Material Block {materialBlockName}
  start albrecht mobility
    start electron parameters
      (one parameter per line in the form):
      {parameter name} = {parameter value}
      (see table 7-3 for available parameters)
    end
    start hole parameters
      (same parameters as electron above)
    end
  end
end
```

Table 7-3 Syntax and parameters for the Albrecht mobility model. Note setting parameter values is optional, Charon provides defaults if a particular parameter is not specified. See (7.2) for parameter meanings.

Input file	Corresponding variable in (7.2)	Description	Units
expa	a	fit parameter 1	$\frac{Vs}{cm^2}$
expb	b	fit parameter 2	$\frac{Vs}{cm^2}$
expc	c	fit parameter 3	$\frac{Vs}{cm^2}$
expN0	N_0	reference concentration density	cm^{-3}
expT0	T_0	reference temperature	K
expT1	T_1	reference temperature in exponent	K

7.3.   Farahmand Mobility Model

In [21], Farahmand *et al.* present a comprehensive study of the transport dynamics of electrons in ternary III-nitride compounds. The work includes a field dependent mobility model and extracted parameters which are included here, respectively, in equations (60 & 61) and tables 7-5 & 7-6. The model should only be used when a real set of parameters are defined.

$$\mu_0(T,N) = \mu_{min}\left(\frac{T}{300}\right)^{\beta_1} + \frac{(\mu_{max} - \mu_{min})\left(\frac{T}{300}\right)^{\beta_2}}{1 + \left[\frac{N}{N_{\text{ref}}\left(\frac{T}{300}\right)^{\beta_3}}\right]^{\alpha(T/300)^{\beta_4}}} \tag{60}$$

In equation (60), T is temperature in Kelvin, N is the total doping density and α, β_1, β_2, β_3, β_4, μ_{min}, μ_{max}, N_{ref} are parameters that are determined either experimentally or via Monte Carlo simulation. In this case the parameters were taken from [21] in which the parameters were determined using Monte Carlo simulation.

The field dependent mobility model is given by (61) where μ_0 is the low field mobility as expressed in (60) and v^{sat}, E_c, α, n_1 and n_2 are parameters taken from [21].

$$\mu = \frac{\mu_0(T,N) + v^{\text{sat}}\frac{E^{n_1-1}}{E_c^{n_1}}}{1 + a\left(\frac{E}{E_c}\right)^{n_2} + \left(\frac{E}{E_c}\right)^{n_1}} \tag{61}$$

The syntax for utilizing this model is

```
start Material Block {materialBlockName}
  start farahmand mobility
    start electron parameters
      (one parameter per line in the form):
```

```
        {parameter name} = {parameter value}
        (see table 7-4 for available parameters)
     end
     start hole parameters
        (same parameters as electron above)
     end
  end
end
```

The table summarizing the syntax for the Farahmand mobility model is given in table 7-4 and tables with default values of the model parameters for various materials can be found in tables 7-5 and 7-6.

Table 7-4 Syntax and parameters for the Farahmand mobility model. Note setting parameter values is optional, Charon provides defaults if a particular parameter is not specified. See (60) and (61) for parameter meanings.

Input file	Corresponding variable in (60) and (61)	Description	Units
mu_1	$\mu_{\min}$	low-field mobility	$\frac{cm^2}{Vs}$
mu_2	$\mu_{\max}$	maximum mobility	$\frac{cm^2}{Vs}$
beta	β_1	temperature ratio exponent 1	*none*
delta	β_2	temperature ratio exponent 2	*none*
gamma	β_3	temperature ratio exponent 3	*none*
eps	β_4	temperature ratio exponent 4	*none*
alpha	α	exponent multiplier	*none*
ncrit	N_{ref}	reference impurity concentration value	cm^{-3}
vsat	v^{sat}	saturation velocity	$\frac{cm}{s}$
ec	E_c	critical field	$\frac{V}{cm}$
an	a	fitting parameter	*none*
n1	n_1	field ratio exponent 1	*none*
n2	n_2	field ratio exponent 2	*none*

Table 7-5 Extracted parameters for the low-field mobility model (60). The reference density is $N_{ref} = 10^{17}\,\text{cm}^{-3}$. $^{\dagger}U_{alloy} = \Delta E_c$ & $^{\ddagger}U_{alloy} = 0$.

		μ_{min} $\left(\text{cm}^2/\text{V s}\right)$	μ_{max} $\left(\text{cm}^2/\text{V s}\right)$	α	β_1	β_2	β_3	β_4
InN		774.0	3138.4	0.68	-6.39	-1.81	8.05	-0.94
$In_{0.8}Ga_{0.2}N$	†	644.3	1252.7	0.82	-1.81	-1.30	4.84	0.41
	‡	646.5	3188.7	0.66	-0.73	-3.35	2.67	0.25
$In_{0.5}Ga_{0.5}N$	†	456.4	758.1	1.04	-1.16	-1.74	2.21	-0.22
	‡	493.8	2659.9	0.66	-9.11	-2.14	7.19	-0.76
$In_{0.2}Ga_{0.8}N$	†	386.4	684.1	1.37	-1.36	-1.95	2.12	-0.99
	‡	360.9	1887.6	0.69	-0.95	-3.58	3.06	0.06
GaN		295.0	1460.7	0.66	-1.02	-3.84	3.02	0.81
$Al_{0.2}Ga_{0.8}N$	†	132.0	306.1	0.29	-1.33	-1.75	6.02	1.44
	‡	312.1	1401.3	0.74	-6.51	-2.31	7.07	-0.86
$Al_{0.5}Ga_{0.5}N$	†	41.7	208.3	0.12	-0.60	-2.08	10.45	2.00
	‡	299.4	1215.4	0.80	-5.70	-2.29	7.57	-1.08
$Al_{0.8}Ga_{0.2}N$	†	47.8	199.6	0.17	-0.74	-2.04	20.65	0.01
	‡	321.7	881.1	1.01	-1.60	-3.69	3.31	0.44
AlN		297.8	683.8	1.16	-1.82	-3.43	3.78	0.86

Table 7-6 Parameters for high field mobility model in (61).
$^{\dagger}U_{alloy} = \Delta E_c$ & $^{\ddagger}U_{alloy} = 0$.

		v^{sat} $\left(10^7\,\text{cm/s}\right)$	E_c (kV/cm)	n_1	n_2	a
InN		1.3959	52.4242	3.8501	0.6078	2.2623
$In_{0.8}Ga_{0.2}N$	†	0.8714	103.4550	4.2379	1.1227	3.0295
	‡	1.4812	63.4305	4.1330	0.6725	2.7321
$In_{0.5}Ga_{0.5}N$	†	0.7973	148.9098	4.0635	1.0849	3.0052
	‡	1.6652	93.8151	4.8807	0.7395	3.7387
$In_{0.2}Ga_{0.8}N$	†	1.0428	207.5922	4.7193	1.0239	3.6204
	‡	1.8169	151.8870	6.0373	0.7670	5.1791
GaN		1.9064	220.8936	7.2044	07857	6.1973
$Al_{0.2}Ga_{0.8}N$	†	1.1219	365.5529	5.3193	1.0396	3.2332
	‡	2.0270	245.5794	7.8138	0.7897	6.9502
$Al_{0.5}Ga_{0.5}N$	†	1.1459	455.4437	5.0264	1.0016	2.6055
	‡	2.1581	386.2440	12.5795	0.8324	8.6037
AlN		2.1670	447.0339	17.3681	0.8554	8.7253

7.4. Philips-Thomas Unified Mobility Model

The Philips unified mobility model [22],[23] is utilized heavily for simulations involving silicon bipolar devices. It's usage is described in this section.

The model segregates the mobility into four parts, the lattice, donor, acceptor and electron-hole scattering contributions. The total mobility is

$$\mu_i^{-1} = \mu_{i,L}^{-1} + \mu_{i,D}^{-1} + \mu_{i,A}^{-1} + \mu_{i,j}^{-1} \tag{62}$$

where the subscripts are

$$i \to e \text{ or } h,$$
$$j \to h \text{ or } e,$$
$$I \to A \text{ or } D,$$
$$i,j \to \text{contribution from electron-hole,}$$
$$L \to \text{contribution from lattice,}$$
$$D \to \text{contribution from donor,}$$
$$A \to \text{contribution from acceptor.}$$

The contribution from lattice scattering is

$$\mu_{i,L} = \mu_{\max} \left(\frac{T}{300.0} \right)^{\Gamma_i}. \tag{63}$$

The sum of the remaining contributions is given by

$$\mu_{i,D+A+j}(N_D, N_A, n, p) = \mu_{i,N} \frac{N_{i,\text{sc}}}{N_{i,\text{sc.eff}}} \left(\frac{N_{\text{ref}}}{N_{i,\text{sc}}} \right)^{\alpha} + \mu_{i,c} \left(\frac{n+p}{N_{i,\text{sc.eff}}} \right) \tag{64}$$

where

$$N_{e,\text{sc}} = N_D^* + N_A^* + p, \tag{65}$$
$$N_{h,\text{sc}} = N_A^* + N_D^* + n, \tag{66}$$
$$N_{e,\text{sc.eff}} = N_D^* + G(P_e)N_A^* + \frac{p}{F(P_e)}, \tag{67}$$

and

$$N_{h,\text{sc.eff}} = N_A^* + G(P_h)N_D^* + \frac{n}{F(P_h)}. \tag{68}$$

The base terms due to impurity scattering and electron-hole scattering in (64) are

$$\mu_{i,N} = \frac{\mu_{\max}^2}{\mu_{\max} - \mu_{\min}} \left(\frac{T}{300} \right)^{3\alpha - 1.5}, \tag{69}$$

$$\mu_{i,c} = \frac{\mu_{\min}\mu_{\max}}{\mu_{\max} - \mu_{\min}} \left(\frac{300}{T} \right)^{0.5}. \tag{70}$$

The functions used in (67) and (68) are

$$F_{i,j}(P_i) = \frac{0.7643P_i^{0.6478} + 2.2999 + 6.5502\frac{m_i}{m_j}}{P_i^{0.6478} + 2.3670 - 0.01552\frac{m_i}{m_j}} \tag{71}$$

$$G(P_i) = 1 - \frac{0.89223}{\left[0.41372 + \left(\frac{m_0}{m_i}\frac{T}{300}\right)^{0.28227}P_i\right]^{0.19778}} + \frac{0.005978}{\left[\left(\frac{m_i}{m_0}\frac{300}{T}\right)^{0.72169}P_i\right]^{1.80618}} \tag{72}$$

where if $P_n < P_{n,\min}$ then $G(P_{n,\min})$ is used to avoid errors, where $P_{n,\min}$ is the value of P_n where $G(P_n)$ reaches it's minimum.

$$P_i = \left[\frac{2.459}{3.97 \times 10^{13}N_{i,\mathrm{sc}}^{-2/3}} + \frac{3.828(n+p)}{1.36 \times 10^{20}\frac{m_i}{m_0}}\right]^{-1}\left(\frac{T}{300}\right)^2 \tag{73}$$

The effective impurity levels that take into account high doping effects are given by

$$N_I^* = N_I\left[1 + \frac{1}{C_I + \left(\frac{NR_I}{N_I}\right)^2}\right]. \tag{74}$$

Additionally the field dependence of the mobility due to [24] can be used and is given by

$$\mu_{i,\mathrm{hf}} = \frac{\mu_{i,\mathrm{lf}}}{\left[1 + \left(\frac{E}{v_{\mathrm{sat}}}\right)^{\beta_i}\right]^{\frac{1}{\beta_i}}} \tag{75}$$

where $\mu_{i,\mathrm{lf}}$ is the low-field mobility as calculated in (62).

The syntax for utilizing this model in Charon is

```
start material block {materialBlockName}
...
  start philips-thomas mobility
    start electron parameters
      high field mobility is on / off
      driving force is electric field / grad quasiFermi
      (one parameter per line in the form):
      {parameter name} = {parameter value}
      (see tables 7-7, 7-8 and 7-9 for available parameters)
    end
    start hole parameters
      (same parameters as electron above)
    end
  end
...
end
```

74

Table 7-7 Syntax and parameters for the Philips-Thomas mobility model.

Input file	Corresponding variable as used in section 7.4	Description	Units
exponent for mobility	α	exponent used in model	*none*
cref_a	C_A	reference acceptor density used in calculate of effective density (74)	$\frac{\#}{cm^3}$
cref_d	C_D	reference donor density used in calculate of effective density (74)	$\frac{\#}{cm^3}$
temperature dependence exponent	Γ_i	exponent for temperature dependence of lattice scattering (63)	K
maximum mobility	μ_{max}	maximum mobility	$\frac{cm^2}{V \cdot s}$
minimum mobility	μ_{min}	minimum mobility	$\frac{cm^2}{V \cdot s}$
reference impurity concentration	N_{ref}	(64)	$\frac{\#}{cm^3}$
nref_a	NR_A	(74)	$\frac{\#}{cm^3}$
nref_d	NR_D	(74)	$\frac{\#}{cm^3}$

Table 7-8 Default values for dopant-related parameters used in Philips-Thomas mobility model.

Parameter \ Dopant	As	P	B
μ_{max}	1417.0	1414.0	470.5
μ_{min}	52.2	68.5	44.9
N_{ref}	9.68×10^{16}	9.20×10^{16}	2.23×10^{17}
α	0.68	0.711	0.719

Table 7-9 Carrier dependent parameters used in Philips-Thomas mobility model.

Parameter \ i or I	h or A	e or D
Γ_i	-2.247	-2.285
C_I	0.5	0.21
NR_I	7.2×10^{20}	4.0×10^{20}
β_i	2.0	-1.0
v_{sat}	$2.4 \times 10^7/(1+0.8\exp(T/600))$	*(same)*

8.   RECOMBINATION AND GENERATION

There are currently five types of recombination and two types of generation models available within Charon. The recombination models include the mid-gap Shockley-Reed-Hall (SRH), Radiative, Auger, generic Trap SRH models and Dynamic Traps models (time-dependent). The generation models include various avalanche generation (also known as impact ionization) and optical generation models. **The first step** in utilizing these models is to turn them on in the *Physics Blocks* section of the input file as illustrated below. When these options are not specified in the input file, they are set to Off by default in the code. **The second step** is to specify the models and relevant parameters in the *Closure Models* section of the input file. Each model and corresponding parameters will be described in the subsections of this chapter.

```
start Physics Block {physicsBlockName}
   srh recombination is on/off
   Auger recombination is on/off
   Radiative recombination is on/off
   avalanche is on/off
end
```

8.1.   Mid-Gap SRH Recombination

The standard SRH recombination model within Charon is given by

$$R_{\text{SRH}} = \frac{np - n_0 p_0}{\tau_p \left(n + \sqrt{n_0 p_0}\right) + \tau_n \left(p + \sqrt{n_0 p_0}\right)}. \tag{76}$$

Here n_0 and p_0 are equilibrium electron and hole densities, respectively. In the case of Boltzmann statistics, the product $n_0 p_0$ is given by n_i^2, where n_i is the intrinsic carrier concentration in a material. This model represents the effect of mid-band gap traps.

The parameters τ_n and τ_p represent electron and hole lifetimes, respectively. They can be constant or dependent on either the dopant concentration or the temperature. For constant lifetimes you can set the values in the input file. For example

```
  start material block {materialBlockName}
   ...
   electron/hole lifetime is constant = {value}
   ...
  end
```

The lifetimes must be in units of seconds.

As noted, carrier lifetimes can also depend on the doping concentration. In this case, the lifetime is given by

$$\tau = \frac{\tau_0}{1 + \frac{N_D + N_A}{N_{srh}}}, \tag{77}$$

where N_D and N_A are the donor and acceptor concentrations, respectively, and N_{srh} is a material dependent parameter. This doping dependent lifetime can be applied to either electrons or holes. The syntax for adding such a dependency is

```
start material block
  ...
  electron/hole lifetime is concentration dependent (with nsrh = {value})
  ...
end
```

where *{value}* is the numerical value of N_{SRH} in (77). The `with nsrh` is optional and if that option is not present then a default will be used as given in table 8-1.

Another common modification to the carrier lifetime is to add a temperature dependence. In Charon, two kinds of temperature dependence are supported for the lifetime, a power law dependence where the lifetime is given by

$$\tau = \tau_0 \left(\frac{T}{300} \right)^{\alpha}, \tag{78}$$

and an exponential type of dependence where the lifetime is equal to

$$\tau = \tau_0 \exp \left[\beta \left(\frac{T}{300} - 1 \right) \right]. \tag{79}$$

The syntax for adding temperature dependence of carrier lifetimes is

```
start material block {materialBlockName}
  ...
  electron / hole lifetime is temperature dependent (with exponential variation)
  ...
end
```

The concentration and temperature dependencies can be combined, but only one of the two temperature dependencies can be specified for a given carrier. For example, you can achieve the following variation for carrier lifetime

$$\tau_n = \frac{\tau_0}{1 + \frac{N_D + N_A}{N_{srh}}} \left(\frac{T}{300} \right)^{\alpha}. \tag{80}$$

using

```
start material block {materialBlockName}
  ...
  electron lifetime is concentration dependent
  electron lifetime is temperature dependent
  ...
end
```

where default values will be used for N_{SRH}.

77

Parameter symbol	Electrons & Holes	Unit
τ_0	10^{-7}	s
N_{srh}	5.0×10^{16}	$\frac{\#}{cm^3}$
α	-1.5	1
β	2.55	1

Table 8-1 Default parameters values for doping and temperature dependent SRH lifetime.

8.2.   Radiative Recombination

The radiative (a.k.a. direct) recombination model in Charon is given by

$$R_{rad} = C(np - n_0 p_0), \tag{81}$$

where C is the radiative recombination coefficient. In the case of Boltzmann statistics, $n_0 p_0 = n_i^2$. By default, Charon uses $C = 1 \times 10^{-10}$ cm³· s⁻¹ for GaAs and 0 for silicon. For other materials, the default C values are often set to 1×10^{-10} cm³·s⁻¹ for convenience. The user is strongly recommended specifying the proper C value when using the model. Figure 8-1 shows an example of setting the coefficient of radiative recomination in the radiative recombination model, where *Coefficient* corresponds to the C value.

```
start Material Block materialBlockName
   Radiative recombination coefficient = coefficient
end Material Block materialBlockName
```

Figure 8-1 Specification of parameters values for the radiative recombination model in the input file.

8.3.   Auger Recombination

The Auger recombination model is given by

$$R_A = (C_n n + C_p p)(np - n_0 p_0), \tag{82}$$

where C_n and C_p are the electron and hole Auger recombination coefficient, respectively. Only constant Auger recombination coefficients are currently supported in Charon. The default $C_n = 2.8 \times 10^{-31}$, $C_p = 9.9 \times 10^{-32}$ cm⁶·s⁻¹ for Si, and $C_n = C_p = 1 \times 10^{-30}$ cm⁶·s⁻¹ for GaAs. For other materials, the default C_n and C_p are often set to 1×10^{-30} cm⁶·s⁻¹ for convenience. The user is strongly recommended specifying the proper parameters values when using the model. Figure 8-2 shows an example of specifying parameters values for the Auger recombination model, where *Electron Auger Coefficient* corresponds to C_n and *Hole Auger Coefficient* corresponds to C_p. By default, Charon uses Eq. (82) only if R_A is positive, and replaces the value by zero if R_A is negative. To use negative R_A, i.e., to allow for Auger generation of electron-hole pairs, one can set the Boolean parameter *With Generation* to true.

```
start Material Block materialBlockName
   start Auger Recombination Parameters
      Auger Coefficient electron/hole = augerCoefficient
      Generation on/off
   end Auger Recombination Parameters
end Material Block materialBlockName
```

Figure 8-2 Specification of parameters values for the Auger recombination model in the input file.

8.4. Generic SRH Recombination

For traps not located in the mid-band gap of a material, one can simulate their effects using a generic trap SRH recombination model available in Charon. The generic trap model is given by

$$R_{traps} = \sum_j \frac{np - n_{ie}^2}{\tau_p^j(n + n_t^j) + \tau_n^j(p + p_t^j)}, \tag{83}$$

where the summation runs over the total number of different types of traps (e.g., two traps located at two different energy levels). n_{ie} is the effective intrinsic concentration in a material. The n_t^j and p_t^j for the jth type of traps are equal to

$$n_t^j = N_C \exp\left(-\frac{E_t^j}{k_B T}\right), \quad p_t^j = N_V \exp\left(-\frac{E_g - E_t^j}{k_B T}\right), \tag{84}$$

where N_C and N_V are the effective density of states in the conduction and valence bands respectively, E_t^j is the jth trap energy measured from the conduction band edge, E_g is the effective band gap, k_B is the Boltzmann constant, and T is the lattice temperature. The lifetimes τ_n^j and τ_p^j depend on the lattice temperature and the electric field. They are given by

$$\tau_n^j(T,F) = \frac{\tau_{n0}^j}{1 + g_n^j(T,F)}, \quad \tau_p^j(T,F) = \frac{\tau_{p0}^j}{1 + g_p^j(T,F)}. \tag{85}$$

Here τ_{n0}^j and τ_{p0}^j are field-independent lifetimes, but can contain temperature dependence. $g_n^j(T,F)$ and $g_p^j(T,F)$ are the electron and hole field enhancement factors, which capture the recombination enhancement due to band-to-trap tunneling and will be described later in this section. Currently, Charon supports two ways of specifying τ_{n0}^j and τ_{p0}^j. The first one is to specify constant values in units of seconds. The second one is to compute their values using

$$\tau_{n0}^j = \frac{1}{\sigma_n^j v_n N_t^j}, \quad \tau_{p0}^j = \frac{1}{\sigma_p^j v_p N_t^j}, \quad \text{with } v_n = \sqrt{\frac{3k_B T}{m_n^*}}, \quad v_p = \sqrt{\frac{3k_B T}{m_p^*}}. \tag{86}$$

σ_n^j and σ_p^j are the electron and hole cross sections in cm^2. v_n and v_p are the electron and hole average thermal velocities [25] in cm/s. N_t^j is the trap density in cm^{-3} for the jth type of traps. m_n^* and m_p^* are the electron and hole effective masses. To compute τ_{n0}^j or τ_{p0}^j, one has to provide the corresponding cross section value.

The traps are categorized into two types: (i) *Acceptor* traps are neutral when unoccupied, and carry the charge of one electron when fully occupied; (ii) *Donor* traps are neutral when unoccupied, and carry the charge of one hole when fully occupied. The steady-state charge due to all acceptor traps is given by

$$-q\sum_j N_t^j f_{ta}^j, \text{ where } f_{ta}^j = \frac{\tau_p^j n + \tau_n^j p_t^j}{\tau_p^j(n+n_t^j)+\tau_n^j(p+p_t^j)}, \text{ the acceptor trap occupation.} \qquad (87)$$

The steady-state charge due to all donor traps is equal to

$$+q\sum_j N_t^j f_{td}^j, \text{ where } f_{td}^j = \frac{\tau_p^j n_t^j + \tau_n^j p}{\tau_p^j(n+n_t^j)+\tau_n^j(p+p_t^j)}, \text{ the donor trap occupation.} \qquad (88)$$

To include the trap charge in the Poisson equation, it must be enabled in the appropriate physics as so,

```
start physics block {PhysicsBlockName}
  trap charge is on
end physics block
```

By default, `trap charge` is off, meaning that trap charge is not included in the Poisson equation.

The field enhancement factors, g_n and g_p, with the superscript j omitted for simpler notation, are widely used to capture the stationary effects of band-to-trap tunneling (a.k.a trap assisted tunneling). That is, band-to-trap tunneling effectively reduces carrier lifetimes and consequently increases carrier recombination. Charon supports four variation forms of the Schenk band-to-trap tunneling model [26] and also a new model [27] designed for heterojunction devices. The expressions and details for g_n are given in Appendix B for different band-to-trap tunneling models. The same expressions are also applicable to g_p as long as the proper hole parameters are used. For example, one needs to replace E_t with $(E_g - E_t)$ and replace m_n^* with m_p^* in the expressions to compute g_p.

Besides the discrete-level generic traps, Charon supports continuous energy distribution traps of uniform, exponential or gaussian type, as described in table 8-2, where N_t is the peak density, $g_t(E)$ is the function describing the distribution and E_σ is the width of the distribution in eV.

Table 8-2 Available energy distribution for the Generic SRH Traps model.

Distribution Type	Description	units
uniform	$g_t(E) = N_t$ for $E_t - E_\sigma < E < E_t + E_\sigma$	$cm^{-3}ev^{-1}$
exponential	$g_t(E) = N_t exp(-\frac{|E-E_t|}{E_\sigma})$ for $E_t - E_\sigma < E < E_t + E_\sigma$	$cm^{-3}ev^{-1}$
gaussian	$g_t(E) = N_t exp(-\frac{(E-E_t)^2}{2E_\sigma^2})$ for $E_t - E_\sigma < E < E_t + E_\sigma$	$cm^{-3}ev^{-1}$

In this case the total recombination rate in (83) can be written as

$$R_{traps} = \int_{E_t-E_\sigma}^{E_t+E_\sigma} \frac{np - n_{ie}^2}{\frac{n+n_t(E)}{\sigma_p v_p g_t(E)[1+g_p(T,F)]} + \frac{p+p_t(E)}{\sigma_n v_n g_t(E)[1+g_n(T,F)]}} dE \qquad (89)$$

where $n_t = N_C exp(-\frac{E}{k_B T})$, $p_t = N_V exp(-\frac{E_g-E}{k_B T})$, σ_n and σ_p are the electron and hole cross sections in cm^2, v_n and v_p are the electron and hole average thermal velocities defined by (86) in cm/s and $g_n(T,F)$, $g_p(T,F)$ are the electron and hole field enhancement factors from (85) described above.

Similar to (87), (88) the charge and occupation probabilities for acceptor traps are given by

$$
\begin{aligned}
Q_{trapped}^A &= -q \int_{E_t-E_\sigma}^{E_t+E_\sigma} f_{ta}(E)g_t(E)dE, \\
f_{ta}(E) &= \frac{\frac{n}{\sigma_p v_p g_t(E)[1+g_p(T,F)]} + \frac{p_t(E)}{\sigma_n v_n g_t(E)[1+g_n(T,F)]}}{\frac{n+n_t(E)}{\sigma_p v_p g_t(E)[1+g_p(T,F)]} + \frac{p+p_t(E)}{\sigma_n v_n g_t(E)[1+g_n(T,F)]}}
\end{aligned}
\tag{90}
$$

and for donor traps by

$$
\begin{aligned}
Q_{trapped}^D &= q \int_{E_t-E_\sigma}^{E_t+E_\sigma} f_{td}(E)g_t(E)dE, \\
f_{td}(E) &= \frac{\frac{n_t(E)}{\sigma_p v_p g_t(E)[1+g_p(T,F)]} + \frac{p}{\sigma_n v_n g_t(E)[1+g_n(T,F)]}}{\frac{n+n_t(E)}{\sigma_p v_p g_t(E)[1+g_p(T,F)]} + \frac{p+p_t(E)}{\sigma_n v_n g_t(E)[1+g_n(T,F)]}}
\end{aligned}
\tag{91}
$$

8.4.1. *Model Usage*

Three steps of specification are needed to enable the generic trap SRH recombination model. First, the trap model must be toggled on by setting

```
start physics block {PhysicsBlockName}
  trap srh is on
end
```

in the appropriate physics blocks.

Second, if a band-to-trap tunneling model is turned on, a driving force needs to be specified in the appropriate physics block

```
start physics block {PhysicsBlockName}
  Driving Force is {drForce}
end
```

because the tunneling model depends on the field. The `drForce` can be one of the three options: `gradient quasi fermi levell`, `effective field`, or `gradient potential`. It is default to `effective field` when not given in a physics block.

Third, we need to specify the model parameters in a trap srh material block as below.

```
start material block {materialBlockName}
  start trap srh
    start trap {trapID}
      (one parameter per line in the form):
      {parameter name} = {parameter value}
```

81

```
      (see table 8-3 for available parameters)
    trap type is {type}
    energy distribution is {distType}
    spatial profile is uniform
    spatial range is {locMin} to {locMax} [ in x [ in y [ in z]]]
    start electron/hole tunneling parameters
      model is {model}
      direction [ is x [ is y [ is z ]]]
        (one parameter per line in the form):
        {parameter name} = {parameter value}
        (see table 8-4 for available parameters)
    end tunneling parameters
  end trap
 end trap srh
end material block
```

`trapID` is an integer equal or greater than 0. Multiple types of traps (up to 50) can be defined by specifying multiple `start trap` blocks, `type` defines the trap type which can be either `donor` or `acceptor`. `distType` select the energy distribution type , `spatial profile` and `spatial range` are used to specify a spatial distribution of traps. The `distType` can be `level`, `uniform`, `exponential` or `gaussian` as described in table 8-2. If the line `energy distribution is distType` is ommited, then the trap has by default a `level` distribution. When a continuous energy distribution type is used then the `energy width` line described in table 8-3 is required. Currently, only a uniform spatial profile is supported which can be placed in a user-defined box. A spatial box is defined using a `spatial range` line for each axis. The `spatial profile` line is optional to specify. When `spatial range` is not given, traps are uniformly distributed in all element blocks which belong to the same physics block.

The `tunneling parameters` block needs to be separately defined for electrons and holes. `model` specifies a tunneling model which can be one of the six options: `none`, `hightemp approx`, `lowtemp approx`, `asymptotic field`, `constant field`, `new`. `none` means no band-to-trap tunneling model is enabled. `hightemp approx` uses the high temperature approximation of the Schenk model, corresponding to Eq. (180). `lowtemp approx` uses the low temperature approximation of the Schenk model, corresponding to Eq. (182). `constant field` uses the original Schenk model, corresponding to Eq. (183) with the constant field density of states given by Eq. (187). `asymptotic field` also uses the original Schenk model, Eq. (183), but uses an asymptotic form [see Eq. (188)]for the electrooptical function. `new` uses a new band-to-trap tunneling model that was developed for heterojunction devices and takes into account both electric field and heterojunction band offset effects. The `new` model uses the new density of states model, Eq. (189), in computing the field enhancement factor. When the `new` tunneling model is enabled, three more parameters and their values should be given, including `heterojunction location`, `heterojunction band offset`, and `direction`. The `direction` keyword specifies the tunneling axis which can be `is x`, `is y`, or `is y`. Note that all the available band-to-trap tunneling models are one-dimensional model. A user needs to be aware of the most important tunneling direction.

If any of the five band-to-trap tunneling models is set for electrons or holes or both carriers, the `phonon energy` and `huang-rhys factor` parameters and their values must be specified. If the band-to-trap tunneling `model` is set to `none`, the `phonon energy` and `huang-rhys factor` parameters

are ignored. The `energy level` parameter specifies the trap energy in a positive value measured from the conduction band edge. Note that traps are located in the band gap.

The field independent lifetimes, i.e., τ_{n0} and τ_{p0} in Eq. (85), can be specified in one of the two ways. The first one is to directly specify them using the `lifetime` parameter in seconds. This is not allowed for continuous distribution. Alternatively, they can be computed using Eq. (86), where the cross sections in cm^2, σ_n and σ_p, must be given through the `cross section` parameter. The `effective mass` parameter is used to provide a value in units of m_0 (the free electron mass) for the carrier effective mass, m_n^* or m_p^*, in the trap model. When they are not given in an input file, they are taken from the default material database within *Charon* for a given material.

Table 8-3 Available parameters for the generic SRH trap model.

Input file	Corresponding variables	Description	units
energy level	E_t in Eq. (84)	trap energy level measured from conduction band edge	eV
trap density	N_t in Eq. (86), Table 8-2	trap density for level traps or peak trap density for continuous distributions	cm^{-3} $cm^{-3}eV^{-1}$
energy width	E_σ in Eqs. (89), (90), (90)	energy width for continuous distributions	eV
number of levels		number of discrete energy levels used for continuous distributions numerical integration; 20 if not specified	unitless
phonon energy	$\hbar\omega$ in Eq. (180)	optical phonon energy	eV
huang-rhys factor	S in Eq. (180)	Huang-Rhys factor	unitless

Table 8-4 Available parameters for the generic SRH band-to-trap tunneling models. m_0 is the free electron mass.

Input file	Corresponding variables	Description	units
lifetime	τ_{n0} or τ_{p0} in Eq. (85)	field independent carrier lifetime	seconds
cross section	σ_n or σ_p in Eq. (86)	carrier cross section	cm^2
effective mass	m_n^* or m_p^* in Eq. (86)	effective mass	m_0
heterojunction location	None	heterojunction location in a simulation device	μ m
heterojunction band offset	e.g., ΔE_V in figure B-2	conduction or valence band offset	eV

An example of specifying trap parameters for the InGaP emitter in an InGaP/GaAs/GaAs HBT is given below.

```
start material block InGaP-Parameter
```

```
    start trap srh
      start trap 0
        trap type is donor
        energy level = 0.93
        trap density = 1e17
        phonon energy = 0.02
        huang-rhys factor = 12.2

        start hole tunneling parameters
          model is new
          direction is x
          heterojunction location = 0.15
          heterojunction band offset = 0.442
          lifetime = 1e-9
        end

        start electron tunneling parameters
          model is none
          lifetime = 1e-9
        end
      end trap 0
    end trap srh
end material block
```

Complete examples on the generic trap and band-to-trap tunneling models can be found at
`tcad-charon/test/nightlyTests/b2ttunnel/`.

8.5.   Dynamic Traps Recombination

Another recombination mechanism handled by Charon is associated with defects in
semiconductor crystalline structure. Dynamic Traps model is an extension of the Generic SRH
Recombination (see 8.4) for modeling time-dependent deep trap states. The model explicitly
computes the capture and emission rates of the trap states with conduction and valence bands,
traps occupation probabilities, the associated bands recombinations rates due to exchange of
carriers and the density of trapped charge on the trap states.

The Dynamic Traps model allows two types of traps: acceptor and donor, and for each type, four
energy distributions: level, uniform, exponential and gaussian. The acceptor traps are neutral
when unoccupied and carry the charge of one electron when fully occupied. The donor traps carry
the charge of one hole when unoccupied and neutral when occupied. Typically, acceptor traps lie
closer to the conduction band and donor traps lie near to the valence band. An occupied acceptor
trap can emit an electron or capture a hole. An unoccupied acceptor trap can capture an electron
or emit a hole. An unoccupied donor trap can emit a hole or capture an electron. An occupied
donor trap can capture a hole or emit an electron.

The energy distributions available in the Dynamic Traps model for both acceptor and donor traps
are described in table 8-5, where N_t is density of traps for single level traps or the peak density for
continuous distributions, $g_t(E)$ is the function describing the distribution and E_σ is the width of
the distribution in eV.

Table 8-5 Available energy distribution for the Dynamic Traps model.

Distribution Type	Description	units
level	N_t for $E = E_t$	cm^{-3} or cm^{-2}
uniform	$g_t(E) = N_t$ for $E_t - E_\sigma < E < E_t + E_\sigma$	$cm^{-3}ev^{-1}$ or $cm^{-2}eV^{-1}$
exponential	$g_t(E) = N_t exp(-\frac{|E-E_t|}{E_\sigma})$ for $E_t - E_\sigma < E < E_t + E_\sigma$	$cm^{-3}ev^{-1}$ or $cm^{-2}eV^{-1}$
gaussian	$g_t(E) = N_t exp(-\frac{(E-E_t)^2}{2E_\sigma^2})$ for $E_t - E_\sigma < E < E_t + E_\sigma$	$cm^{-3}ev^{-1}$ or $cm^{-2}eV^{-1}$

The trap occupation probabilities f^n for an acceptor type or f^p for a donor type are given by the trap dynamic equations:

$$
\begin{aligned}
\frac{df^n}{dt} &= r_C^n + r_V^n, \\
\frac{df^p}{dt} &= r_C^p + r_V^p, \\
r_C^n &= (1-f^n)c_C^n - f^n e_C^n, \\
r_V^n &= (1-f^n)c_V^n - f^n e_V^n, \\
r_C^p &= (1-f^p)c_C^p - f^p e_C^p, \\
r_V^p &= (1-f^p)c_V^p - f^p e_V^p
\end{aligned} \tag{92}
$$

where r_C^n, r_V^n are the net electron capture rates due to coupling with the conduction and valence bands, r_C^p, r_V^p are the net hole capture rates due to coupling with the conduction and valence bands, f^n is the trap electron occupation probability, f^p is the trap hole occupation probability, c_C^n, c_V^n are the electron capture rates for an empty trap and e_C^n, e_V^n are the electron emission rates for a full trap.

Using equilibrium conditions ($\frac{df^n}{dt} = 0$ and $\frac{df^n}{dt} = 0$) and the principle of detailed balance, the trap capture and emission rates, the net recombination rates and occupation probabilities in equilibrium can be derived. For an acceptor trap they are given by

$$
\begin{aligned}
c_{C,A}^n &= \sigma_{n,A} v_{th}^n n, \\
e_{C,A}^n &= \frac{\sigma_{n,A} v_{th}^n \gamma_n n_{1,A}}{g_A}, \\
c_{V,A}^n &= g_A \sigma_{p,A} v_{th}^p \gamma_p p_{1,A}, \\
e_{V,A}^n &= \sigma_{p,A} v_{th}^p p, \\
r_{C,A}^n &= \frac{c_{C,A}^n(e_{C,A}^n + e_{V,A}^n) - e_{C,A}^n(c_{C,A}^n + c_{V,A}^n)}{c_{C,A}^n + e_{C,A}^n + c_{V,A}^n + e_{V,A}^n}, \\
r_{V,A}^n &= \frac{c_{V,A}^n(e_{C,A}^n + e_{V,A}^n) - e_{V,A}^n(c_{C,A}^n + c_{V,A}^n)}{c_{C,A}^n + e_{C,A}^n + c_{V,A}^n + e_{V,A}^n}, \\
f_{eqn}^n &= \frac{c_{C,A}^n + c_{V,A}^n}{c_{C,A}^n + e_{C,A}^n + c_{V,A}^n + e_{V,A}^n}
\end{aligned} \tag{93}
$$

and for a donor trap by

$$
\begin{aligned}
c_{V,D}^p &= \sigma_{p,D} v_{th}^p p, \\
e_{V,D}^p &= g_D \sigma_{p,D} v_{th}^p \gamma_p p_{1,D}, \\
c_{C,D}^p &= \frac{\sigma_{n,D} v_{th}^n \gamma_n n_{1,D}}{g_D}, \\
e_{C,D}^p &= \sigma_{n,D} v_{th}^n n, \\
r_{C,D}^p &= \frac{c_{C,D}^p (e_{C,D}^p + e_{V,D}^p) - e_{C,D}^p (c_{C,D}^p + c_{V,D}^p)}{c_{C,D}^p + e_{C,D}^p + c_{V,D}^p + e_{V,D}^p}, \\
r_{V,D}^p &= \frac{c_{V,D}^p (e_{C,D}^p + e_{V,D}^p) - e_{V,D}^p (c_{C,D}^p + c_{V,D}^p)}{c_{C,D}^p + e_{C,D}^p + c_{V,D}^p + e_{V,D}^p}, \\
f_{eqn}^p &= \frac{c_{C,D}^p + c_{V,D}^p}{c_{C,D}^p + e_{C,D}^p + c_{V,D}^p + e_{V,D}^p}
\end{aligned}
\tag{94}
$$

where $\sigma_{n,A}$, $\sigma_{p,A}$ are the acceptor trap electron and hole capture cross sections, $\sigma_{p,D}$, $\sigma_{n,D}$ are the donor trap electron and hole capture cross sections, g_A, g_D are the acceptor trap and donor trap degeneracy factors, γ_n, γ_p are the Fermi-Dirac degeneracy factors, $n_{1,A} = n_i exp(\frac{E_{t,A} - E_i}{k_B T})$, $p_{1,A} = n_i exp(\frac{E_i - E_{t,A}}{k_B T})$, $p_{1,D} = n_i exp(\frac{E_i - E_{t,D}}{k_B T})$, $n_{1,D} = n_i exp(\frac{E_{t,D} - E_i}{k_B T})$ and v_{th}^n, v_{th}^p are the electron and hole thermal velocities. Although the capture and emission rates were derived for equilibrium, it is assumed that they are valid for transient states too.

In order to solve for the transient behavior of trap states and compute the trapped charge and the carrier exchange rates with the bands, the dynamic trap equations in Eq. (92) are discretized in time

$$
\begin{aligned}
\frac{f^n(t + \delta t) - f^n(t)}{\delta t} &= r_C^n(t + \delta t) + r_V^n(t + \delta t), \\
\frac{f^p(t + \delta t) - f^p(t)}{\delta t} &= r_C^p(t + \delta t) + r_V^p(t + \delta t)
\end{aligned}
\tag{95}
$$

to obtain analytical relations for the trap occupation probabilities. The trap occupation probability and the net recombination rate as a function of time and occupation probability at a previous time are then given by

$$
\begin{aligned}
f^n(t_k) &= \frac{f^n(t_{k-1}) + (t_k - t_{k-1})(c_{C,A}^n(t_k) + c_{V,A}^n(t_k)}{1 + (t_k - t_{k-1})(c_{C,A}^n(t_k) + e_{C,A}^n(t_k) + c_{V,A}^n(t_k) + e_{V,A}^n(t_k))}, \\
r_{C,A}^n(t_k) &= (1 - f^n(t_k))c_{C,A}^n(t_k) - f^n(t_k)e_{C,A}^n(t_k), \\
r_{V,A}^n(t_k) &= (1 - f^n(t_k))c_{V,A}^n(t_k) - f^n(t_k)e_{V,A}^n(t_k)
\end{aligned}
\tag{96}
$$

for acceptor traps, and by

$$
\begin{aligned}
f^p(t_k) &= \frac{f^p(t_{k-1}) + (t_k - t_{k-1})(c_{V,D}^p(t_k) + c_{C,D}^p(t_k)}{1 + (t_k - t_{k-1})(c_{V,D}^p(t_k) + e_{V,D}^p(t_k) + c_{C,D}^p(t_k) + e_{C,D}^p(t_k))}, \\
r_{V,D}^p(t_k) &= (1 - f^p(t_k))c_{V,D}^p(t_k) - f^p(t_k)e_{V,D}^p(t_k), \\
r_{C,D}^p(t_k) &= (1 - f^p(t_k))c_{C,D}^p(t_k) - f^p(t_k)e_{C,D}^p(t_k)
\end{aligned}
\tag{97}
$$

86

for donor traps.

The recursive relations (96) and (97) allow Charon to compute the trap occupation probability and the net recombination rate for a trap if f^n, $r_{C,A}^n$, $r_{V,A}^n$ for an acceptor trap or f^p, $r_{V,D}^p$, $r_{C,D}^p$ for a donor trap are known at $t = 0$. As an initial guess for the recursive equations (96) and (97) Charon uses equilibrium relations (93), (94).

The conduction band net recombination rates due to acceptor trap A_i and donor trap D_k at time t are

$$
\begin{aligned}
R_{A_i}^{net}(t) &= N_{t,A_i} r_{C,A_i}^n(t) = N_{t,A_i}[(1 - f_{A_i}^n(t))c_{C,A_i}^n(t) - f_{A_i}^n(t)e_{C,A_i}^n(t)], \\
R_{D_k}^{net}(t) &= -N_{t,D_k} r_{C,D_k}^p(t) = N_{t,D_k}[f_{D_k}^p(t)e_{C,D_k}^p(t) - (1 - f_{D_k}^p(t))c_{C,D_k}^p(t)]
\end{aligned}
\tag{98}
$$

When multiple acceptor and donor dicrete level traps are present, the total conduction band net recombination rate becomes

$$
\begin{aligned}
R_C^{net}(t) = \sum_{A_i} &\{N_{t,A_i}[(1 - f_{A_i}^n(t))c_{C,A_i}^n(t) - f_{A_i}^n(t)e_{C,A_i}^n(t)]\} + \\
\sum_{D_k} &\{N_{t,D_k}[f_{D_k}^p(t)e_{C,D_k}^p(t) - (1 - f_{D_k}^p(t))c_{C,D_k}^p(t)]\}
\end{aligned}
\tag{99}
$$

Similarly, the total valence band net recombination rate is given as

$$
\begin{aligned}
R_V^{net}(t) = \sum_{D_k} &\{N_{t,D_k}[(1 - f_{D_k}^p(t))c_{V,D_k}^p(t) - f_{D_k}^p(t)e_{V,D_k}^p(t)]\} + \\
\sum_{A_i} &\{N_{t,A_i}[f_{A_i}^n(t)e_{V,A_i}^n(t) - (1 - f_{A_i}^n(t))c_{V,A_i}^n(t)]\}
\end{aligned}
\tag{100}
$$

The net electron recombination rate in (99) is added as a recombination term in electron drift diffusion equation and the net hole recombination rate in (100) is added as a recombination term to the hole drift diffusion equation.

The total trapped charge is computed by summing over the charge trapped on each individual trap species:

$$
Q_{trapped}^{net}(t) = q(\sum_{D_k} f_{D_k}^p N_{t,D_k} - \sum_{A_i} f_{A_i}^n N_{t,A_i})
\tag{101}
$$

Charon adds total trapped charge $Q_{trapped}^{net}$ to the right-hand side of Poisson equation to properly account for the total charge in the device.

In the case of continuous distributions (uniform, exponential and gaussian) defined by Table 8-5 the recombination rates given by (99) and (100) change to

$$
\begin{aligned}
R_C^{net}(t) = \int_{E_t - E_\sigma^A}^{E_t + E_\sigma^A} &g_t^A(E)[(1 - f_A^n(E,t))c_{C,A}^n(t) - f_A^n(E,t)e_{C,A}^n(E,t)]dE + \\
\int_{E_t - E_\sigma^D}^{E_t + E_\sigma^D} &g_t^D(E)[f_D^p(E,t)e_{C,D}^p(E,t) - (1 - f_D^p(E,t))c_{C,D}^p(t)]dE
\end{aligned}
\tag{102}
$$

and

$$R_V^{net}(t) = \int_{E_t-E_\sigma^D}^{E_t+E_\sigma^D} g_t^D(E)[(1-f_D^p(E,t))c_{V,D}^p(t) - f_D^p(E,t)e_{V,D}^p(E,t)]dE +$$

$$\int_{E_t-E_\sigma^A}^{E_t+E_\sigma^A} g_t^A(E)[f_A^n(E,t)e_{V,A}^n(E,t) - (1-f_A^n(E,t))c_{V,A}^n(t)]dE \qquad (103)$$

and the total trapped charge $Q_{trapped}^{net}$ in (101) becomes

$$Q_{trapped}^{net}(t) = q(\int_{E_t-E_\sigma^D}^{E_t+E_\sigma^D} f_D^p(E,t)g_t^D(E)dE - \int_{E_t-E_\sigma^A}^{E_t+E_\sigma^A} f_A^n(E,t)g_t^A(E)dE) \qquad (104)$$

To activate/use Dynamic Traps model in semiconductor bulk, the user must first set the *dynamic traps* flag to *on* in the *Physics Block* as shown below

```
start Physics Block {physicsBlockName}
  ...
  dynamics traps is on
  ...
end
```

Secondly, the user must specify the Dynamic Traps model along with its relevant parameters in the *Material Block* section of the input file associated with the *Physics Block* where the model has been turned on:

```
start material block {materialBlockName}
  start dynamic traps
    start trap {trapID}
      trap type is {trapType}
      energy distribution is {enDistr}
      (one parameter per line in the form):
      {parameter name} = {parameter value}
      (see table 8-6 for available parameters)
      spatial range is {locMin} to {locMax} [ in x [ in y [ in z]]]
      thermal velocity calculation is {thVelType}
      number of levels = {NL}
    end trap
  ...
  end dynamic traps
  ...
end material block
```

In the user input above `trapID` is an integer equal or greater than 0. Multiple traps (up to 50) can be defined by specifying multiple `start trap` blocks. `trapType` defines the trap type which can be either `donor` or `acceptor`. The trap energetic distribution is defined by the parameter `enDistr` which can be `level`, `uniform`, `exponential` or `gaussian`. If `energy distribution` line is omitted, the `enDistr` is assumed to be of `level` type. Each indiviual trap defined by `trapID` can be spatially confined in a user-defined box, or distributed throughout the entire material block where the trap has been defined. To limit the trap spatial distribution along an axis the user must specify `spatial range` for

88

that particular axis. A spatial box is defined using a `spatial range` line for each axis. If no `spatial range` line is specified then is assumed that the trap is distributed uniformly throughout the entire material block. The thermal velocities used in calculation of capture and emission rates (see (93) and (93)) can defined as mean velocities $v_{th}^n = \sqrt{\frac{8k_BT}{\pi m_e}}$, $v_{th}^p = \sqrt{\frac{8k_BT}{\pi m_h}}$ or as root mean square velocities $v_{th}^n = \sqrt{\frac{3k_BT}{\pi m_e}}$, $v_{th}^p = \sqrt{\frac{3k_BT}{\pi m_h}}$. The user can select the thermal velocity calculation by setting `thVelType` to `mean` for mean velocity or to `root mean square` for root mean square velocity. For continuous distribution traps, Charon discretize internally the enegy distribution in equally spaced energy intervals. By default, the number of energy levels is set to 20. The user can change the number of energy levels by setting the parameter `NL`. If the line `number of levels` is omitted, then the default value is used. The rest of the parameters are described in table 8-6

Table 8-6 Available bulk parameters for the Dynamic Traps model.

Input file	Corresponding variables	Description	units
energy level	$E_{t,A}$, $E_{t,D}$ for level traps or E_t for continuous distribution traps	trap energy level measured from conduction band edge for acceptor traps or from valence band edge for donor traps; for continuous distributions E_t is the energy center	eV
trap density	$N_{t,A}$, $N_{t,D}$ for level traps or N_t for continuous distribution traps	trap density for level traps or peak trap density for continuous distributions	cm^{-3} or $cm^{-3}ev^{-1}$
energy width	E_σ in Eq. (102) (103) (101) valid for continuous distributions only	for continuous distributions, energy spread around energy peak	ev
degeneracy factor	g_A, g_D in Eq. (93) and Eq. (94)	acceptor or donor trap degeneracy	1
electron cross section	$\sigma_{n,A}$, $\sigma_{n,D}$ in Eq. (93) and Eq. (94)	acceptor or donor electron cross section	cm^2
hole cross section	$\sigma_{p,D}$, $\sigma_{p,A}$ in Eq. (93) and Eq. (94)	acceptor or donor hole cross section	cm^2

An example of specifying bulk trap parameters for part of a silicon block in given below:

```
start Material Block {siliconParameter}
  ...
  start dynamic traps
    start trap 0
      trap type is acceptor
      energy distribution is uniform
      energy level = 0.54
      energy width = = 0.2
      trap density = 1.0e13
      degeneracy factor = 2.0
      electron cross section = 1.7e-16
```

89

```
      hole cross section = 1.1e-14
      # confine acceptor traps to n-region
      spatial range is 0.0 to 1.0 in x
      spatial range is 0.0 to 1.0 in y
      thermal velocity calculation is mean
      number of levels = 20
    end
  end
  ...
end
```

To visualize the total electron and hole recombination rates and total electron and hole trapped charge (summation over all trap species in a certain block) the user must specify in the `start output parameters` block at the root level of input file

```
start output parameters
  ...
  output cell average variables in {semicBlock} for scalar "Dynamic Traps eRecombination,
      Dynamic Traps hRecombination,Electron Trapped Charge,Hole Trapped Charge,Trapped
      Charge"
  ...
end
```

where `semicBlock` is the semiconductor block where dynamic traps are located.

To activate/use Dynamic Traps model at an semiconductor/insulator interface defined as a sideset, the user must specify

```
start dynamic traps bc for {sidesetName}
  geometry block is {geometryBlock}
  start dynamic traps
    start trap {trapID}
      trap type is {trapType}
      energy distribution is {enDistr}
      (one parameter per line in the form):
      {parameter name} = {parameter value}
      (see table 8-6 for available parameters)
      spatial range is {locMin} to {locMax} [ in x [ in y [ in z]]]
      thermal velocity calculation is {thVelType}
      number of levels = {NL}
    end
    ...
  end
  ...
end
```

where `trapID` is an integer equal or greater than 0 allowing multiple traps (up to 50) to be defined by specifying multiple `start trap` blocks, `geometryBlock` is the semiconductor geometry block of the interface and all the other parameters similar to those described for dynamic traps in bulk.

Charon also allows field-dependent capture cross sections for Dynamic Traps in bulk or on interface. Two models are available for cross sections dependence on electric field, Kimpton and

saturation. The Kimpton cross section field-dependence are expressed as

$$\sigma(E) = \begin{cases} (\frac{3\sqrt{\pi}}{4N_t})^{3/2} & for\ E \leq 10^6[\frac{\sigma^{1\frac{MV}{cm}}}{(\frac{3\sqrt{\pi}}{4N_t})^{3/2}}]^{1/x} \\ \sigma^{1\frac{MV}{cm}}(\frac{E}{10^6\frac{V}{cm}})^{-x} & for\ E > 10^6[\frac{\sigma^{1\frac{MV}{cm}}}{(\frac{3\sqrt{\pi}}{4N_t})^{3/2}}]^{1/x} \end{cases} \tag{105}$$

for bulk traps and as

$$\sigma(E) = \begin{cases} \frac{1}{N_t} & for\ E \leq 10^6[N_t\sigma^{1\frac{MV}{cm}}]^{1/x} \\ \sigma^{1\frac{MV}{cm}}(\frac{E}{10^6\frac{V}{cm}})^{-x} & for\ E > 10^6[N_t\sigma^{1\frac{MV}{cm}}]^{1/x} \end{cases} \tag{106}$$

for interface traps. For the saturation model, the cross sections field-dependence for both bulk and interface are given by

$$\sigma(E) = \begin{cases} \sigma^{1\frac{MV}{cm}} & for\ E < 10^6 \\ \sigma^{1\frac{MV}{cm}}(\frac{E}{10^6\frac{V}{cm}})^{-x} & for\ E \geq 10^6 \end{cases} \tag{107}$$

In Eqns. (105), (106) and (107) above $E = -\nabla\phi$, N_t is the dynamic traps density in bulk or on interface, $\sigma^{1\frac{MV}{cm}}$ is the capture cross section at $E = 1.0\ MV/cm$, x is the power dependency of the cross section on the electric field.

To activate/use the field-dependent capture cross sections, the user must specify inside the start trap blocks the power dependency factors:

```
start trap {trapID}
  ...
  electron electric field power dependency = {ePowerDep}
  hole electric field power dependency = {hPowerDep}
  ...
end
```

where ePowerDep and hPowerDep are positive numbers.

To select a dependency model, besides the power dependency line, the user has to select the model with electron electric field power dependency and hole electric field power dependency lines:

```
start trap {trapID}
  ...
  electron electric field power dependency = {ePowerDep}
  hole electric field power dependency = {hPowerDep}
  electron field dependence is {eDepModel}
  hole field dependence is {hDepModel}
  ...
end
```

where `eDepModel` and `hDepModel` can be either `kimpton` or `saturation`. If any of the line `electron field dependence` or `hole field dependence` is not specified, then automatically the `saturation` model is selected. The electron and hole capture cross sections can be made field-dependent independently. For example, if only `electron electric field power dependency` line is specified, then the trap will have a field dependent electron cross section and a constant hole cross section.

The syntax described above is valid for both bulk and inteface dynamic traps.

Complete examples on dynamic trap recombination model can be found at `tcad-charon/test/nightlyTests/dynamic_traps/`.

8.6.   Avalanche Generation

A few models for avalanche generation (impact ionization) are available in *Charon*. The generation rate of electron-hole pairs produced by these models is described by

$$G \;=\; \alpha_n(F_n)\frac{|\mathbf{J}_n|}{q} + \alpha_p(F_p)\frac{|\mathbf{J}_p|}{q} \tag{108}$$

where $\alpha_n(F_n)$ and $\alpha_p(F_p)$ are the electron and hole ionization coefficients, F_n, F_p are electron and hole driving forces and $\mathbf{J}_e$, $\mathbf{J}_h$ are the electron and hole current densities.

An avalanche generation model in Charon computes the electron and hole ionization coefficients $\alpha_n(F_n)$ and $\alpha_p(F_p)$ as functions of driving forces F_n and F_p.

To activate/use an avalanche model, the user must first set the *avalanche* flag to *on* and specify the *driving force* in the *Physics Block* as shown below

```
start Physics Block {physicsBlockName}
  ...
  avalanche is on
  driving force is {drForce}
  ...
end
```

where *drForce* can be one of three options: *effective field*, *gradient potential* or *gradient quasi fermi level*. By default *driving force* is set to *effective field* when it is not given in the input file.

Secondly, the user must specify the avalanche model used along with its relevant parameters in the *Material Block* section of the input file

```
start Material Block {materialBlockName}
  ...
  start avalanche generation
    threshold behavior model is {avalancheModel}
    driving force is {drForceType}
    minimum field = {minField}
    start {avalancheModel} parameters
      critical field is fixed
      (one parameter per line in the form):
```

92

```
      {parameter name} = {parameter value}
   end
  end
  ...
end
```

Charon implements two avalanche models, Selberherr and Crowell-Sze, which can be selected by *avalancheModel* above. To select Selberherr model *avalancheModel* must be set to *selberherr*. For Crowell-Sze model *avalancheModel* must be set to *crowell-sze*.

The driving force *drForceType* above must match the *drForce* in the *Physics Block* section. If *drForce* in the *Physics Block* is *effective field* then *drForceType* can be either *effective field J* or *effective field Jtot*. If *drForce* is *gradient potential* then *drForceType* can be either *gradient potential J* or *gradient potential Jtot*. Finally, if *drForce* is *gradient quasi fermi level* then *drForceType* must be *gradient quasi fermi level*.

minField sets the minimum value of the electric field from which the avalanche model is turn on internally.

More details about the avalanche model specification will be described in the following sections.

8.6.1. Selberherr Model

Selberherr avalanche generation model computes the electron and hole ionization coefficients $\alpha_n(F_n)$ and $\alpha_p(F_p)$ in Eq.(108) as

$$
\begin{aligned}
\alpha_n &= \alpha_n^\infty(T)\exp\left[-\left(\frac{F_n^{crit}}{F_n}\right)^{\delta_n}\right], \\
\alpha_p &= \alpha_p^\infty(T)\exp\left[-\left(\frac{F_p^{crit}}{F_p}\right)^{\delta_p}\right]
\end{aligned}
\tag{109}
$$

The critical fields F_n^{crit} and F_p^{crit} in Eq. (109) can be set to a fixed value specified in the input file or computed using

$$
\begin{aligned}
F_n^{crit} &= \frac{E_g(T)}{q\lambda_n(T)}, \\
F_p^{crit} &= \frac{E_g(T)}{q\lambda_p(T)}.
\end{aligned}
\tag{110}
$$

where λ_n and λ_p are the optical-phonon mean free paths for electrons and holes, given by

$$
\begin{aligned}
\lambda_n(T) &= \lambda_n^0\tanh\left(\frac{\hbar\omega_n}{2k_BT}\right), \\
\lambda_p(T) &= \lambda_p^0\tanh\left(\frac{\hbar\omega_p}{2k_BT}\right)
\end{aligned}
\tag{111}
$$

93

The coefficients α_n^∞ and α_n^∞ are polynomial functions of temperature

$$\begin{aligned}
\alpha_n^\infty(T) &= a_{0n} + a_{1n}T + a_{2n}T^2 \\
\alpha_p(T) &= a_{0n} + a_{1n}T + a_{2n}T^2
\end{aligned} \tag{112}$$

with the constant coefficients a_{0n}, a_{1n}, a_{2n}, a_{0p}, a_{1p} and a_{2p} specified by the user.

To improve numerical stability, the driving fields, F_n and F_p in Eq. (109) can be computed as $\frac{nF_n}{n+n_{ref}}$ and $\frac{pF_p}{p+p_{ref}}$ where n, p are the electron and hole densities and n_{ref}, p_{ref} are numerical damping density parameters. Using positive values for n_{ref} and p_{ref} may improve convergence for problems where strong generation-recombination occurs in regions with small carrier densities.

To use Selberherr avalanche model the user must first turn the avalanche on and specify the driving force in the *Physics Block*

```
start Physics Block {physicsBlockName}
  ...
  avalanche is on
  driving force is {drForce}
  ...
end
```

Secondly, the user must specify Selberherr avalanche model along with parameters in the *Material Block* section of the input file

```
start Material Block {materialBlockName}
  ...
  start avalanche generation
    threshold behavior model is selberherr
    driving force is {drForceType}
    minimum field = {minField}
    electron driving force reference density = {eDFRefDens}
    hole driving force reference density = {hDFRefDens}
    start selberherr parameters
      critical field is {critField}
      (one parameter per line in the form):
      {parameter name} = {parameter value}
      (see table 8-7 for available parameters)
    end
  end
  ...
end
```

For a description of *drForceType* and *minField* see 8.6. Critical field can be either specified in the input file or computed. To specify critical field in the input file set *critField* to *fixed*. To force computing critical field according to Eq. (110) set *critField* to *computed*. The other Selberherr avalanche model parameters and their description are listed in table 8-7.

If *electron driving force reference density* and/or *hole driving force reference density* are specified in the list of avalanche generation parameters (see above) then the driving fields, F_n and F_p are

computed as $\frac{nF_n}{n+n_{ref}}$ and $\frac{pF_p}{p+p_{ref}}$ where n, p are the electron and hole densities and n_{ref}, p_{ref} are numerical damping density parameters which can be set by the user (*eDFRefDens* and *hDFRefDens*).

Table 8-7 Syntax and parameters for the Selberherr avalanche model.

Input file	Corresponding variable in (109), (110), (111), (112)	Description	units
electron a0	a_{0n}	α_n^∞ coefficient	$\frac{1}{cm}$
electron a1	a_{1n}	α_n^∞ coefficient	$\frac{1}{cmK}$
electron a2	a_{2n}	α_n^∞ coefficient	$\frac{1}{cmK^2}$
hole a0	a_{0p}	α_p^∞ coefficient	$\frac{1}{cm}$
hole a1	a_{1p}	α_p^∞ coefficient	$\frac{1}{cmK}$
hole a2	a_{2p}	α_p^∞ coefficient	$\frac{1}{cmK^2}$
electron delta	δ_n	electron field ratio exponent	*none*
hole delta	δ_p	hole field ratio exponent	*none*
electron E0	F_n^{crit}	electron critical field when *critField* is *fixed*	$\frac{V}{cm}$
hole E0	F_p^{crit}	hole critical field when *critField* is *fixed*	$\frac{V}{cm}$
electron lambda300	λ_n^0	Phonon mean free path for electrons at 300K	*cm*
hole lambda300	λ_p^0	Phonon mean free path for hole at 300K	*cm*
electron h bar omega	$\hbar\omega_n$	Optical phonon energy for electrons	*eV*
hole h bar omega	$\hbar\omega_p$	Optical phonon energy for holes	*eV*

8.6.2. *Crowell-Sze Model*

Crowell-Sze avalanche generation model [28] estimates the electron and hole ionization coefficients $\alpha_n(F_n)$ and $\alpha_p(F_p)$ in Eq.(108) based on Baraff's theory [29] in terms of physical parameters $E_{opt,ph}$, the Raman optical phonon energy, E_{ioniz}, the ionization energy and λ, the carrier free path for optical generation. The ionization coefficients are computed as

$$\begin{aligned}
\alpha_n &= \frac{1}{\lambda_n} exp\left[C_0(r_n) + C_1(r_n)x_n + C_2(r_n)x_n^2\right], \\
\alpha_p &= \frac{1}{\lambda_p} exp\left[C_0(r_p) + C_1(r_p)x_p + C_2(r_p)x_p^2\right]
\end{aligned} \qquad (113)$$

where the coefficients C_0, C_1 and C_2 are given by the polynomials

$$\begin{aligned}
C_0(r) &= -1.92 + 75.5r - 757r^2, \\
C_1(r) &= 1.75 \times 10^{-2} - 11.9r + 46r^2, \\
C_2(r) &= 3.9 \times 10^{-4} - 1.17r + 11.5r^2
\end{aligned} \tag{114}$$

and

$$\begin{aligned}
r_n &= \frac{E_{opt,ph}}{E_{ioniz}^n}, \\
r_p &= \frac{E_{opt,ph}}{E_{ioniz}^p}, \\
x_n &= \frac{E_{ioniz}^n}{q\lambda_n F_n}, \\
x_p &= \frac{E_{ioniz}^p}{q\lambda_p F_p}
\end{aligned} \tag{115}$$

Similar to Selberherr model (see Eq.(112)), the carrier free paths depend on Raman optical phonon energy

$$\begin{aligned}
\lambda_n(T) &= \lambda_n^0 \frac{\tanh\left(\frac{E_{opt,ph}}{2k_B T}\right)}{\tanh\left(\frac{E_{opt,ph}}{2k_B 300K}\right)}, \\
\lambda_p(T) &= \lambda_p^0 \frac{\tanh\left(\frac{E_{opt,ph}}{2k_B T}\right)}{\tanh\left(\frac{E_{opt,ph}}{2k_B 300K}\right)}
\end{aligned} \tag{116}$$

To use Crowell-Sze avalanche model the user must first turn the avalanche on and specify the driving force in the *Physics Block*

```
start Physics Block {physicsBlockName}
  ...
  avalanche is on
  driving force is {drForce}
  ...
end
```

Secondly, the user must specify Crowell-Sze avalanche model along with parameters in the *Material Block* section of the input file

```
start Material Block {materialBlockName}
  ...
  start avalanche generation
    threshold behavior model is crowell-sze
    driving force is {drForceType}
    minimum field = {minField}
    electron driving force reference density = {eDFRefDens}
    hole driving force reference density = {hDFRefDens}
```

```
    start crowell-sze parameters
       (one parameter per line in the form):
       {parameter name} = {parameter value}
       (see table 8-8 for available parameters)
    end crowell-sze parameters
  end
  ...
end
```

For a description of *drForceType* and *minField* see 8.6. The other Crowell-Sze avalanche model parameters and their description are listed in table 8-8.

If *electron driving force reference density* and/or *hole driving force reference density* are specified in the list of avalanche generation parameters (see above) then the driving fields, F_n and F_p are computed as $\frac{nF_n}{n+n_{ref}}$ and $\frac{pF_p}{p+p_{ref}}$ where n, p are the electron and hole densities and n_{ref}, p_{ref} are numerical damping density parameters which can be set by the user (*eDFRefDens* and *hDFRefDens*).

Table 8-8 Syntax and parameters for the Crowell-Sze avalanche model.

Input file	Corresponding variable in (113), (114), (115), (116)	Description	units
electron lambda300	λ_n^0	Phonon mean free path for electrons at 300K	cm
hole lambda300	λ_p^0	Phonon mean free path for hole at 300K	cm
optical phonon energy	$E_{opt,ph}$	Raman optical phonon energy	eV
electron ionization energy	E_{ioniz}^n	ionization energy for electrons	eV
hole ionization energy	E_{ioniz}^p	ionization energy for holes	eV

8.7. Optical Generation

The effect of electron-hole pair generation due to photon absorption and/or ionizing radiation can be modeled by adding optical generation to the carrier continuity equations. Charon supports analytical and tabulated optical generation profiles. Any number of analytical and tabulated profiles can be specified, and are added together when they are located at the same position. To enable optical generation, we must first turn it on in the *Physics Block*

```
start Physics Block {physicsBlockName}
  ...
  optical generation is on
  ...
end
```

8.7.1.   *Tabulated Optical Generation*

The simplest tabulated optical generation profile is a two-column table read from an external file. The first column is the time in seconds, while the second column is the optical generation rate in $cm^{-3}.s^{-1}$. Such a time-dependent optical generation profile can be applied to an entire simulation domain or a user-specified spatial region. When there is only one time-dependent optical generation file, we can use this syntax

```
start Material Block {materialBlockName}
  ...
  start optical generation named Function{i}
    read optical generation from {temporalFile}
    spatial range is {min} to {max} in x / y / z
  end
  ...
end
```

Here `i` is an integer number, `temporalFile` is a time-dependent optical generation file, `min` and `max` are the minimum and maximum coordinate values. If the optical generation is applied to an entire simulation domain, the `spatial range` line can be removed. However, when there are multiple time-dependent files to be used in a simulation, we need to use

```
start Material Block {materialBlockName}
  ...
  start optical generation named Function{i}
    read temporal file from {temporalFile}
    spatial range is {min} to {max} in x / y / z
  end
  ...
end
```

The second approach can be expanded to include a tabulated spatial dependence, which is either 1D or 2D. A 1D space dependent profile is a two-column table, where the first column contains position values in μm and the second column is unitless factors. These spatial dependent factors are multiplied to the time-dependent optical generation rates to produce spatially dependent optical generation profile. This feature is used via

```
start Material Block {materialBlockName}
  ...
  start optical generation named Function{i}
    read temporal file from {temporalFile}
    read 1d spatial file from {space1DFile}
    spatial direction is in x / y / z
    spatial range is {min} to {max} in x / y / z
  end
  ...
end
```

Here `space1DFile` is a 1D space dependent file, `spatial direction` specifies the axis direction along which the coordinate values in `space1DFile` are located, `spatial range` defines the spatial range along the axes that are perpendicular to the `spatial direction` axis. A 2D space dependent

profile is a three-column table, where the first two columns are the position values in μm, while the third column is unitless factors. To multiply 2D spatial factors to time-dependent optical generation rates, one can use the following syntax

```
start Material Block {materialBlockName}
  ...
  start optical generation named Function{i}
    read temporal file from {temporalFile}
    read 2d spatial file from {space2DFile}
    spatial buffer = {value}
  end
  ...
end
```

where `spatial buffer` is to expand the spatial range contained in `space2DFile` by an amount of `value` in μm along both x and y axes.

An example of reading multiple time- and space-dependent optical generation files is given below

```
start Material Block SiliconParameter
    start optical generation named Function1
        read temporal file from timeDepOpt.txt
        read 1d spatial file from space1DFile1.txt
        spatial direction is in y
        spatial range is 0.0 to 5.0 in x
    end optical generation named Function1

    start optical generation named Function2
        read temporal file from timeDepOpt.txt
        read 1d spatial file from space1DFile2.txt
        spatial direction is in y
        spatial range is 5.0 to 10.0 in x
    end optical generation named Function2

    start optical generation named Function3
        read temporal file from timeDepOpt.txt
        read 2d spatial file from space2DFile.txt
        spatial buffer = 0.05
    end optical generation named Function3

    start optical generation named Function4
        read temporal file from timeDepOpt.txt
        spatial range is 0.0 to 1.0 in x
        spatial range is 0.5 to 1.5 in y
    end optical generation named Function4
  end Material Block SiliconParameter
```

8.7.2.   *Analytical Optical Generation*

In addition to a tabulated radiation pulse as outlined in Section 8.7.1, a radiation pulse for optical generation may be specified analytically. As of this version, only a Gaussian temporal profile is

implemented. The syntax to specify an analytical pulse appears in the same type of material block as the tabulated input and the same spatial extents for the pulse are likewise honored. The full specification for an analytical pulse requires the function to be named as Gauss or Gaussian, the time of the peak of the pulse, time extents of the pulse, the maximum and optionally the minimum of the magnitude of the pulse and the width of the Gaussian.

```
start Material Block {materialBlockName}
  ...
  start optical generation named Function{i}
    temporal function is gauss
    temporal range is {startTime} to {endTime}
    maximum pulse...
    minimum pulse...
    pulse peak time = {peakTime}
  end
  ...
end
```

The specification of the `temporal range` specifies hard start and hard stop times of the generation of carriers and has no bearing on the functional form of the Gaussian. It will cut off the tails of the Gaussian if it is desired. The specification of `pulse peak time` is the time at which the Gaussian hits its peak value.

The magnitude of the pulse can be specified in two ways; it can be either the maximum and minimum of the pulse in terms of the total number of electron-hole pairs created per unit volume per second,

```
maximum pulse rate = {maxPulseRate}
minimum pulse rate = {minPulseRate}
```

with the maximum occurring at the peak time and the minimum the asymptotic value at infinite time, or it can be specified in units of rad(Si),

```
maximum pulse rate RadSi = {maxPulseRate}
minimum pulse rate RadSi = {minPulseRate}
```

with the same implications for maximum and minimum.

The width of the Gaussian can likewise be specified in two ways. The most common way to specify a Gaussian pulse for radiation is by the full-width, half max of the pulse. In other words, it is the full time width of the pulse at the midpoint between the minimum and maximum magnitudes of the pulse:

```
pulse time FWHM = {pulseWidth}
```

Alternatively, the time width of the pulse may be specified by a more generic time pulse time width,

```
pulse time width = {pulseWidth}
```

which is approximately equal to $4.71 \times full-width-half-max$.

The time integrator will automatically discretize the pulse in time to ensure that the pulse is accurately captured in the integration. By default, the pulse between +/- 5 standard deviations of the peak will be discretized into 25 time steps. This may be modified in one of two ways. To specify a number of guaranteed time steps iver the pulse, use the line:

```
pulse time integrator steps = {pulseDiscretization}
```

where `pulseDiscretization` is the number of *guaranteed* time steps within the pulse. A second way to do this is to specify the maximum step time step size taken through the pulse:

```
Pulse maximum time step size = {timestep}
```

where the `timestep` is an evenly space time between time integrator points through the pulse. Note that the integrator may capture more points than this if it is required to meet error tolerances specified in the time integrator.

8.8.   Band-to-Band Tunneling

Band-to-band tunneling (BTBT) is an important effect in highly doped and/or high field regions in semiconductor devices, especially in tunneling field effect transistors (TFET) where current conduction relies on BTBT. In principle, BTBT is a nonlocal quantum tunneling process [30], which is very challenging to model within any MPI (Message Passing Interface) parallel TCAD device simulator. Over the years, researchers have developed simplified BTBT models [31][32][33] that depend on local electric fields and are easy to be implemented in TCAD codes. Charon currently supports two such local models for modeling BTBT.

The first one is the well-known model by Kane [31],

$$G_{bbt} = D\frac{A}{\sqrt{E_g}}F^{\gamma}\exp\left(-E_g^{3/2}\frac{B}{F}\right), \tag{117}$$

where G_{bbt} is the band-to-band tunneling generation rate in unit of $\mathrm{cm}^{-3}.\mathrm{s}^{-1}$, E_g is the band gap in eV, F is the electric field magnitdue in F/cm, A, B, and γ are user-defined parameters which are often used as fitting parameters. It was found [32] that $\gamma = 2$ for direct transitions and $\gamma = 2.5$ for indirect transitions including phonon assisted band-to-band tunneling. The D factor is given by [32]

$$D = \frac{n_{ie}^2 - np}{(n+n_{ie})(p+n_{ie})}(1-|\alpha|) - \alpha, \tag{118}$$

where n_{ie} is the effective intrinsic concentration, n and p are the electron and hole concentration, respectively, with all concentrations in unit of cm^{-3}. The value of α can be 0, -1, or 1. When $\alpha = 1$, we have $D = -1$, $G_{bbt} < 0$, which indicates the BTBT is a pure recombination process. When $\alpha = -1$, we have $D = 1$, $G_{bbt} > 0$, which indicates the BTBT is a pure generation process. When $\alpha = 0$, we have

$$D = \frac{n_{ie}^2 - np}{(n+n_{ie})(p+n_{ie})}, \tag{119}$$

101

which can be either positive or negative, indicating that the BTBT can be a either generation or recombination process.

The second supported BTBT local model was proposed by Hurkx *et al.* [32],

$$G_{bbt} = DAF^{\gamma}\exp\left(-\frac{B}{F}\right). \tag{120}$$

This model is essentially identical to Kane's model, except lumping the band-gap contribution into the adjustable parameters.

The *D* expression in Eq. (119) was derived under multiple assumptions including the Boltzmann statistics assumption. At the thermal equilibrium condition and under the Boltzmann statistics assumption, the $n_{ie}^2 - np$ is equal to zero, which ensures zero BTBT rate at equilibrium as expected physically. However, due to the Boltzmann statistics and other assumptions, the *D* expression is not sufficient to ensure zero BTBT rate at equilibrium in simulations with high doping and/or high field regions. Additionally, since the *D* expression introduces more nonlinearlity in the system of transport equations, it may cause difficult conergence problem in certain simulations. To address this issue, we have followed the empirical approach by Hiu-Yong *et al.* [34] and introduced an additional field factor as given below

$$G_{bbt} = \left(\frac{|F - F_0|}{F_0}\right)^{\beta} \alpha A F^{\gamma}\exp\left(-\frac{B}{F}\right). \tag{121}$$

Here α is either 1 or -1, and β is a user-defined parameter. F_0 is the simulated, position-dependent electric field (magnitude) in a device at a zero-current condition. For field effect transistors, a zero-current condition includes any non-zero gate voltage condition. The same additional field factor is also applicable to the Kane model in Eq. (117) with the *D* factor removed.

To use the BTBT model in a Charon simulation, we need to first turn on the model and its driving force in a physics block as follows,

```
start physics block {PhysicsBlockName}
  band2band tunneling is on
  driving force is {drForce}
end physics block
```

The `drForce` can be one of the three options: `gradient potential`, `effective field`, or `gradient quasi fermi level`. It is default to `effective field` when not given in a physics block. The `drForce` is the electric field strength used in the model. Next we specify the model and its relevant parameters in a material block shown below,

Charon currently supports either `Hurkx` or `Kane` model. `drForce` here needs to loosely correspond to the `drForce` given a physics block. Specifically, if `drForce` = `gradient potential` in a physics block, then `drForce` here can be one of three options: `grad potential`, `grad potential parallel J`, or `grad potential parallel Jtot`. Similary, if `drForce` = `effective field` given in a physics block, then `drForce` here needs to be one of three options: `effective field`, `effective field parallel J`, or `effective field parallel Jtot`. The last possibility is that, if `drForce` = `grad quasi fermi level` in a physics block, then `drForce` in the model block is `grad quasi fermi`. When `driving force` is

```
start Material Block {materialBlockName}
   start band2band tunneling
      behavior model is Hurkx / Kane
      driving force is {drForce}
      min field = {Fmin}
      Hurkx / Kane A = {valueA}
      Hurkx / Kane B = {valueB}
      Hurkx / Kane alpha = {alpha}
      Hurkx / Kane beta = {beta}
      Hurkx / Kane gamma = {gamma}
      spatial range is {locMin} to {locMax} in x / y / z
   end band2band tunneling
 end Material Block {materialBlockName}
```

Figure 8-3 Specification of the BTBT local model and its relevant parameters in the input file.

not specified by a user, it defaults to effective field. The min field provides a field value in V/cm below which the band-to-band tunneling rate is set to 0. When it is not given by a user, it defaults to 10^3 V/cm. The spatial range is to limit the BTBT model to a user-defined box region. The default values for other parameters are listed in Table 8-9. It's noted that, to make the Hurkx and the Kane models produce similar results for a device, one needs to modify their *A* and *B* values. For example, since the band gap of silicon is about 1 eV, we need to change Kane A and Kane B values to match the values in the Hurkx model to achieve similar results for the two models. In addition, to use the modified model in Eq. (121), one needs to (i) specify the value of beta in the input file, (ii) set alpha to -1 or 1, (iii) save the equilibrium potential gradient obtained from an input Exodus file to be used in the modified BTBT calculation. The latter is done by setting save initial potential gradient is true in a material block.

Table 8-9 Default parameters values for the band-to-band tunneling local models.

Parameter Name	Default Value	Unit
Hurkx A	4.0×10^{14}	$1/(\text{cm}.\text{V}^2.\text{s})$ if gamma=2.0; $1/(\text{cm}^{\frac{1}{2}}.\text{V}^{\frac{5}{2}}.\text{s})$ if gamma=2.5
Hurkx B	1.9×10^7	V/cm
Hurkx alpha	0 or -1 or +1	1
Hurkx beta	0.0	1
Hurkx gamma	2.5 (indirect) or 2.0 (direct)	1
Kane A	3.5×10^{21}	$(\text{eV})^{\frac{1}{2}}/(\text{cm}.\text{V}^2.\text{s})$ if gamma=2.0; $(\text{eV})^{\frac{1}{2}}/(\text{cm}^{\frac{1}{2}}.\text{V}^{\frac{5}{2}}.\text{s})$ if gamma=2.5
Kane B	2.25×10^7	$\text{V}/(\text{cm}.(\text{eV})^{\frac{3}{2}})$
Kane alpah	0 or -1 or +1	1
Kane beta	0.0	1
Kane gamma	2.5 (indirect) or 2.0 (direct)	1

Below is an example of using the Kane model with the additional field factor in Eq. (121).

Complete examples on the band-to-band tunneling local models can be found at
`tcad-charon/test/nightlyTests/b2btunnel/`.

```
start Material Block siliconParameter
   start band2band tunneling
      behavior model is Kane
      driving force is grad potential
      min field = 1e5
      Kane A = 4.15e14
      Kane B = 1.69e7
      Kane alpha = -1.0
      Kane beta = 1.5
      Kane gamma = 2.5
      spatial range is 0.23 to 0.26 in x
   end band2band tunneling
   save initial potential gradient is true
end Material Block siliconParameter
```

Figure 8-4 An example of using the Kane model.

9.　　INCOMPLETE IONIZATION

In silicon the energy levels of dopants are shallow compared to the thermal energy $k_B T$ at room temperature. At room temperature, the dopants in silicon can be regarded for practical purposes as fully ionized. For materials where the dopant levels are deep or for silicon at low temperatures, the dopant species are only partially ionized and incomplete ionization must be accounted for in order to determine the correct effective doping concentrations.

9.1.　　Model Implementation

The densities of the ionized dopants are derived based on the Fermi-Dirac distribution of the dopant energy level:

$$
\begin{aligned}
N_D^+ &= \frac{N_{D,0}}{1 + g_D \exp\left(\frac{E_{Fn} - E_D}{k_B T}\right)} \quad N_{D,0} \leq N_{D,\text{crit}}, \\
N_A^- &= \frac{N_{A,0}}{1 + g_A \exp\left(\frac{E_A - E_{Fp}}{k_B T}\right)} \quad N_{A,0} \leq N_{A,\text{crit}}.
\end{aligned}
\tag{122}
$$

where N_D^+ is the ionized donor density, N_A^- is the ionized acceptor density, $N_{D,0}$ and $N_{A,0}$ are the substitutional dopant densities, g_D, g_A are the dopant energy level degeneracy factors, E_D, E_A are the donor and acceptor ionization energies and $N_{D,\text{crit}}$ and $N_{A,\text{crit}}$ are the dopant critical densities above which the dopants are considered fully ionized.

Using the Fermi-Dirac expressions for the carrier densities expressed as pertubations in Maxwell Boltzmann approximation:

$$
\begin{aligned}
n &= N_C \mathcal{F}_{1/2}(\eta_n) = \gamma_n N_C \exp(\eta_n), \\
p &= N_V \mathcal{F}_{1/2}(\eta_p) = \gamma_p N_V \exp(\eta_p), \\
\eta_n &= (E_{Fn} - E_C)/k_B T, \\
\eta_p &= (E_V - E_{Fp})/k_B T.
\end{aligned}
$$

where γ_n and γ_p are the electron and hole degeneracy factors, the ionized dopant densities in (122) are expressed as a function of Charon solution variables (carrier concentrations):

$$
\begin{aligned}
N_D^+ &= \frac{N_{D,0}}{1 + g_D \frac{n}{\gamma_n n_1}} \quad N_{D,0} \leq N_{D,\text{crit}}, \\
N_A^- &= \frac{N_{A,0}}{1 + g_A \frac{p}{\gamma_p p_1}} \quad N_{A,0} \leq N_{A,\text{crit}}.
\end{aligned}
\tag{123}
$$

where $n_1 = N_C \exp\left(-\frac{\Delta E_D}{k_B T}\right)$ and $p_1 = N_V \exp\left(-\frac{\Delta E_A}{k_B T}\right)$ are two computational energy levels depending on ΔE_D, the energy difference between conduction band edge and donor energy level

and respectively ΔE_A, the energy difference between the acceptor energy level and the valence band edge.

For the case where the Maxwell-Boltzmann approximation is used instead of Fermi-Dirac integral for computing the carrier densities, $\gamma_n = 1$ and $\gamma_p = 1$ and (123) degenerates into:

$$
\begin{aligned}
N_D^+ &= \frac{N_{D,0}}{1 + g_D \frac{n}{n_1}} \quad N_{D,0} \leq N_{D,\text{crit}}, \\
N_A^- &= \frac{N_{A,0}}{1 + g_A \frac{p}{p_1}} \quad N_{A,0} \leq N_{A,\text{crit}}.
\end{aligned}
\tag{124}
$$

When incomplete ionization models are enabled (for donor, acceptor or both), Charon uses ionized dopant densities defined by (123) and (124) instead of the substitutional dopant densities ($N_{D,0}$ and $N_{A,0}$) for all closure models except mobility, intrinsic density and SRH lifetime models. For mobility, intrinsic density and SRH lifetime models the fully ionized dopant densities (substitutional densities) $N_{D,0}$ and $N_{A,0}$ are always used.

For contacts, ionized dopant densities are also used instead of substitutional dopant densities $N_{D,0}$ and $N_{A,0}$ when the corresponding ionization model is turned on. In the most general case, with the incomplete ionization, the charge neutrality and equilibrium conditions at the contact become:

$$
\begin{aligned}
n_0 - p_0 &= N_D^+ - N_A^- + N_{\text{ion}} = C, \\
n_0 p_0 &= \gamma_n \gamma_p n_i^2.
\end{aligned}
\tag{125}
$$

where N_{ion} is ion charge (zero if the ionic transport is not used).

Depeding on the contact type (n-type or p-type) and the statistics used for carrier densities (Maxwell-Boltzmann or Fermi-Dirac) implicit algebraic equations of order 4, 3 or 2 have to be solved for carrier densities at the contacts. Charon allows three types of approximations for minority carrier dopant in charge neutrality condition in (125). In the first case, the minority carrier dopant is assumed to be fully ionized, $N_D^+ = N_{D,0}$ for a p-type contact or $N_A^- = N_{A,0}$ for a n-type contact. In the second case, the minority carrier dopant is ignored, $N_D^+ \simeq 0$ for a p-type contact or $N_A^- \simeq 0$ for a n-type contact. Finally, for the third case, the minority carrier dopant can be speciifed by the exact formula in (124).

9.2.   Model Usage

The incomplete ionization model can be activated independently for donor and acceptor species. To activate the ionization model for either carrier first the appropriate `incomplete ionization` parameter has to be set to `on` in the relevant `physics block` section of the input file. For example,

```
start physics block {physics name}
  .
  acceptor incomplete ionization is on
  donor incomplete ionization is on
  .
end
```

Next the blocks setting parameters for the incomplete ionization, for each carrier that was enabled in the physics block, must be added to the relevant `material block` section of the input file. For example,

```
start material block {materialBlockName}
  .
  start incomplete ionization acceptor
    (one parameter per line in the form):
    {parameter name} = {parameter value}
    (see table 9-1 for a list of parameters)
  end
  .
  start incomplete ionization donor
    (same parameter set as acceptor)
  end
  .
end
```

Table 9-1 Syntax and parameters for the incomplete ionization model.

Input file	Corresponding variables in (122) through (125)	Description	Units
critical doping	$N_{D,\text{crit}}$ or $N_{A,\text{crit}}$	doping value	$\frac{\#}{\text{cm}^2}$
degeneracy factor	g_D or g_A	dopant energy level degeneracy factors	*none*
ionization energy	E_D or E_A	ionization energies	*eV*
file	*none*	existing file containg doping versus ionization energy data	*none*
approximation	*none*	I - fully ionized II - no ionization III - use full model	*none*

As an option, the ionization energies E_D and E_A can be defined as doping-dependent quantities rather than constant values. The ionization energies versus doping concentration are specified in two separate files with two columns, the first column being the doping concentration in cm^{-3} and second the corresponding ionization energy for that doping in eV. Internally, for a certain raw doping concentration (fully ionized) the ionization energy used in incomplete ionization model is computed based on the tabulated data defined in the file. If the raw doping concentration matches a value in the file then the corresponding ionization energy is used. If the doping level is outside the doping range specified in the file then the closest doping level from file is taken and the corresponding ionization is used. For the other values of the raw doping concentration within the range defined in the file a logarithmic interpolation on doping concentration scale is used to

compute the corresponding ionization energy. If using the `file` option Charon expects the file to be a standard text file with the following format:

$$\text{Doping}_1 \quad E_{\text{ioniz},1}$$
$$\text{Doping}_2 \quad E_{\text{ioniz},2}$$
$$.$$
$$.$$
$$.$$
$$\text{Doping}_n \quad E_{\text{ioniz},n}$$

10.   MOLE-FRACTION DEPENDENT MATERIALS

Charon has added support for binary and ternary compound semiconductors with the composition defined by the user-specified mole fractions. Three material properties, bandgap, relative permittivity and effective density of states are allowed to have all parameters mole-fraction depedent. Although Charon provide a general framework for mole-fraction dependence of binary and ternary compound materials, for the time being only one binary material $Si_{1-x}Ge_x$ and one ternary material $Al_xGa_{1-x}N$ are available for use.

10.1.   Model Implementation

For binary compound materials, the mole-fraction dependent parameters are computed as a function of the mole fraction x and their values for side materials (for x = 0 and x = 1). If p is a mole-fraction dependent parameter, $p = p(x)$ and p_A and p_B are the values of the parameter on the side materials A and B then Charon interpolates p from the side material values. For a binary compound of the type $A_{1-x}B_x$ the parameter p is computed as:

$$p(x) = (1-x)p_A + xp_B + bx(x-1) \tag{126}$$

where b is a parameter defined by the user which is by default zero. Default values are used for side parameters p_A and p_B unless specified by the user in the input file. If parameter b is zero, $p(x)$ becomes a linear interpolation of the side parameters. It should be noted that for $x = 0$ and $x = 1$ the parameter p takes the values of the side materials, $p(0) = p_A$ and $p(1) = p_B$. Calculation of the mole-fraction parameter $p(x)$ in Eq. (126) is valid for any binary compound provided that the mole fraction multiplying each side material corresponds to the mole fraction associated with the side material in the definition of the compound. A binary compound can be one of the following types: $A_{1-x}B_x$ or A_xB_{1-x}.

For ternary compound materials, the mole-fraction dependent parameters are computed as a function of the mole fraction x and their values for side materials (for x = 0 and x = 1). If $A_xB_{1-x}C$ is a ternary compound material composed of the side materials AC and BC and p_{AC}, p_{BC} are the values of the parameter p for the side materials AC, BC then $p(x)$ is computed as:

$$p(x) = xp_{AC} + (1-x)p_{BC} + bx(x-1) + cx(x^2-1) \tag{127}$$

where b and c are parameters defined by the user which are by default zero. Default values are used for side parameters p_{AC} and p_{BC} unless specified by the user in the input file. If parameters b and c are zero, $p(x)$ becomes a linear interpolation of the side parameters p_{AC} and p_{BC}. For $x = 0$ and $x = 1$ the parameter p takes the values of the side materials, $p(0) = p_{BC}$ and $p(1) = p_{AC}$. Calculation of the mole-fraction parameter $p(x)$ in Eq. (127) is valid for any ternary compound provided that the mole fraction multiplying each side material corresponds to the mole fraction associated with the side material in the definition of the compound. A ternary compound can be one of the following types: $A_xB_{1-x}C$ or $A_{1-x}B_xC$ or $AB_{1-x}C_x$ or AB_xC_{1-x}.

The mole-fraction x has a spatial dependence and is a number between 0.0 and 1.0. Charon allows a few spatial profiles for the mole-fraction x: uniform, linear, erfc, gaussian, mgaussian and halo.

10.2. Model Usage

To activate the mole-fraction dependence of the materials available ($Si_{1-x}Ge_x$ and $Al_xGa_{1-x}N$ for now), first, the the material name of the mole-fraction dependent compound has to be specified in the `material block` section of the input file. For instance, to use $Si_{1-x}Ge_x$ material in a device, the user must specify in the input file:

```
start material block {materialBlockName}
  .
  material name is Si(1-x)Ge(x)
  .
end
```

Secondly, the mole-fraction spatial profile or profiles throughout the device must be provided in the `material block` section of the input file. For instance, to define a uniform mole-fraction with the value 0.5 in a square region from 0.0 to 0.3 *μm* in x direction and from 0.0 to 0.6 *μm* in y direction the user must specify:

```
start material block {materialBlockName}
  .
  material name is Al(x)Ga(1-x)N
  .
  start uniform mole fraction Function1
     xmole value = 0.5
     spatial range is 0.0 to 0.3 in x
     spatial range is 0.0 to 0.6 in y
  end
  .
end
```

In order to make the parameters $E_G(300)$, α, β and $\chi(300)$ defining the bandgap and affinity (see Section 6.2) mole-fraction dependent, the user must specify the interpolating coefficients *b* for binary compounds and *b* and *c* for ternary compounds in the `mole fraction parameters` block of the `band gap` block

```
start material block {materialBlockName}
  .
  material name is Al(x)Ga(1-x)N
  .
  start band gap
    start mole fraction parameters
      start Eg300
        b = val1
        c = val2
      end Eg300
      start Chi300
        b = val3
        c = val4
      end Chi300
      start alpha
        b = val5
        c = val6
```

110

```
      end alpha
      start beta
        b = val7
        c = val8
      end beta
    end mole fraction parameters
  end band gap
  .
  end
```

If the parameters b and c are not specified then by default are zero.

Relative permittivity can be made mole-fraction dependent and interpolated from its side materials by specifying the interpolating coefficients b for binary compounds and b and c for ternary compounds in the `mole fraction parameters` block of the `relative permittivity` block

```
  start material block {materialBlockName}
    .
  material name is Al(1x)Ga(1-x)N
    .
  start relative permittivity
    start mole fraction parameters
      start value
        b = val1
        c = val2
      end value
    end mole fraction parameters
  end relative permittivity
    .
  end
```

If the parameters b and c are not specified then by default are zero.

The effective electron and hole densities of states given by $N_C = N_{C,300}\left(\frac{T}{300}\right)^{N_{C,F}}$ and $N_V = N_{V,300}\left(\frac{T}{300}\right)^{N_{V,F}}$ can be made mole-fraction depedent through the parameters N_C, $N_{C,F}$, N_V, $N_{V,F}$. To interpolate N_C, $N_{C,F}$, N_V, $N_{V,F}$ parameters from their side materials, the user must specify the interpolating coefficients b for binary compounds and b and c for ternary compounds in the `mole fraction parameters` block of the `effective DOS` block:

```
  start material block {materialBlockName}
    .
  material name is Al(x)Ga(1-x)N
    .
  start effective DOS
    start mole fraction parameters
      start Nc300
        b = val1
        c = val2
      end Nc300
      start Nc_F
        b = val3
        c = val4
      end Nc_F
```

```
      start Nv300
        b = val5
        c = val6
      end Nv300
      start Nv_F
        b = val7
        c = val8
        end Nv_F
    end mole fraction parameters
  end effective DOS
  .
end
```

If the parameters *b* and *c* are not specified then by default are zero.

For all available mole-fraction dependent quantities (bandgap, relative permittivity, effective densities of states), the side materials values of the mole-fraction parameters are taken internally from Charon. For instance, in the case of $Al_xGa_{1-x}N$, the permittivity for $x = 0$ (GaN) is 8.9 and for $x = 1$ (AlN) is 8.5, the default values for GaN and respectively AlN. The user can change the default values of the side materials parameters in the the input file by specifying the parameter values at $x = 0$ and $x = 1$. For instance, to change $E_G(300)$ parameter of the bandgap to 3.4 for GaN and 6.2 for AlN the user must specify:

```
start material block {materialBlockName}
  .
  material name is Al(x)Ga(1-x)N
  .
  start band gap
    start mole fraction parameters
      .
      start Eg300
        b = val1
        c = val2
      end Eg300
      .
      eg300 for x at 0 is 3.4
      eg300 for x at 1 is 6.2
    end mole fraction parameters
  end band gap
  .
end
```

Mole-fraction spatial profiles are similar to doping profiles (see §16.7) expect that *concentration* line is replaced with *xmole value* and there is no *type* line. To define a mole-fraction with a *erfc* profile in $Si_{1-x}Ge_x$ with a peak at $x = 0.5$, negative direction, x from 0.0 to 1.0 μm, y from 0.0 to 0.5 μm and the mole fraction variation from 0.001 to 1.0, the user must specify:

```
start material block {materialBlockName}
  .
  material name is Si(1-x)Ge(x)
  .
  start erfc mole fraction named Function1
    gradient center = 0.5 in x
```

```
      gradient width = 0.5 in x
      direction is Negative in x
      spatial range is 0.0 to 1.0 in x
      spatial range is 0.0 to 0.5 in y
      x mole fraction range is 0.001 to 1.0
    end
    .
  end
```

When defining multiple profiles, the total mole fraction at a spatial point will the sum of the individual mole-fractions ath that location. The total mole fraction must be less or equal to 1.0.

11.   QUANTUM MODELS

11.1.   Density Gradient Model

As the features of the modern devices becomes smaller and smaller, the wave nature of the charge carriers cannot be neglected any longer. To simulate the effects of quantum mechanical confinement, Charon introduces the density gradient model which accounts for the quantization effects using potential-like perturbations in the classical expressions of electron and hole densities:

$$
\begin{aligned}
n &= N_C \mathcal{F}_{1/2}\left(\frac{E_{Fn} - E_C - q\Lambda_n}{k_B T}\right), \\
p &= N_V \mathcal{F}_{1/2}\left(\frac{E_V - E_{Fp} - q\Lambda_p}{k_B T}\right),
\end{aligned}
\tag{128}
$$

where Λ_n and Λ_p are the electron and hole quantum correction potentials derived in [35] and given by

$$
\begin{aligned}
\Lambda_n &= -\frac{\hbar^2}{12 m_n m_0 kT}\left[\Delta\left(\psi + \gamma_n \Lambda_n\right) + \frac{1}{2kT}\left(\nabla\psi + \gamma_n \nabla\Lambda_n\right)^2\right], \\
\Lambda_p &= \frac{\hbar^2}{12 m_p m_0 kT}\left[\Delta\left(\psi - \gamma_p \Lambda_p\right) - \frac{1}{2kT}\left(\nabla\psi - \gamma_p \nabla\Lambda_p\right)^2\right],
\end{aligned}
\tag{129}
$$

with γ_n and γ_p fitting parameters.

Typical applications for the density gradient model are simulations of threshold voltage shifts and reductions in gate capacitance in HEMT devices and thin-gate oxide MOSFETs.

Charon implements the density gradient model by solving for quantum correction potentials in Eqn. (129) and correcting the drift parts of the current density terms given by Eq. (9) of the drift diffusion equations in [Eqn. (8)]. The corrected current densities are then given by:

$$
\begin{aligned}
\mathbf{J}_n &= -qn\mu_n \nabla\left(\psi + \Lambda_n\right) + qD_n \nabla n, \\
\mathbf{J}_p &= -qp\mu_p \nabla\left(\psi - \Lambda_p\right) - qD_p \nabla p,
\end{aligned}
\tag{130}
$$

To activate/use density gradient model, the user must specify the *quantum correction* block with its relevant parameters in the *Physics Block* section of the input file.The electron density correction is turned on with the line `electron density correction is on`. Similarly, the hole density correction is turned on with the line `hole density correction is on`. Simplified versions of the model with the quadratic terms (field terms) in Eqns. ((129) neglected are possible by specifying the line `electron simplified formulation is on` for electron and the line `hole simplified formulation is on` for hole. Currently, the density gradient model only works for *drift diffusion cvfem* discretization.

```
start Physics Block {physicsBlockName}
  standard discretization type is drift diffusion cvfem
  ...
  start quantum correction
     electron density correction is on
     hole density correction is on
     electron simplified formulation is on
     (one parameter per line in the form):
     (see table 11-1 for available parameters)
  end quantum correction
  ...
end
```

Table 11-1 Available parameters for the Density Gradient Model.

Input file	Default Value	Corresponding Variable	Description	Unit
electron fit parameter	0.3	γ_n	fitting parameter	1
hole fit parameter	0.3	γ_p	fitting parameter	1
electron effective mass	1.08	m_n	effective mass	1
hole effective mass	0.81	m_p	effective mass	1

12.   HETEROJUNCTION

Carrier transport across a heterojunction (HJ) interface in a semiconductor heterostructure such as heterojunction bipolar transistor (HBT) requires special treatment. Conventional transport equations become invalid at a HJ, and the current at an abrupt HJ between two materials should be defined by interface condition at the junction.

Taking the InGaP/GaAs/GaAs NPN HBT illustrated in Fig. 12-1(a) as an example, the corresponding band diagram in the emitter and base regions is sketched in Fig. 12-1(b). Due to the wider band gap of InGaP than GaAs, the emitter-base (E-B) junction has discontinuities in both the conduction and valence bands, which cause discontinuities in electron and hole densities at the junction. Current across the E-B junction is determined by electron forward injection from the emitter to the base through the thermionic emission and tunneling processes and hole backward injection from the base to the emitter through the thermionic emission process. Given the valence band profile in Fig. 12-1(b), there is no hole tunneling across the E-B junction. Currently, Charon supports thermionic emission (TE) and local tunneling (LT) models for electrons and holes at a HJ.
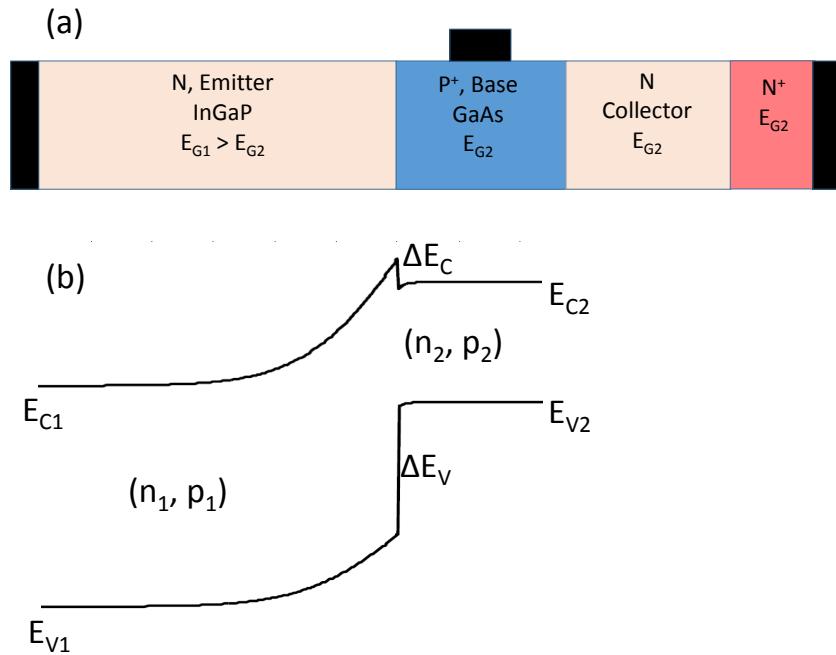
Figure 12-1 (a) Schematic of a InGaP/GaAs/GaAs NPN HBT device. (b) Band diagram in the emitter and base regions.

Implementation of the TE and LT models in Charon requires the support of discontinuities in carrier densities and material properties at an abrupt HJ. This is made possible by Panzer, the Trilinos package on which Charon is built. Panzer provides the heterojunction infrastructure that allows for adding different suffix to the carrier densities and material properties names for the two sides at a HJ, which enables Charon to access carrier densities and material properties of both sides. For example, given the E-B junction shown in Fig. 12-1, the emitter side is defined as side

1 and the electron (hole) density on side 1 is denoted as n_1 (p_1), while the base side is defined as side 2 and the electron (hole) density on side 2 is denoted as n_2 (p_2). It is noted that since the collector has the same material as the base, there is no carrier discontinuity at the base-collector junction, hence the electron (hole) density in collector is also denoted as n_2 (p_2). For simpler implementation, the left side of a HJ is always defined as side 1, while the right side of a HJ is always defined as side 2. The conduction band offset is defined as $\Delta E_C = E_{C1} - E_{C2}$ where E_{C1} (E_{C2}) is the conduction band of side 1 (side 2), whereas the valence band offset is defined as $\Delta E_V = E_{V2} - E_{V1}$ where E_{V2} (E_{V1}) is the valence band of side 2 (side 1). It is noted that ΔE_C and ΔE_V can be either positive or negative depending on the band diagram, as described in Sec. 12.1

The TE and LT models are based on the foundational work by Wu and Yang [36]. In the following, the final forms of the models are described, while derivation of the models is detailed in Appendix C.

12.1. Thermionic Emission

Given the conduction and valence band diagram in Fig. 12-1(b), the net electron and hole current densities due to thermionc emission across the E-B junction are given by [see the Eq. (191)]

$$
\begin{aligned}
J_{TE,n} &= A_n^* T^2 \left[-\exp\left(\frac{E_{Fn1} - E_{C1}}{k_B T} \right) + \exp\left(\frac{E_{Fn2} - E_{C2} - \Delta E_C}{k_B T} \right) \right], \\
J_{TE,p} &= A_p^* T^2 \left[\exp\left(\frac{E_{V1} - E_{Fp1}}{k_B T} \right) - \exp\left(\frac{E_{V2} - E_{Fp2} - \Delta E_V}{k_B T} \right) \right].
\end{aligned}
\tag{131}
$$

Where A_n^* and A_p^* are the electron and hole Richardson constants, E_{Fn} and E_{Fp} are the electron and hole quasi-Fermi levels, E_C and E_V are the conduction and valence band edges, respectively, k_B is the Boltzmann constant, T is the temperature, 1 and 2 denote side 1 and side 2. Note that ΔE_C and ΔE_V here are both positive. The current densities here are scalar quantities and represent the normal components perpendicular to the HJ. The Richardson constants are defined as

$$
\begin{aligned}
A_n^* &= \frac{4\pi q k_B^2 m_n^*}{h^3}, \\
A_p^* &= \frac{4\pi q k_B^2 m_p^*}{h^3},
\end{aligned}
\tag{132}
$$

where m_n^* and m_p^* are the electron and hole effective mass, respectively, and h is the Planck constant. There is an inconsistency in literature [36, 37, 38] regarding which side of effective mass should be used in computing the Richardson constant. In Charon, the effective mass used in computing the Richardson constant is not taken from either side of a HJ, but set to be a user-defined parameter. It is worthy of noting that, under the thermal equilibrium, the quasi-Fermi levels are constant, i.e., $E_{Fn1} = E_{Fn2}$ and $E_{Fp1} = E_{Fp2}$, then $J_{TE,n} = 0$ and $J_{TE,p} = 0$ as it should be.

Since the fundamental basic variables in Charon are electric potential, electron and hole densities, we need to rewrite Eq. (131) in terms of carrier densities and material properties. For Boltzmann

statistics, the electron and hole densities are given by $n = N_C \exp\left(\frac{E_{Fn}-E_C}{k_B T}\right)$ and $p = N_V \exp\left(\frac{E_V - E_{Fp}}{k_B T}\right)$, from which Eq. (131) can be rewritten as

$$
\begin{aligned}
J_{TE,n} &= A_n^* T^2 \left[-\frac{n_1}{N_{C1}} + \frac{n_2}{N_{C2}} \exp\left(-\frac{\Delta E_C}{k_B T}\right) \right], \\
J_{TE,p} &= A_p^* T^2 \left[\frac{p_1}{N_{V1}} - \frac{p_2}{N_{V2}} \exp\left(-\frac{\Delta E_V}{k_B T}\right) \right].
\end{aligned}
\tag{133}
$$

If one defines $v_n = A_n^* T^2/qN_C$ and $v_p = A_p^* T^2/qN_V$, Eq. (133) becomes

$$
\begin{aligned}
J_{TE,n} &= -q v_{n1} n_1 + q v_{n2} n_2 \exp\left(-\frac{\Delta E_C}{k_B T}\right), \\
J_{TE,p} &= q v_{p1} p_1 - q v_{p2} p_2 \exp\left(-\frac{\Delta E_V}{k_B T}\right)
\end{aligned}
\tag{134}
$$

From Eq. (134), one can interpret the electron thermionic emission current density for the band diagram given in Fig. 12-1(b) as follows: electrons do not see a ΔE_C energy barrier when thermionically emitting from side 1 to side 2, hence the electron current density from 1 to 2 is $(-q)v_{n1}n_1$ where the minus sign is due to the negative charge of an electron; whereas, electrons going from side 2 to side 1 do see a ΔE_C energy barrier, hence the electron current density is reduced by $\exp\left(-\frac{\Delta E_C}{k_B T}\right)$ and becomes $(-q)v_{n2}n_2\exp\left(-\frac{\Delta E_C}{k_B T}\right)$; the difference of the two current densities determines the net electron current across the HJ. A similar interpretation is applicable to the hole thermionic emission current density except that the charge of holes is positive.

For Fermi-Dirac statistics, the electron and hole densities are given by

$$
\begin{aligned}
n &= N_C F_{1/2}\left(\frac{E_{Fn} - E_C}{k_B T}\right), \\
p &= N_V F_{1/2}\left(\frac{E_V - E_{Fp}}{k_B T}\right),
\end{aligned}
\tag{135}
$$

where $F_{1/2}(\cdot)$ is the one-half Fermi-Dirac integral [39]. From the above equations, one can inversely solve for the arguments of the Fermi-Dirac integrals, i.e.,

$$
\begin{aligned}
\eta_n &= \frac{E_{Fn} - E_C}{k_B T} = F_{1/2}^{-1}\left(\frac{n}{N_C}\right), \\
\eta_p &= \frac{E_V - E_{Fp}}{k_B T} = F_{1/2}^{-1}\left(\frac{p}{N_V}\right),
\end{aligned}
\tag{136}
$$

where $F_{1/2}^{-1}(\cdot)$ is the inverse of the one-half Fermi-Dirac integral [40]. Substituting η_n and η_p into Eq. (131), we obtain

$$
\begin{aligned}
J_{TE,n} &= A_n^* T^2 \left[-\exp(\eta_{n1}) + \exp(\eta_{n2})\exp\left(-\frac{\Delta E_C}{k_B T}\right) \right], \\
J_{TE,p} &= A_p^* T^2 \left[\exp(\eta_{p1}) - \exp(\eta_{p2})\exp\left(-\frac{\Delta E_V}{k_B T}\right) \right].
\end{aligned}
\tag{137}
$$

118

The application of either Eq. (133) or Eq. (137) is determined by a single user-defined cutoff density. When the electron or hole density is larger than this cutoff density, Eq. (133) is used; otherwise, Eq. (137) is used.

Due to some limitation in the heterojunction implementation infrastructure, it is not straightforward to automatically compute the band offset at a HJ, therefore, the band offset is provided as a user-defined parameter in the input xml. Since the left (right) side always corresponds to side 1 (side 2), $\Delta E_C = E_{C1} - E_{C2}$, and $\Delta E_V = E_{V2} - E_{V1}$, as mentioned earlier, the band offsets can be either positive or negative. The equations described above are applicable to positive ΔE_C and ΔE_V. For other cases of ΔE_C and ΔE_V, the equations require some modification. Figure 12-2 summarizes the different cases of ΔE_C and ΔE_V, the corresponding band structures, and the corresponding net thermionic emission current densities for Boltzmann statistics (similar modifications hold for Fermi-Dirac statistics). All the four scenarios in Fig. 12-2 are taken into account in Charon.

(a) E_{C2} E_{C1} E_{V2} E_{V1}	$\Delta E_C > 0, \Delta E_V > 0$ $$J_{TE,n} = A_n^* T^2 \left[-\frac{n_1}{N_{C1}} + \frac{n_2}{N_{C2}} exp\left(\frac{-\Delta E_C}{k_B T}\right) \right]$$ $$J_{TE,p} = A_p^* T^2 \left[\frac{p_1}{N_{V1}} - \frac{p_2}{N_{V2}} exp\left(\frac{-\Delta E_V}{k_B T}\right) \right]$$
(b)	$\Delta E_C > 0, \Delta E_V < 0$ $$J_{TE,n} = A_n^* T^2 \left[-\frac{n_1}{N_{C1}} + \frac{n_2}{N_{C2}} exp\left(\frac{-\Delta E_C}{k_B T}\right) \right]$$ $$J_{TE,p} = A_p^* T^2 \left[\frac{p_1}{N_{V1}} exp\left(\frac{\Delta E_V}{k_B T}\right) - \frac{p_2}{N_{V2}} \right]$$
(c)	$\Delta E_C < 0, \Delta E_V < 0$ $$J_{TE,n} = A_n^* T^2 \left[-\frac{n_1}{N_{C1}} exp\left(\frac{\Delta E_C}{k_B T}\right) + \frac{n_2}{N_{C2}} \right]$$ $$J_{TE,p} = A_p^* T^2 \left[\frac{p_1}{N_{V1}} exp\left(\frac{\Delta E_V}{k_B T}\right) - \frac{p_2}{N_{V2}} \right]$$
(d)	$\Delta E_C < 0, \Delta E_V > 0$ $$J_{TE,n} = A_n^* T^2 \left[-\frac{n_1}{N_{C1}} exp\left(\frac{\Delta E_C}{k_B T}\right) + \frac{n_2}{N_{C2}} \right]$$ $$J_{TE,p} = A_p^* T^2 \left[\frac{p_1}{N_{V1}} - \frac{p_2}{N_{V2}} exp\left(\frac{-\Delta E_V}{k_B T}\right) \right]$$

Figure 12-2 Different cases of ΔE_C and ΔE_V, the corresponding band diagrams, and the corresponding net thermionic emission current densities for Boltzmann statistics.

12.2. Local Tunneling

If the potential barrier at an abrupt HJ is sufficiently narrow, carrier tunneling will contribute significantly to the net carrier flux across the junction. This occurs when the doping in one side is high causing a large electric field in the depletion region adjacent to the junction and therefore a thin barrier. For the conduction band profile of the NPN HBT in Fig. 12-1, electrons can tunnel from the emitter (side 1) to the base (side 2) due to the high electric field in the emitter. Tunneling across a HJ can be categorized into local and non-local tunneling. The local tunneling (LT) model is basically a simplified version of the more accurate non-local tunneling model, with the local model being much easier to implement than the non-local one, especially in a distributed parallel running environment. Charon currently supports only the LT model. The LT model is derived in

detail from the original paper [36] in Appendix C, and its final form is given below.

Taking the conduction band profile in Fig. 12-2(a) as an example, the net electron current density across the HJ including both thermionic emission and local tunneling is given by

$$J_{HJ,n} = J_{TE,n}(1 + \delta_n), \tag{138}$$

where the tunneling factor δ_n takes the form of [see Eq. (192)]

$$\delta_n = \frac{1}{k_B T} \int_0^{\Delta E_C} \exp\left[-\frac{8\pi}{3hq\xi}\sqrt{2m_{nt}^*}(\Delta E_C - E)^{3/2}\right] \exp\left(\frac{\Delta E_C - E}{k_B T}\right) dE.$$

In this equation, h is the Planck constant, q is the elemental charge, ξ is the larger electric field of the two sides [i.e., the electric field in side 1 for Fig. 12-2(a)], and m_{nt}^* is the electron tunneling effective mass. To perform the integration over energy efficiently, one can rewrite δ_n in the following form:

$$\delta_n = \int_0^{\frac{\Delta E_C}{k_B T}} \exp\left[u - \left(\frac{u}{u_0}\right)^{\frac{3}{2}}\right] du, \tag{139}$$

where u and u_0 are defined as

$$u = \frac{\Delta E_C - E}{k_B T},$$

$$u_0 = \frac{1}{k_B T}\left(\frac{3hq\xi}{8\pi\sqrt{2m_{nt}^*}}\right)^{\frac{2}{3}}. \tag{140}$$

The above tunneling model is also applicable to the electron tunneling over the conduction band profile in Fig. 12-2 (c), except that the absolute value of ΔE_C and the larger electric field in side 2 should be used in computing δ_n.

For hole tunneling through the valence band profiles in Fig. 12-3, the same equations [Eqs. (138) and (139)] are applicable, except that ΔE_C and m_{nt}^* are replaced by ΔE_V and m_{pt}^*, respectively. Note that the valence band profiles in Fig. 12-2(a) and (c) do not allow for hole tunneling, while hole tunneling over the valence band profiles in Fig. 12-2(b) and (d) is negligible due to the fact that the tunneling source side has a smaller electric field which is used in computing the local tunneling factor δ_p.

12.3.   Model Implementation

Implementation of the TE and LT models in Charon consists of two main parts: (i) compute the net current density across a HJ according to the models described above, (ii) add the HJ current density to the residual of the carrier continuity equations as an interface flux condition. The carrier continuity equations are discretized using both the finite element method with the streamline upwinding Petrov-Galerkin stabilization (FEM-SUPG) [10], and the control volume finite element method with the Scharfetter-Gummel stabilization (CVFEM-SG) [11]. Hence the
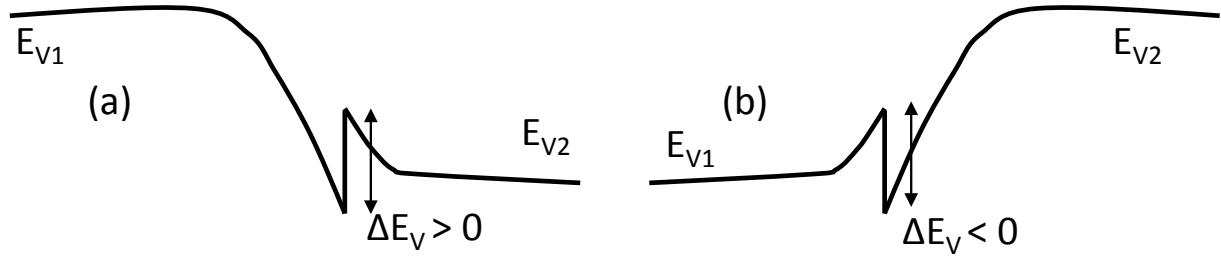
Figure 12-3 Examples of valence band diagrams that allow for hole tunneling.

HJ models are implemented for the two discretization schemes. In the following, the implementation of the HJ models is highlighted for the FEM-SUPG discretization, and a similar implementation is also done for the CVFEM-SG method.

The dimensionless carrier continuity equations take the form of

$$
\begin{aligned}
\frac{\partial n}{\partial t} - \nabla \cdot \mathbf{J_n} + R &= 0, \\
\frac{\partial p}{\partial t} + \nabla \cdot \mathbf{J_p} + R &= 0.
\end{aligned}
\tag{141}
$$

For implementation purpose, the equations need to be rewritten in the FEM weak forms. Considering the two element blocks (e.g., eb1 and eb2) that share a HJ as illustrated in Fig. 12-4, the carrier densities are discontinuous at the HJ, hence the basic variables are (n_1, p_1, ϕ) for eb1, and (n_2, p_2, ϕ) for eb2, where ϕ is the to-be-solved electric potential which is continuous across the HJ. The Galerkin finite element weak forms of the continuity equations are
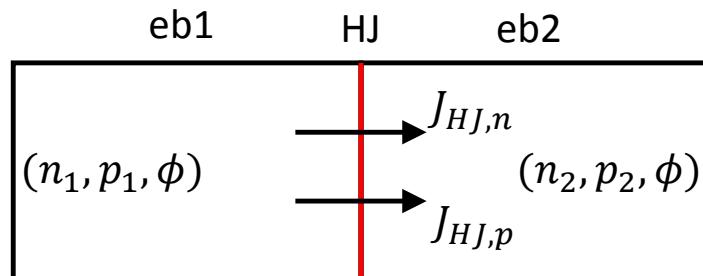
Figure 12-4 Schematics of two element blocks with a heterojunction, showing the basic variables in each element block and the currents across the junction.

$$\int_\Omega \left(\frac{\partial n_1}{\partial t} + R \right) w \, d\Omega + \int_\Omega \mathbf{J_{n1}} \cdot \nabla w \, d\Omega - \int_{HJ} J_{HJ,n} w \, d\Omega = 0,$$

$$\int_\Omega \left(\frac{\partial n_2}{\partial t} + R \right) w \, d\Omega + \int_\Omega \mathbf{J_{n2}} \cdot \nabla w \, d\Omega + \int_{HJ} J_{HJ,n} w \, d\Omega = 0,$$

$$\int_\Omega \left(\frac{\partial p_1}{\partial t} + R \right) w \, d\Omega - \int_\Omega \mathbf{J_{p1}} \cdot \nabla w \, d\Omega + \int_{HJ} J_{HJ,p} w \, d\Omega = 0,$$

$$\int_\Omega \left(\frac{\partial p_2}{\partial t} + R \right) w \, d\Omega - \int_\Omega \mathbf{J_{p2}} \cdot \nabla w \, d\Omega - \int_{HJ} J_{HJ,p} w \, d\Omega = 0, \tag{142}$$

where w is the FE nodal basis function, $J_{HJ,n}$ and $J_{HJ,p}$ are the net heterojunction current density for electrons and holes respectively, with $J_{HJ,n}$ given in Eq. (138). The same $J_{HJ,n}$ is applied to the n_1 and n_2 equations except for the opposite sign, and the same holds true for $J_{HJ,p}$

12.4.   Model Usage

To use the TE and LT models for simulating a heterodevice, besides the common settings used for simulating a homojunction device such as a silicon pn diode provided in an input file, we need two additional settings: (i) specify discontinuous suffix numbers in physics blocks; (ii) specify a heterojunction block. Taking the InGaP/GaAs/GaAs NPN HBT device in Fig. 12-1 as an example, the emitter-base junction can have electron and hole thermionic emission currents and electron tunneling current, but does not support hole tunneling.

First, the discontinuous suffix numbers are specified for the InGaP and GaAs physics blocks, as shown in the following snippet. The InGaP physics block corresponds to the emitter InGaP region which is defined as side 1, while the GaAs physics block corresponds to the base and collector regions which are defined as side 2.

```
start Physics Block InGaP
     discontinuity for hbt with suffix 1
     geometry block is emitter-ingap
     material model is InGaP-Parameter
     ...
end

start Physics Block GaAs
     discontinuity for hbt with suffix 2
     geometry block is base-gaas
     geometry block is collector-gaas
     material model is GaAs-Parameter
     ...
end Physics Block GaAs
```

Second, one needs to specify the heterojunction block as follows

```
start heterojunction on {sidesetName}
   junction is between blocks {block1var} and {block2var}
   start electron
      discretization method is fem or cvfem
      local tunneling is on or off
```

```
        (one parameter per line in the form):
        {parameter name} = {parameter value}
        (see table 12-1 for available parameters)
    end electron
    start hole
        discretization method is fem or cvfem
        local tunneling is on or off
        (same parameters as electron above)
    end hole
end
```

The `sidesetName` keyword specifies the sideset at which the heterojunction BC is applied. The sideset needs to be defined in Cubit as a one-sided sideset. `block1var` and `block2var` are the two element blocks that touch the same sideset defined by `sidesetName`. `discretization method` specifies the discretization method for computing heterojunction current density, which can be either `fem` (finite element discretization) or `cvfem` (Scharfetter-Gummel discretization). `local tunneling` specifies if one wants to include tunneling current in heterojunction BC calculation. When it is `on`, Eq. (138) is used; otherwise, Eq. (131) is used in computing heterojunction current density.

Table 12-1 Syntax and parameters for the heterojunction boundary condition. m_0 **is the free electron effective mass.**

Input file	Corresponding variables	Description	units
fermi dirac density	None	carrier density above which the Fermi-Dirac approach is used in computing current density	cm^{-3}
effective mass	m_n^* or m_p^* in Eq. (132)	carrier effective mass for computing the Richardson constants	m_0
band offset	ΔE_C or ΔE_V in Eq. (131)	conduction or valenche band offset	eV
loca tunneling mass	m_{nt}^* or m_{pt}^* in Eq. (140)	carrier effective mass for local tunneling calculation	m_0

For the InGaP/GaAs/GaAs NPN HBT example in Fig. 12-1, the heterojunction block looks like

```
start heterojunction on ebjunction
  junction is between blocks emitter-ingap and base-gaas
  start hole
    effective mass = 0.58
    band offset = 0.444
    fermi dirac density = 1e11
    discretization method is fem
  end hole
  start electron
    effective mass = 0.088
    band offset = 0.064
    fermi dirac density = 1e11
    discretization method is fem
    local tunneling is on
```

```
    tunneling effective mass = 0.088
  end electron
end heterojunction
```

Complete examples on the heterojunction BC can be found at `tcad-charon/test`
`/nightlyTests/b2ttunnel/`.

12.5. Fixed Charge at Interface

In GaN/AlGaN/GaN-based HEMTs an important cause of electrons in the 2DEG channel is the existence of a fixed charge at the boundary between the wide bandgap semionductors bordering the channel. Charon provides the capability to place fixed positive or negative charge at semiconductor/semiconductor interfaces (heterointerfaces).

When adding charge at heterointerfaces, the electric field on both sides of the interface is affected in accordance with the Gauss's Law. Given the charge density σ in units of cm^{-2} at the interface between materials 1 and 2, the Gauss's Law determines a relation between the normal components of electric field displacements or gradients of potential on each side of the interface:

$$
\begin{aligned}
-\mathbf{D_1} \cdot \mathbf{\eta_1} - \mathbf{D_2} \cdot \mathbf{\eta_2} &= q\sigma, \\
\varepsilon_0 \varepsilon_{r2} \nabla\phi_2 \cdot \mathbf{\eta_2} + \varepsilon_0 \varepsilon_{r1} \nabla\phi_1 \cdot \mathbf{\eta_1} &= q\sigma
\end{aligned}
\tag{143}
$$

where we used

$$
\begin{aligned}
\mathbf{D_1} &= -\varepsilon_0 \varepsilon_{r1} \nabla\phi_1, \\
\mathbf{D_2} &= -\varepsilon_0 \varepsilon_{r2} \nabla\phi_2
\end{aligned}
\tag{144}
$$

Here, q is the elemental charge in units of Coulombs (C), ε_0 is the vacuum permittivity with $\varepsilon_0 = 8.8542 \times 10^{-14} \, CV^{-1} cm^{-1}$, ε_{r1} and ε_{r2} are the relative permittivity of materials 1 and 2, $\nabla\phi_1$ and $\nabla\phi_2$ are the potential gradients in units of $V cm^{-1}$ in materials 1 and 2 respectively, η_1 is the normal unit vector of side 1 pointing toward side 2 and η_2 is the normal unit vector of side 2 pointing toward side 1.

The boundary conditions on each side of the interface are derived by expressing the normal component of the electric displacement field of a function of the electric displacement field on the opposite side and the charge on the interface using Eq. (143):

$$
\begin{aligned}
\mathbf{D_1} \cdot \mathbf{\eta_1} &= \varepsilon_0 \varepsilon_{r2} \nabla\phi_2 \cdot \mathbf{\eta_2} - q\sigma, \\
\mathbf{D_2} \cdot \mathbf{\eta_2} &= \varepsilon_0 \varepsilon_{r1} \nabla\phi_1 \cdot \mathbf{\eta_1} - q\sigma
\end{aligned}
\tag{145}
$$

or scaled, as used internally by Charon

$$
\begin{aligned}
\mathbf{D_1^s} \cdot \mathbf{\eta_1} &= \varepsilon_{r2} \nabla\phi_2^s \cdot \mathbf{\eta_2} - \frac{q\sigma X_0}{\varepsilon_0 V_0}, \\
\mathbf{D_2^s} \cdot \mathbf{\eta_2} &= \varepsilon_{r1} \nabla\phi_1^s \cdot \mathbf{\eta_1} - \frac{q\sigma X_0}{\varepsilon_0 V_0}
\end{aligned}
\tag{146}
$$

Here the superscript s denotes that the electric field displacements and potential gradients are scaled and X_0 and V_0 are the length and voltage scaling factors.

Then the contributions to the non-linear Poisson equation residual for side 1 and side 2 are given by

$$-\lambda^2(-\frac{q\sigma X_0}{\varepsilon_0 V_0} + \varepsilon_{r2}\nabla\phi_2^s \cdot \eta_2) = \frac{\sigma}{X_0 C_0} - \lambda^2\varepsilon_{r2}\nabla\phi_2^s \cdot \eta_2,$$

$$-\lambda^2(-\frac{q\sigma X_0}{\varepsilon_0 V_0} + \varepsilon_{r1}\nabla\phi_1^s \cdot \eta_1) = \frac{\sigma}{X_0 C_0} - \lambda^2\varepsilon_{r1}\nabla\phi_1^s \cdot \eta_1 \tag{147}$$

where $\lambda^2 = \frac{V_0\varepsilon_0}{qX_0^2 C_0}$ and C_0 is the concentration scaling factor.

Under the assumption that the potential is continuous at the hetero interface (single point), the total residual contribution due to the interface charge to the non-linear Poisson is given by sum of the individual contributions from each side in Eq. (147):

$$2\frac{\sigma X_0}{\varepsilon_0 C_0} - \frac{\lambda^2}{\frac{\varepsilon_0 V_0}{X_0}}(\varepsilon_{r2}\nabla\phi_2 \cdot \eta_2 + \varepsilon_{r1}\nabla\phi_1 \cdot \eta_1) \tag{148}$$

Using the second expresssion in Eq. (145) in its scaled form, the total residual contribution due the interface charge reduces to $\frac{\sigma X_0}{\varepsilon_0 C_0}$.

Two steps are required to add charge at heterointerfaces. First, we need to define the charge interface as a single-sided sideset in Cubit. Second, details about interface BC need to be specified in the Charon's input file, inside thr potential heterojunction block, as follows

```
start potential heterojunction on {sidesetName}
   junction is between blocks {block1var} and {block2var}
   start potential
      discretization method is {discMethod}
      fixed charge = {surfCharge}
   end potential
end
```

Here, the `sidesetName` keyword specifies the sideset at which the potential heterojunction BC is applied. The sideset needs to be defined in Cubit as a one-sided sideset. `block1var` and `block2var` are the two element blocks that touch the same sideset defined by `sidesetName`. `discMethod` specifies the discretization method for computing heterojunction charge induced flux, which can be either `fem` (finite element discretization) or `cvfem` (Scharfetter-Gummel discretization).

13.   HARMONIC BALANCE

Frequency domain simulation of the coupled Drift-Diffusion and Poisson equations is possible in Charon via the harmonic balance (HB) method. All physical discretization schemes are supported, and either a small-signal or large-signal analysis can be chosen for the frequency-domain analysis method. Some boundary conditions, such as the ohmic contact and contact on insulator, are presently supported. In this section, we will refer to the standard Charon formulation as the time-domain (TD) formulation to distinguish it from the HB formulation.

13.1.   Formulation

For brevity, we outline Charon's harmonic balance method for the electron drift-diffusion equation in a Galerkin finite-element spatial discretization; the treatment is similar for the hole, Poisson, and lattice temperature equations in any discretization scheme. In the time domain, the electron drift-diffusion equation can be expressed as

$$\frac{\partial n}{\partial t} + \mathcal{F}_n(n, p, \phi) = 0, \tag{149}$$

where $n(t)$, $p(t)$, and $\phi(t)$ is the electron density, hole density, and electric potential, respectively at time t. The quantity $\mathcal{F}_n$ consists of terms which are not explicitly time-dependent. Explicit time-dependent terms do not appear here, but will arise numerically as boundary condition forcing terms through periodic time-varying voltage boundary conditions. Using a spatial basis function $\Lambda(\vec{x})$ on a spatial discretization element V, a component of the residual is

$$\int_V \frac{\partial n}{\partial t} \Lambda(\vec{x}) \, d\vec{x} + \mathcal{R}_n^\Lambda(n(t), p(t), \phi(t)) = 0. \tag{150}$$

Note that $\mathcal{R}_n^\Lambda(t)$ denotes the spatial residual of the steady-state TD equation. The term $\frac{\partial n}{\partial t}$ is approximated using a time discretization method (e.g., backward-Euler) for a TD transient simulation, or else omitted for a steady-state simulation. In the harmonic balance method, this term is transformed into the frequency domain.

The HB method (in either the small-signal and large-signal modes) has a solution ansatz whose form depends on the applied contact voltage frequencies ω_1 Hz, ω_2 Hz, $\ldots, \omega_\ell$ Hz, called the fundamental frequencies of a simulation. Without loss of generality, we assume that $\omega_1 < \omega_2 < \cdots < \omega_\ell$, and write $\vec{\omega} = (\omega_1, \ldots, \omega_\ell)$ for conciseness. A choice is made for the degree of intermodulation frequencies to be captured by the HB method, and the collection of those intermodulation frequencies is called a *truncation scheme*. Truncation schemes $\mathcal{T}$ are recorded as integer lattice points $\vec{k} = (k_1, k_2, \ldots, k_\ell) \in \mathbb{Z}^\ell$ which correspond to a linear combination $\vec{\omega} \cdot \vec{k}$ of the fundamental frequencies. Thus, the HB solution ansatz takes the form

$$n(\vec{x}, t) = N_0(\vec{x}) + \sum_{\vec{k} \in \mathcal{T}} \left[N_{\vec{k}}^c(\vec{x}) \cos\left(2\pi\vec{\omega} \cdot \vec{k}t\right) + N_{\vec{k}}^s(\vec{x}) \sin\left(2\pi\vec{\omega} \cdot \vec{k}t\right) \right] \tag{151}$$

Note that the physical solution contains more frequencies than those fundamental frequencies applied, and the ansatz captures this but necessarily truncates the frequencies observed [41]. Common truncation schemes interpolate between the Box and Diamond truncation schemes [42].

To arrive at the HB equations, we first multiply (150) by a Fourier basis function, i.e.,

$$\cos\left(2\pi\vec{\omega}\cdot\vec{k}t\right) \text{ or } \sin\left(2\pi\vec{\omega}\cdot\vec{k}t\right) \text{ for } \vec{k} \in \mathcal{T},$$

and integrate over a period. This results in $2|\mathcal{T}|+1$ residual equations, disregarding $\sin(2\pi 0 t)$. In the following, we explicitly describe the procedure involving $\cos(2\pi\omega_i t)$ because all other cases are similar. We next integrate the time-derivative summand analytically, and the time-independent summand numerically using a trapezoidal rule (which converges exponentially for periodic functions [43]). Thus, we obtain the $\cos(2\pi\omega_i t)$ HB residual equation:

$$0 = \pi\omega_i \int_V N^s_{\omega_i}(\vec{x})\Lambda(\vec{x})\, d\vec{x} + \sum_{m=0}^{L} w^m_{\omega_i} \mathcal{R}^\Lambda_{n}(n(t_m), p(t_m), \phi(t_m)) \tag{152}$$

involving time collocation points and quadrature weights

$$t_m = \frac{m}{L} \quad \text{and} \quad w^m_{\omega_i} \equiv \frac{2 - \delta_{0m} - \delta_{mL}}{2}\cos(2\pi\omega_i t_m)$$

where $\delta_{ab} = 1$ when $a = b$ and $\delta_{ab} = 0$ otherwise, and $L = 2\omega_\ell$ by the Nyquist Sampling Theorem.

Note that the arguments $n(t_m)$, $p(t_m)$, and $\phi(t_m)$ of $\mathcal{R}^\Lambda_{n}$ are evaluated via the ansatz expression (151). For this evaluation, the small-signal and large-signal formulations differ slightly: for the small-signal analysis, the summation is restricted to only $\vec{k}$ which yields $\vec{\omega}\cdot\vec{k} = \omega_i$; for the large-signal analysis, the summation ranges over all $\vec{k} \in \mathcal{T}$. This has the following effect: the small-signal response at the ω_i frequency is only influenced by the contact voltage amplitudes at 0 and ω_i hertz, whereas the large-signal response at any frequency is influenced by the contact voltage amplitudes of all frequencies and their intermodulations.

Since the HB residual equation (152) is assembled through evaluation of the steady-state TD model $\mathcal{R}^\Lambda_{n}$ at multiple collocation points (with the corresponding value of the HB solution ansatz (151) at that time), any discretization method and physical models that are implemented for the TD analysis can be carried over into the HB residual. Practically, the HB residual involves orders of magnitude more computation than a TD residual, but parallelization accelerates the summation operations which appear in the ansatz (151) and the residual (152).

The Nyquist Sampling Theorem recommends a minimum of $2\omega_\ell + 1$ time collocation points, which is too great for large frequencies. This makes the transform (152) prohibitively expensive. To address this, frequency remaping is typically leveraged for harmonic balance methods. We have developed an efficient *minimal iso-frequency remapping* as an alternative to various frequency mapping methods in the literature [44]. For example, if the fundamental frequencies for an order 3 box truncation scheme hb analysis are $1.0 MHz$ and $1.3 MHz$, the algorithm shows that the remapped frequencies 3Hz and 4Hz yield an equivalent problem.

```
Applying the isofrequency remapping algorithm to

omega = [1000000.0, 1130000.0]
with a truncation order = 3
There are 25 truncation coefficients.

  Determining the cone N and annihilators A ... 0.005552053451538086s
   Creating input deck for Normaliz ... 0.02010822296142578s
    Running Normaliz to find the Hilbert basis ... 0.013696670532226562s
     Using Hilbert basis to find minimal element ... 0.008527994155883789s

Complete! The minimal eta is: [4 5]
```

Important note: if remapped frequencies are used, the boundary conditions should be specified
with respect to these remapped frequencies.

13.2.   Usage

The specificationun of a harmonic balance analysis is similar to that of a time-domain simulation,
except for input deck specifications in the

- physics block frequency domain options (as a sub-block),

- boundary conditions (possibly applying a small-signal DC sweep), and

- initial conditions (which can come from a previously steady-state simulation).

The output of the HB Charon simulation will be in the frequency domain. Thus, the Fourier
coefficients of the current and degrees of freedom are returned. For example, where a TD
simulation returns `ELECTRIC_POTENTIAL`, an HB simulation returns
`ELECTRIC_POTENTIAL_CosH0.000000_`, `ELECTRIC_POTENTIAL_SinH1.000000_`,
`ELECTRIC_POTENTIAL_CosH1.000000_`, etc. In particular, the
`ELECTRIC_POTENTIAL_CosH0.000000_` field is the DC component of the solution field; in a
small-signal analysis, `ELECTRIC_POTENTIAL_SinH1.000000_` and
`ELECTRIC_POTENTIAL_CosH1.000000_` are the small-signal components of the solution field. In
the following, we will describe the three necessary input deck blocks in order to perform a
harmonic balance analysis in Charon.

13.2.1.   Physics Block

A harmonic balance analysis needs to be specified in the physics block, choosing either a
small-signal or large-signal analysis. A physics sub-block specifying frequency domain
parameters is also required. Within a physics block corresponding to one named
physicsBlockName, this has form:

129

```
start Physics Block physicsBlockName
  apply harmonic balance
  start harmonic balance parameters
    ..
  end
end
```

In Table 13-1, we provide a description of the options and parameters which can be set for a harmonic balance analysis.

Table 13-1 Syntax for HB analysis in physics block.

apply harmonic balance [for small signal analysis [for large signal analysis]]		
start harmonic balance parameters		
truncation order = {*truncationOrder*}		
truncation scheme is {*truncationScheme*}		
fundamental harmonics = {*fundamentalHarmonics*}		
remapped harmonics = {*remappedHarmonics*}		
end		
Option	Description	Required
for small signal analysis	Specify a small-signal frequency domain analysis.	No
for large signal analysis	Specify a large-signal frequency domain analysis. This is the default mode of harmonic balance analysis, if neither option is specified.	No
remapped harmonics =	Specify a sequence of remapped fundamental harmonics. If remapped frequencies are used, then the boundary conditions should be specified with respect to these values.	*No*
Variables	Description	Default
truncationOrder	Specify the order of truncation scheme.	*None*
truncationScheme	Specify the type of truncation scheme to use, chosen among: box, diamond.	*None*
fundamentalHarmonics	Space-separated sequence of fundamental harmonics.	*None*
remappedHarmonics	Space-separated sequence of remapped fundamental harmonics.	*None*

13.2.2.   Boundary Conditions

In place of the standard boundary condition option which takes the form

```
BC is bcType for contactName on materialName
```

we have instead, for a harmonic balance analysis, a block which takes the form

```
start hb boundary conditions for contactName on materialName
  type is bcType
  ..
end
```

Within this block, the kind of boundary condition applied can be specified. The parameters omitted in ellipses are described in Table 13-2, and pertain to the parameterization of the periodic voltage boundary condition for the analysis. In the case of a small-signal analysis, a DC voltage boundary condition around which the small-signal perturbation is applied can be swept. Where the TD boundary voltage sweep is specified with

```
BC is bcType for contactName on materialName swept from potential1 to potential2
```

we have instead, for a small-signal harmonic balance analysis with a DC bias sweep, a block which takes the form

```
start hb boundary conditions for contactName on materialName
  type is bcType
  small signal sweep from {potential1}to potential2 with amplitude ssAmplitude /with phase shift
ssPhaseShift with frequency ssFrequencyend
```

We emphasize: the frequency specified for the small-signal analysis needs be entered with respect to the remapped fundamental frequencies. The small-signal HB boundary condition parameters are described in Table 13-3.

Table 13-2 Syntax for HB Boundary Conditions. Note that the list of *frequencyValues*, *amplitudeValues*, and *phaseshiftValues* must have the same number of parameters.

start Harmonic Balance BC for *contactName* on *materialName*
    type is *bcType*
    frequencies = "*frequencyValues*"
    amplitudes = "*amplitudeValues*"
    phase shifts = "*phaseshiftValues*"
end		

Option	Description	Required
	None	
Variables	Description	Default
contactName	Specify a contact name on which to apply this boundary condition.	*None*
materialName	Specify the material name on which the contact is provided.	*None*
bcType	Choose among: `ohmic`, ... (additional parameters may be required)	*None*
frequencyValues	Space-separated values of (remapped) frequencies, in quotes.	*None*
amplitudeValues	Space-separated values of amplitudes, in quotes.	*None*
phaseshiftValues	Space-separated values of phase shifts, in quotes.	*None*

To specify an ohmic contact boundary condition, the following syntax is used:

```
start Harmonic Balance BC for contactName on materialName
  type is ohmic for anode on silicon
end Harmonic Balance BC for contactName on materialName
```

To specify a gate contact or contact on insulator, the following syntax is used:

Table 13-3 Syntax for HB Boundary Conditions for performing a DC sweep for a small-signal analysis.

start Harmonic Balance BC for *contactName* on *materialName*		
type is *bcType*		
small signal sweep from *potential1* to *potential2* with amplitude *ssAmplitude* /		
with frequency *ssFrequency* with phase shift *ssPhaseShift*		
end		
Option	Description	Required
	None	
Variables	Description	Default
contactName	Specify a contact name on which to apply this boundary condition.	*None*
materialName	Specify the material name on which the contact is provided.	*None*
bcType	Choose among: `ohmic`, ... (additional parameters may be required)	*None*
ssFrequency	single (remapped) frequecy value in Hz for the small-signal perturbation	*None*
ssAmplitude	single voltage value for small-signal perturbation.	*None*
ssPhaseShift	single percentage value for small-signal phase shift	*None*

```
start Harmonic Balance BC for contactName on materialName
  type is ohmic for contact on inulator for silicon with work function {workFunction}
end Harmonic Balance BC for contactName on materialName
```

Note that boundary conditions can be mixed, so that, for example, a MOS capacitor can be analyzed using harmonic balance. Please see 13.3.3 for this use case.

13.2.3.   Initial Conditions

In place of the TD boundary condition option which takes the form

```
initial conditions for degreeOfFreedom in materialName is icType
```

for HB the specification takes the form described in Table 13-4.

13.3.   Example usage

13.3.1.   Modifying a steady-state time-domain input deck for frequency-domain analysis

After a drift-diffusion equilibrium simulation, the solution degrees of freedom can be used to initialize the steady-state solution components of a harmonic balance small-signal analysis. In this example, we describe the procedure of modifying the input deck for a (time-domain) equilibrium simulation of a p-n diode in order to apply a small-signal analysis on the same p-n diode to obtain its steady-state frequency-domain behavior. Although the small-signal analysis can be performed without first simulating the time-domain equilibrium simulation, solution initialization may be too far for convgence.

Table 13-4 Syntax for HB Initial Conditions.

Harmonic Balance Initial Conditions for {*degreeOfFreedom*} in {*materialName*} is [Exodus File [constant = constValue [remapped named remapDOF [DC remapped [uninitialized for higher harmonics]]]]]		
Option	Description	Required
	None	
Variables	Description	Default
degreeOfFreedom	Choose one of: `ELECTRIC_POTENTIAL`, `ELECTRON_DENSITY`, `HOLE_DENSITY`, `LATTICE_TEMPERATURE`	*None*
materialName	Specify the material in which this initial condition is to be specified.	*None*
exodus file	This will read in the degree of freedom from the input exodus file.	*None*
constant	This specifies that a constant value equal to *constValue* will be set for this degree of freedom.	*None*
DC remapped	This will use `ELECTRIC_POTENTIAL` from the input exodus file to initialize the `ELECTRIC_POTENTIAL_CosH0.000000_` field, etc.	*None*
uninitialized for higher harmonics	This leaves the degree of freedom uninitialized (initialized to 0.0) for any harmonic not specified. Typically, the DC remap option described above will be used in conjunction with this.	*None*

In the following, we refer to the two nightly test input files:

```
tcad-charon/test/nightlyTests/capability/pndiode.sg.equ.inp
tcad-charon/test/nightlyTests/freqdom/pndiode/pndiode.hb-sgcvfem-dd.inp
```

for these p-n diode simulations. Note that both input decks use the common discretization type of drift-diffusion cvfem. As described in the preceding usage section, modifications need to be made to the physics block, initial conditions, and boundary conditions to adapt the former input deck into the latter input deck to execute this change of analysis.

The equilibrium (time-domain) simulation of the p-n diode as specified in

```
tcad-charon/test/nightlyTests/capability/pndiode.sg.equ.inp
```

applies the boundary conditions

$$V_{\text{anode}} = 0.0V \text{ and } V_{\text{cathode}} = 0.0V$$

whereas the small-signal (frequency-domain) simulation of the same pn-diode as specified in

```
tcad-charon/test/nightlyTests/freqdom/pndiode.hb-sgcvfem-dd.ac1-linear-SS.inp
```

applies the boundary conditions

$$V_{\text{anode}} = (4.0 + 2.0 \cdot \sin(2\pi \cdot 1Hz \cdot t))V \text{ and } V_{\text{cathode}} = 0.0V$$

In the following excerpt from

`tcad-charon/test/nightlyTests/capability/pndiode.sg.equ.inp`

we highlight the commands in violet which were modified to support a harmonic balance analysis.

```
start Physics Block Semiconductor
     geometry block is silicon
     standard discretization type is drift diffusion cvfem
     material model is siliconParameter
     ...
end Physics Block Semiconductor
...
BC is ohmic for anode on silicon fixed at 0
BC is ohmic for cathode on silicon fixed at 0
...
initial conditions for ELECTRIC_POTENTIAL in silicon is Exodus File
Initial Conditions for ELECTRON_DENSITY in silicon is Equilibrium Density
Initial Conditions for HOLE_DENSITY in silicon is Equilibrium Density
```

In the following excerpt from

`tcad-charon/test/nightlyTests/freqdom/pndiode/pndiode.hb-sgcvfem-dd.inp`

we highlight the commands in teal which were added to, or modified from, the excerpt above.

```
start Physics Block Semiconductor
     geometry block is silicon
     standard discretization type is drift diffusion cvfem
     apply harmonic balance for large signal analysis
     start harmonic balance parameters
      truncation order = 1
      truncation scheme is box
      fundamental harmonics = 1
     end harmonic balance parameters
     material model is siliconParameter
     ...
end Physics Block Semiconductor
...
start Harmonic Balance BC for anode on silicon
 type is ohmic
 frequencies "0 1"
 amplitudes "4 2"
 phase shifts "0 0"
end Harmonic Balance BC for anode on silicon
start Harmonic Balance BC for cathode on silicon
 type is ohmic
 frequencies "0"
 amplitudes "0"
 phase shifts "0"
end Harmonic Balance BC for cathode on silicon
...
Harmonic Balance initial conditions for ELECTRIC_POTENTIAL in silicon is uninitialized for higher
harmonics
```

```
Harmonic Balance Initial Conditions for ELECTRON_DENSITY in silicon is uninitialized for higher
harmonics
Harmonic Balance Initial Conditions for HOLE_DENSITY in silicon is uninitialized for higher harmonics
Harmonic Balance Initial Conditions for LATTICE_TEMPERATURE in silicon is uninitializedfor higher
harmonics
```

13.3.2.   PN diode small-signal capacitance-voltage profiling

Using the harmonic balance formulation, a small-signal perturbation to a contact DC voltage
sweep can be performed to obtain the capacitance-voltage profile of a p-n diode. Sinuoidal
steady-state analysis, equivalent to a specific configuration of harmonic balance analysis, has
been demonstrated to accurately capture the small-signal capacitance of a p-n diode [45],
comparing favorably in numerical experiment to measurement for both a long-base and short-base
diode. We will refer to these input decks and their corresponding test in this example:

```
tcad-charon/test/nightlyTests/freqdom/loca_pndiode_cv/pndiode.hb-dd.loca.longbase.inp
tcad-charon/test/nightlyTests/freqdom/loca_pndiode_cv/pndiode.hb-dd.loca.shortbase.inp
```

These tests produce the Charon HB values plotted as a solid curve in Figure 13-1, matching the
values obtained in [45] using sinusoidal steady-state analysis.
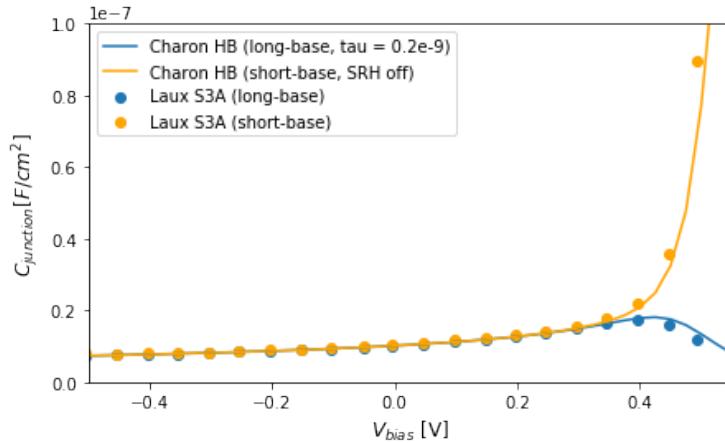
**Figure 13-1 Capacitance-voltage profile comparison between example long-base and
short-base PN diodes. Obtained using a small-signal perturbation to a contact DC
voltage sweep, applied with a harmonic balance boundary condition.**

In order to perform the small-signal perturbation of a contact DC voltage sweep, these input
decks specify the following boundary conditions. The anode contact voltage is held at 0V and a
$1e-3$V amplitude sinusoidal small-signal perturbation is applied to the cathode contact DC
voltage, which itself is swept from -0.55V to 0.55V.

```
start Harmonic Balance BC for anode on silicon
  type is ohmic
  small signal sweep from -0.55 to 0.55 with amplitude 1e-3
end Harmonic Balance BC for anode on silicon
```

```
start Harmonic Balance BC for cathode on silicon
  type is ohmic
  frequencies "0"
  amplitudes "0"
  phase shifts "0"
end Harmonic Balance BC for cathode on silicon
```

13.3.3.   *MOS capacitor small-signal capacitance-voltage profiling*

As in the previous p-n diode capacitance-voltage profiling example, a small-signal DC sweep can
be performed using a hamonic balance analysis to obtain the frequency-dependent
capacitance-voltage characteristic of a MOS capacitor. However, unlike the p-n diode where the
coupled Poisson-drift-diffusion equations are simulated throughout the device, the MOS capacitor
requires a Laplace equation set simulated in the oxide and a Poisson-Drift-Diffusion equation set
simulated in the substrate. We will refer to the input decks:

tcad-charon/test/nightlyTests/freqdom/loca_moscap_cv/moscap.hb.locaOxide.1e0Hz.inp
tcad-charon/test/nightlyTests/freqdom/loca_moscap_cv/moscap.hb.locaSubstrate.1e6Hz.inp

In the first input deck, the substrate contact is fixed at 0V while the gate contact DC voltage is
swept with a small-signal perturbations. In the second input deck, the gate contact is fixed at 0V
hile the substrate contact DC voltage is swept with a small-signal perturbation. In Figure 13-2, the
small-signal capacitance is plotted versus the DC V_{gs} voltage bias across the MOS capacior from
the gate contact to the substrate contact. Note that the frequecy dependence of the capacitance is
captured.
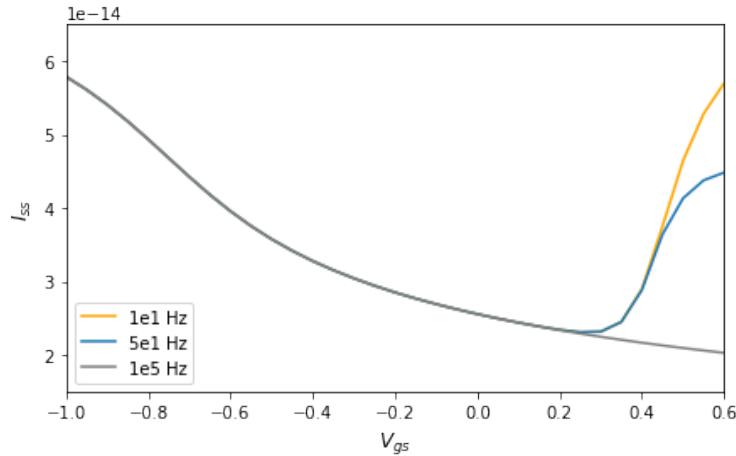
**Figure 13-2 Frequency-dependent capacitance-voltage profile of an example MOS ca-
pacitor. Obtained using a small-signal perturbation to a contact DC voltage sweep,
applied with a harmonic balance boundary condition.**

Here is the small signal analysis boundary condition specification for this analysis, with the
substrate contact held at 0V and the gate contact swept from 0.29V to 0.47V. Note that V_{gs} takes
values in the positive range $[0.29, 0.47]$:

136

```
start Harmonic Balance BC for substrate on silicon
  type is ohmic
  frequencies "0"
  amplitudes "0"
  phase shifts "0"
end Harmonic Balance BC for substrate on silicon
start Harmonic Balance BC for gate on oxide
  type is contact on insulator with work function 4.05
  small signal sweep from 0.29 to 0.47 with amplitude 1e-4
end Harmonic Balance BC for gate on oxide
```

The physically equivalent analysis can be performed with the gate contact held at 0V and the substrate contact swept from -0.29V to -0.47V. Again, V_{gs} takes positive values in the same range. Although numerically different, the output should be the same. This equivalent analysis is specified through a different set of boundary conditions:

```
start Harmonic Balance BC for substrate on silicon
  type is ohmic
  small signal sweep from -0.29 to -0.47 with amplitude 1e-4
end Harmonic Balance BC for substrate on silicon
start Harmonic Balance BC for gate on oxide
  type is contact on insulator with work function 4.05
  frequencies "0"
  amplitudes "0"
  phase shifts "0"
end Harmonic Balance BC for gate on oxide
```

14.   RADIATION MODELS

Charon includes capabilities to model what the effects of various forms of radiation are on semiconductor devices.

14.1.   Empirical Displacement Damage Model

As the name implies the empirical radiation model utilizes data obtained from experiments to model the effects of displacement damage radiation [46]. To accomplish this a table representing this data must be input into the code.

In order to utilize the empirical model first the `physics block` sections for the portion of the problem in which it is active must be specified, for example

```
start physics block {physicsBlockName}
  ...
  empirical damage model is on / off
  ...
end
```

Next a `start empirical model parameters` section is used to specify parameters for the model. That section has the following format

```
start empirical model parameters
  mu table data file is {fileName}
  emitter base junction bounding box is {min} to {max} in x / y / z
  (one parameter per line in the form):
  {parameter name} = {parameter value}
  (see table 14-1 for available parameters)
end
```

Table 14-1 Syntax and parameters for the empirical radiation damage model.

Input file	Description	units
thermal velocity	thermal velocity of carriers	$\frac{cm}{s}$
cross section	collison cross sectional area	cm^2
emitter-base voltage override	voltage across the emitter-base junction	V
Parameters for Analytic Radiation Pulse		
pulse start time	set the start time of the radiation pulse	s
pulse end time	set the end time of the radiation pulse	s
pulse magnitude	magnitude of the peak of the radiation pulse	—
pulse resolution	pulse is sampled at this integer value	—
pulse waveform	analytic form of the pulse (see table 14-2)	—

Table 14-2 Analytic pulse definitions. Unless otherwise noted the pulses are sampled over the range $t = [\text{pulse start}, \text{pulse end}]$ at the resolution given by the user-specified *pulse resolution* parameter.

Function name	Equation
delta	$$\text{pulse}(t) = (\text{pulse magnitude})\,\delta(t - \text{pulse start})$$
square	$$\text{pulse}(t) = \text{pulse magnitude}$$
gaussian	$$T_m = \frac{1}{2}(\text{pulse start} + \text{pulse end})$$ $$P_w = \frac{(\text{pulse start} - \text{pulse end})}{6}$$ $$\text{pulse}(t) = (\text{pulse magnitude})\,\exp\left(-\frac{(t - T_m)^2}{2P_w^2}\right)$$
gaussianlog	$$T_m = \frac{1}{2}\left(\log(\text{pulse start}) + \log(\text{pulse end})\right)$$ $$P_w = \frac{\log(\text{pulse end}) - \log(\text{pulse start})}{6}$$ $$\text{pulse}(t) = (\text{pulse magnitude})\exp\left(-\frac{(t - T_m)^2}{2P_W^2}\right)$$ NOTE: for this waveform $\log(t)$ is incremented rather than t such that the data can encompass large time spans, possibly many decades, for example $t = [10^{-6} : 10^{-3}]$.

14.2.   Total Ionizing Dose Models

Charon provides several models to simulate total ionizing dose (TID) effects.

14.2.1.   *Fixed Oxide Charge Model*

The simplest TID model is to assign positive or negative fixed charges in oxide regions. The effect of fixed charges is included through the Laplace equation for insulators. Currently, only a uniform spatial distribution in a specified box is supported. New analytic spatial profiles can be easily added.

To use the model, one needs to specify it in two places. The first step is to specify

```
bulk fixed charge is on/off
```

in an insulator physics block. When neither option is given, it defaults to `off`. The second step is to add the following block

```
start bulk fixed charge parameters
    distribution is uniform
    charge density = {value} or is swept from {value1} to {value2}
    spatial range is {locMin} to {locMax} in x or in y or in z
end
```

inside a material block. The `distribution` keyword specifies the spatial profile and currently only supports `uniform`. The `charge density` keyword specifies the fixed charge density in units of cm^{-3}, which can be positive for positive charges or negative for negative charges. The `charge density` can also be swept from one density to another density. This is done by using

```
charge density is swept from {value1} to {value2}
```

The `spatial range` keyword specifies a box region where bulk fixed charges are located. The `spatial range` line can be defined for each spatial axis.

An example of specifying a fixed charge density in a defined box is given below. Two complete examples are `oxide.lapl.bq-.inp` for negative charges and `oxide.sg.lapl.bq.inp` for positive charges, located under `tcad-charon/test/nightlyTests/oxidecharge/`.

```
start Material Block SiO2Parameter
     material name is SiO2
     relative permittivity = 3.9

     start bulk fixed charge parameters
        distribution is uniform
        charge density = -1e17
        spatial range is 0.2 to 0.8 in x
        spatial range is 0.1 to 0.3 in y
     end bulk fixed charge parameters
end Material Block
```

An example of sweeping charge density in a defined box is given by

```
start Materal Block SiO2Parameter
     material name is SiO2
     relative permittivity = 3.9

     start bulk fixed charge parameters
```

```
        distribution is uniform
        charge density is swept from -1e16 to -1e17
        spatial range is 0.2 to 0.8 in x
        spatial range is 0.1 to 0.3 in y
     end bulk fixed charge parameters
end Material Block
```

In addition to the fixed charge block, one also needs to specify a `sweep options` block with an example given below. Two complete examples are `oxide.lapl.bq-.loca.inp` for negative charge sweeping and `oxide.sg.lapl.bq.loca.inp` for positive charge sweeping, located under `tcad-charon/test /nightlyTests/oxidecharge/`.

```
start Sweep Options
    Initial Step Size = -1e16
    Minimum Step Size = 1e13
    Maximum Step Size = 5e16
    continuation method is Arc Length
    predictor method is tangent
    successful step increase factor = 2
    failed step reduction factor = 0.5
    Step Size Aggressiveness = 0.5
end Sweep Options
```

Here the values for `initial step size`, `maximum step size`, `maximum step size` are all in units of cm^{-3} for a charge sweeping option.

14.2.2. *Interface Fixed Charge Model*

TID effects in MOSFETs are often modeled by radiation induced charges at semiconductor/insulator interfaces. Charon provides the capability to place fixed positive or negative charges at outer surfaces and/or internal interfaces.

When there exist charges at an interface, the charges affect the electric field at the interface according to Gauss's Law. Given a charge density of σ, in units of cm^{-2}, at an interface between materials 1 and 2, Gauss's Law determines that the gradient of the electric potential needs to satisfy:

$$(\varepsilon_0 \varepsilon_{r2} \nabla \phi_2 - \varepsilon_0 \varepsilon_{r1} \nabla \phi_1) \cdot \eta_1 = q\sigma. \tag{153}$$

Here, q is the elemental charge in units of Coulombs (C), ε_0 is the vacuum permittivity with $\varepsilon_0 = 8.8542 \times 10^{-14} \, \text{C} \, \text{V}^{-1} \, \text{cm}^{-1}$, ε_{r1} and ε_{r2} are the relative permittivity of materials 1 and 2 respectively. $\nabla \phi_1$ and $\nabla \phi_2$ are the potential gradient in units of $\text{V} \, \text{cm}^{-1}$ in materials 1 and 2 respectively, and η_1 is a normal unit vector pointing from 2 to 1. If the charges are located at an outer surface of a simulation device, the potential gradient at the surface is given by:

$$\varepsilon_0 \varepsilon_r \nabla \phi \cdot \eta = q\sigma, \tag{154}$$

where η is a normal unit vector pointing outward from the device.

Two steps are required to use the Neumann charge BC. First, we need to define the charge interface as a sideset in Cubit. Care should be taken to capture the correct potential gradient when

specifiying the mesh density in the vacinity of the interface. For example, if $\sigma = 10^{11}$ cm^{-2} is specified at an interface, the mesh interval adjacent to the interface should not be greater than 1 nanometer (nm). If a higher σ is specified, a mesh size finer than 1 nm at the interface would be required. The general rule of thumb is, the higher the charge density is, the finer the mesh interval required. Next, we need to specify the details of the BC in Charon's input file. The syntax for doing this is

```
start surface charge bc for {sidesetName}
  geometry block is {geometryBlockName}
  fixed charge = {chargeValue}
end
```

where {sidesetName} is the name of the sideset as given during mesh generation in Cubit, {geometryBlockName} is the name of the element block that is adjacent to the surface charge, again, as assigned during mesh generation in Cubit. If a charge sideset is located at the outer boundary of an element block, the associated element block is uniquely defined. If charges are located at an internal interface in a simulation domain then two element blocks are needed at the shared interface. When specifying the interface charge, either of the two element blocks can be used for {geometryBlockName} without affecting the simulation

To verify the accuracy of the charge BC, we first need to consider that all the quantities except energy related fields from Charon are scaled by scaling factors. The scaling factors are internally defined except the Concentration Scaling which can be specified in the Charon input file. Taking into account the scaling factors, the charge BC at an internal interface becomes:

$$(\varepsilon_{r2}\nabla\phi_2^s - \varepsilon_{r1}\nabla\phi_1^s) \cdot \eta_1 = \frac{q\sigma X_0}{\varepsilon_0 V_0}, \tag{155}$$

and the charge BC at an outer surface becomes:

$$\varepsilon_r \nabla\phi^s \cdot \eta = \frac{q\sigma X_0}{\varepsilon_0 V_0}. \tag{156}$$

Here the superscript ^{s} denotes that the potential gradient is scaled and therefore is unitless. X_0 and V_0 are the length and voltage scaling factors respectively. All the scaling factors' values are output to the screen at the beginning of a Charon simulation. In Charon, X_0 is hard-coded to be 10^{-4} cm, and $V_0 = k_B T/q$ with k_B being the Boltzmann constant and T being the lattice temperature. If Lattice Temperature is set to 300 K, V_0 is approximately 0.0258 V.
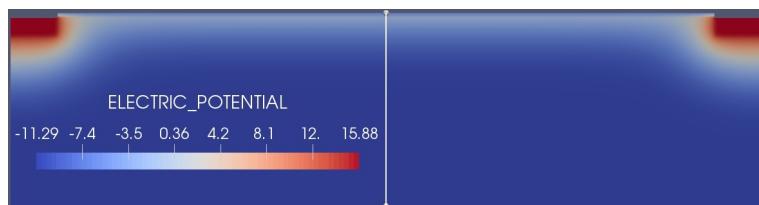
Figure 14-1 Scaled electric potential at equilibrium for a n-channel silicon MOSFET with a positive charge density of 10^{11} cm^{-2} at the Si/SiO$_2$ interface

Figure 14-1 shows the scaled electric potential profile for the n-channel MOSFET example given in tcad-charon/test/nightlyTests/surfacecharge/nmosfet.ifq+.sg.equ.xml. The example has

142

positive charge with a surface density of 10^{11} cm^{-2} at the Si/SiO$_2$ interface. The scaled potential gradient along the white line is shown in figure 14-2. Using the potential gradient values at the Si/SiO$_2$ interface in figure 14-2, the left hand side of Eq. (155) is equal to $11.9 \times 31.49 - 3.9 \times (-83.27) = 699.48$ V cm^{-1}, while the right hand side of Eq. (155) is given by $q\sigma X_0/(\varepsilon_0 V_0) = (1.602 \times 10^{-19} \times 10^{11} \times 10^{-4})/(8.854 \times 10^{-14} \times 0.0258) = 700.75$ V cm^{-1}. Therefore, the potential gradient at the Si/SiO$_2$ interface obtained from Charon indeed satisfies the theoretical relation in Eq. (155).
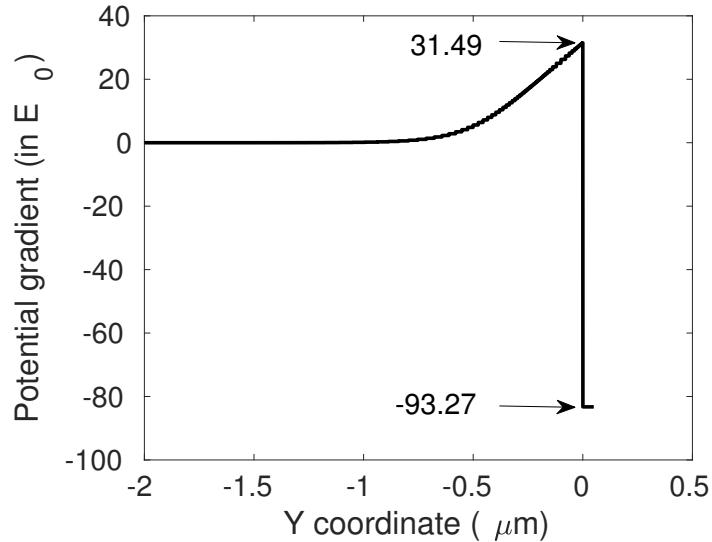
Figure 14-2 Scaled potential gradient along the white line in figure 14-1. The Si/SiO$_2$ interface is located at $y = 0$. The $y < 0$ region is the Si region, while $y > 0$ is the SiO$_2$ region. The denoted numbers are the potential gradients at the Si/SiO$_2$ interface obtained from Charon.

14.2.3. *Interface Static Trap Model*

Charon also provides an interface static trap model that allows to capture voltage dependent threshold voltage shift in MOSFETs due to TID effects. Discrete interface traps act as carrier recombination centers. Traps induced surface recombination rate is given by

$$R_{surf} = \sum_j \frac{np - n_{ie}^2}{(n + n_t^j)/s_p^j + (p + p_t^j)/s_n^j}, \tag{157}$$

where the summation runs over the total number of discrete of traps (e.g., two traps located at two different energy levels). For simpler notation, the superscript j is omitted in the following. n_{ie} is the effective intrinsic concentration in a material. s_p (s_n) is the hole (electron) surface recombination velocity. n_t and p_t for the jth trap are equal to

$$n_t = n_{ie} \exp\left(\frac{E_t}{k_B T}\right), \quad p_t = n_{ie} \exp\left(-\frac{E_t}{k_B T}\right), \tag{158}$$

143

where E_t is the trap energy measured from the intrinsic Fermi level, E_i. E_t is positive for traps above E_i and negative for traps below E_i. k_B is the Boltzmann constant, and T is the lattice temperature. The carrier surface recombination velocity can be specified directly by user or computed from the following expression

$$s_\nu = \sigma_\nu v_\nu N_{it}, \quad \nu = n, \, p. \tag{159}$$

σ is the carrier capture cross section in cm^2, v is the carrier thermal velocity in cm/s, N_{it} is the interface trap density in cm^{-2}. In Charon, the thermal velocity is computed using

$$v_\nu = \sqrt{\frac{3k_B T}{m_\nu^*}}, \quad \nu = n, \, p, \tag{160}$$

where m^* is the carrier effective mass.

Charon supports two types of interface traps, i.e., `Acceptor` and `Donor` traps. `Acceptor` traps are electron traps which are neutral when unoccupied and carry the charge of electrons when occupied. `Donor` traps are hole traps which are neutral when empty and carry the charge of holes when occupied. The occupation probability of the jth `Acceptor` trap is given by

$$f_{tn} = \frac{s_p n + s_n p_t}{s_n(n + n_t) + s_p(p + p_t)}. \tag{161}$$

The occupation function of the jth `Donor` trap is

$$f_{tp} = \frac{s_n p + s_p n_t}{s_p(n + n_t) + s_n(p + p_t)}. \tag{162}$$

The charge density due to occupied acceptor or donor traps contributes to the right hand side (RHS) of the Poisson equation. The charge contribution is equal to $-q\sum_j N_t^j f_{tn}$ for acceptor traps, and it is $q\sum_j N_t^j f_{tp}$ for hole traps, with q being the elemental charge.

Besides the discrete level interface traps, Charon supports continuous energy distribution traps of uniform, exponential or gaussian type, as described in table 14-3, where N_{it} is the peak density, $g_{it}(E)$ is the function describing the distribution and E_σ is the width of the distribution in eV.

<div align="center">

Table 14-3 Available energy distribution for the interface trap model.

</div>

Distribution Type	Description	units
uniform	$g_{it}(E) = N_{it}$ for $E_t - E_\sigma < E < E_t + E_\sigma$	$cm^{-2}ev^{-1}$
exponential	$g_{it}(E) = N_{it}exp(-\frac{|E-E_t|}{E_\sigma})$ for $E_t - E_\sigma < E < E_t + E_\sigma$	$cm^{-2}ev^{-1}$
gaussian	$g_{it}(E) = N_{it}exp(-\frac{(E-E_t)^2}{2E_\sigma^2})$ for $E_t - E_\sigma < E < E_t + E_\sigma$	$cm^{-2}ev^{-1}$

In the case of continuous distributions, the total recombination rate in (157) can be written as

$$R_{surf} = \int_{E_t - E_\sigma}^{E_t + E_\sigma} \frac{np - n_{ie}^2}{\frac{n + n_t(E)}{\sigma_p v_p g_{it}(E)} + \frac{p + p_t(E)}{\sigma_n v_n g_{it}(E)}} dE \tag{163}$$

where $n_t(E) = n_{ie}exp(\frac{E}{k_BT})$ and $p_t(E) = n_{ie}exp(-\frac{E}{k_BT})$. The occupation probabilities in (161),(162) become in this case

$$f_{tn}(E) = \frac{\frac{n}{\sigma_p v_p g_{it}(E)} + \frac{p_t(E)}{\sigma_n v_n g_{it}(E)}}{\frac{n+n_t(E)}{\sigma_p v_p g_{it}(E)} + \frac{p+p_t(E)}{\sigma_n v_n g_{it}(E)}} \tag{164}$$

for `Acceptor` traps and

$$f_{tp}(E) = \frac{\frac{p}{\sigma_n v_n g_{it}(E)} + \frac{n_t(E)}{\sigma_p v_p g_{it}(E)}}{\frac{n+n_t(E)}{\sigma_p v_p g_{it}(E)} + \frac{p+p_t(E)}{\sigma_n v_n g_{it}(E)}} \tag{165}$$

for `Donor` traps. Finally, the charge contributions can be written as

$$Q^A_{trapped} = -q \int_{E_t-E_\sigma}^{E_t+E_\sigma} f_{tn}(E)g_{ti}(E)dE \tag{166}$$

for `Acceptor` traps and

$$Q^D_{trapped} = q \int_{E_t-E_\sigma}^{E_t+E_\sigma} f_{tp}(E)g_{ti}(E)dE \tag{167}$$

for `Donor` traps.

Specification of the interface trap model is similar to that of the interface fixed charge model except that trap related parameters must be given. The mesh size near the interface of interest should be fine as in the interface fixed charge model. The syntax for specifying the model is given below

```
start surface charge bc for {sidesetName}
  geometry block is {geometryBlockName}
  start surface trap
    effective mass for electrons = {eMass}
    effective mass for holes = {hMass}
    start trap {trapID}
      type is {trapType}
      energy distribution is {distType}
      (one parameter per line in the form):
      {parameter name} = {parameter value}
      (see table 14-4 for available parameters)
    end
  end
  start surface recombination
    (one parameter per line in the form):
    {parameter name} = {parameter value}
    (see table 14-4 for available parameters)
  end
end
```

`effective mass for electrons` (holes) corresponds to m_n^* (m_p^*) in Eq. (160). Its value must be given in units of the free electron mass m_0. Multiple types of traps can be given by specifying multiple `start trap` blocks with different `trapID` numbers. `trapID` is an integer equal or greater than 0. `trapType` must be either `Acceptor` or `Donor`. `Acceptor` traps are electron traps which are

145

neutral when empty and carry the charge of one electron when occupied. `Donor` traps are hole traps which are neutral when empty and carry the charge of one hole when occupied. `distType` select the energy distribution type for the traps. The `distType` can be `level`, `uniform`, `exponential` or `gaussian` as described in table 14-3. If the line `energy distribution is distType` is ommited, then the trap has by default a `level` distribution. When a continuous energy distribution type (`uniform`, `exponential` or `gaussian`) is used, then the `energy width` line described in table 14-4 is required.

Another way of modeling surface recombination due to traps is by specifying the carrier surface recombination velocity directly. This is done by specifying the `start surface recombination` block which has three parameters, i.e., `energy`, `surface velocity for electrons`, and `surface velocity for holes`.

Table 14-4 Available parameters for the interface trap model.

Parameter name	Corresponding variables	Description	units
energy	E_t in Eq. (158)	trap energy level measured from the intrinsic Fermi level E_i, positive above E_i and negative below E_i	eV
density	N_{it} in Eq. (159), Table 14-3	trap density for level traps or peak trap density for continuous distributions	cm^{-2} $cm^{-2}ev^{-1}$
cross section for electrons	σ_n in Eq. (159)	electron capture cross section	cm^2
cross section for holes	σ_p in Eq. (159)	hole capture cross section	cm^2
energy width	E_σ in table 14-3	energy width for continuous distributions	eV
surface velocity for electrons	TBW	electron surface recombination velocity	cm/s
surface velocity for holes	TBW	hole surface recombination velocity	cm/s

An example of specifying the interface trap model with two types of traps is given below.

```
start surface charge bc for sheet
    geometry block is silicon
    start surface trap
        effective mass for electrons = 0.25
        effective mass for holes = 0.50
        start trap 0
            type is Acceptor
            density = 1e11
            energy = 0.3
            cross section for electrons = 1e-12
            cross section for holes = 1e-12
        end trap 0
        start trap 1
```

```
            type is Donor
            density = 1e10
            energy = -0.2
            cross section for electrons = 1e-12
            cross section for holes = 1e-14
        end trap 1
    end surface trap
end surface charge bc for sheet
```

Several complete examples on the model can be found at `test/nightlyTests/surfacecharge`
including `nmosfet.sg.it.equ.inp`, `nmosfet.sg.it.loca.inp`, and `nmosfet.dd.it.a1e11.equ.inp`.

14.2.4. *Kimpton Model*

Kimpton model is a more advanced TID model for radiation tolerant insulators where
radiation-induced charge and threshold voltage shift depend not only on the radiation dose but
also on the electric fields. It is a trapping-detrapping model taking into account interface and
volume traps distributed within the irradiated insulator.

The Kimpton model assumes that the exposure to ionizing radiation results in excess positive
charge trapping in the insulator (gate oxides in MOSFETs) due to the predominance of hole traps.
In radiation-tolerant insulators the hole traps behave as energy wells with no significant bonding
allowing reversible trapping-detrapping processes. Detrapping process of the trapped carriers is
assisted by the energy released from the recombination of some of the electron-hole pairs
generated during irradiation.

For positive electric fields (holes are pushed towards the interface by the field) inteface traps and
hole trapping/detrapping at the interface account for the thresholds voltage shifts for positive
electric fields greater than 0.1 $\frac{MV}{cm}$. The trapping, detrapping and conservation equations for
interface traps are given by

$$
\begin{aligned}
p_{\text{ttp}}^i &= \left(N_t^i - N_{\text{tf}}^i\right)\sigma E_{\text{ins}}^{\text{-x}} f_{(\text{ins})} n_{\text{pairs}}^{\text{gen}}, \\
p_{\text{tde}}^i &= N_{\text{tf}}^i \sigma E_{\text{ins}}^{-x}\left(1 - f_{(\text{ins})}\right) n_{\text{pairs}}^{\text{gen}}, \\
p_{\text{ttp}}^{i,\text{net}} &= N_{\text{tf}}^i + p_{\text{ttp}}^i - p_{\text{tde}}^i
\end{aligned}
\tag{168}
$$

where p_{ttp}^i is the interface density of new holes trapped after a small dose irradiation, p_{tde}^i is the
interface density of detrapped holes assisted by the energy released from geminate recombination
of electron-hole pairs after the small dose irradiation, $p_{\text{ttp}}^{i,\text{net}}$ is the net interface density of trapped
holes, N_t^i is the interface trap density, N_{tf}^i is the interface occupied trap density before irradiation,
σ is the trap capture cross section for a positive field of 1 $\frac{MV}{cm}$, E_{ins} is the electric field in the
insulator, x is a constant depending on technology, $n_{\text{pairs}}^{\text{gen}}$ is the electron-hole pair density
generated during irradiation and $f_{(\text{ins})}$ is the fractional hole yield.

Electron-hole pairs density $n_{\text{pairs}}^{\text{gen}}$ generated during irradiation is given by

$$
n_{\text{pairs}}^{\text{gen}} = \rho_{\text{ins}} 10^{-5} Dose \frac{C_{\text{DEF}}}{qE_{\text{form}}}
\tag{169}
$$

where ρ_{ins} is the insulator mass density in $\frac{g}{cm^3}$, *Dose* is the radiation dose in *rad*, C_{DEF} is the effective dose enhancement factor and E_{form} is the effective electron-hole pair formation energy.

The fractional hole yield $f_{(ins)}$ also depends on the electrical field in the insulator E_{ins} as

$$f_{(ins)} = \left[\left(\frac{1.35}{E_{ins}} \right) - 1 \right]^{-0.9} \tag{170}$$

Charon computes the net interface density of trapped holes $p_{ttp}^{i,net}$ and add the charge to the Laplace equation solved in the irradiated insulator. The user can visualise the induced charge in the irradiated insulator and perform subsequent simulations for computing the threshold voltage shifts due to radiation induced charge.

To use the Kimpton model with interface traps in an insulator, the user must first turn on the TID model in the appropriate physics section

```
start Physics Block {physicsBlockName}
  ...
  tid is on
  ...
end
```

and specify the Kimpton model with general parameters defined in table 14-5 and interface parameters defined in table 14-6 inside the tid section of the material block section

```
start Material Block {materialBlockName}
  start tid
    start kimpton model
      (one parameter per line in the form):
      {parameter name} = {parameter value}
      (see table 14-5 for available parameters)

      start interface traps
        sideset id is {tidInterface}

        (one parameter per line in the form):
        {parameter name} = {parameter value}
        (see table 14-6 for available parameters)
      end
    end
  end
end
```

where *tidInterface* is the sideset name of the interface between the irradiated insulator and the semiconductor where the interface traps are located.

For small positive electric fields ($< 0.1 \frac{MV}{cm}$) and negative electric fields the Kimpton model with interface traps only fails to match experimental data. The difference can be accounted for by

Table 14-5 Syntax and general parameters for Kimpton model.

Input file	Corresponding variable in (168), (169), (170)	Description	units
dose	*Dose*	ionizing radiation dose applied	*rad*
effective dose enhancement factor	C_{DEF}	effective dose enhancement factor	*none*
electron-hole pair formation energy	E_{form}	electron-hole pair formation energy	*eV*
electric field power dependency	x	electric field power dependency	*none*

Table 14-6 Syntax and parameters for Kimpton interface trap model.

Input file	Corresponding variable in (168), (169), (170)	Description	units
total density	N_t^i	interface trap total density	cm^{-2}
initial filling factor	$\frac{N_{\text{tf}}^i}{N_t^i}$	initial filling factor	*none*
capture cross section	σ	trap capture cross section at $1\frac{MV}{cm}$	cm^2

considering trapping/detrapping in oxide volume. The trapping, detrapping and conservation equations for volume traps are given by

$$
\begin{aligned}
p_{\text{ttp}}^v &= P_{\text{capt}} \left(N_t^v - N_{\text{tf}}^v \right) T_{\text{vol}}^{\frac{1MV}{cm}} E_{\text{ins}}^{-1.5x} f_{(\text{ins})} n_{\text{pairs}}^{\text{gen}}, \\
p_{\text{tde}}^v &= \left(1 - P_{\text{capt}} \right) N_{\text{tf}}^v T_{\text{vol}}^{\frac{1MV}{cm}} E_{\text{ins}}^{-1.5x} f_{(\text{ins})} n_{\text{pairs}}^{\text{gen}}, \\
p_{\text{ttp}}^{v,\text{net}} &= N_{\text{tf}}^v + p_{\text{ttp}}^v - p_{\text{tde}}^v
\end{aligned}
\tag{171}
$$

where $T_{\text{vol}}^{\frac{1MV}{cm}}$ is the capture volume of the trap at $1\frac{MV}{cm}$, P_{capt} is the probability of hole capture from an electron-hole pair generated within a trap volume, p_{ttp}^v is the volume density of the new holes trapped after a small dose irradiation, p_{tde}^v is the volume density of the detrapped holes after the small dose irradiation, $p_{\text{ttp}}^{v,\text{net}}$ is the net volume density of trapped holes after irradiation, N_t^v is the volume trap density and N_{tf}^v is the occupied trap volume density before irradiation.

Because the effective trap capture cross section $\sigma_{\text{eff}} = \sigma E_{\text{ins}}^{-x}$, then the effective capture volume of the trap becomes also field dependent $T_{\text{vol}}^{\text{eff}} = \frac{4}{3}\pi \left(\frac{\sigma_{\text{eff}}}{\pi} \right)^{\frac{3}{2}} = T_{\text{vol}}^{\frac{1MV}{cm}} E_{\text{ins}}^{-1.5x}$. For small electric fields the effective trap volume saturates to the value $\left(T_{\text{vol}}^{\text{eff}} \right)_{\text{max}} = \frac{1}{2N_t^v}$ by imposing the condition that the total maximum capture volume for the traps to be limited to half of the volume of the

149

insulator. The probability P_{capt} is computed as

$$
\begin{aligned}
P_{\text{capt}} &= \frac{V_{\text{crit}}}{V_{\text{crit}} + T_{\text{vol}}^{\text{eff}}}, \\
V_{\text{crit}} &= \frac{4}{3}\pi \left(\frac{\sigma_{\text{crit}}}{\pi}\right)^{\frac{3}{2}}
\end{aligned}
\tag{172}
$$

where V_{crit} represents the core volume around the trap within $T_{\text{vol}}^{\text{eff}}$ where the probability of capturing the hole is greater than pair recombination and σ_{crit} is a user specified parameter.

In general volume traps are used in addition to interface traps in Kimpton model. To do this, the user must specify in addition to what has been specified for the interface traps a volume parameter section for the Kimpton model

```
start Material Block {materialBlockName}
  start tid
    start kimpton model
      (one parameter per line in the form):
      {parameter name} = {parameter value}
      (see table 14-5 for available parameters)

      start interface traps
         sideset id is {tidInterface}

         (one parameter per line in the form):
         {parameter name} = {parameter value}
         (see table 14-6 for available parameters)
      end

      start volume traps
         (one parameter per line in the form):
         {parameter name} = {parameter value}
         (see table 14-7 for available parameters)
      end
    end
  end
end
```

Table 14-7 Syntax and parameters for Kimpton volume trap model.

Input file	Corresponding variable in (171), (172)	Description	units
total density	N_t^v	volume trap total density	cm^{-3}
initial filling factor	$\frac{N_{tf}^v}{N_t^v}$	initial filling factor	none
capture cross section	σ	volume trap capture cross section at $1\frac{MV}{cm}$	cm^2
critical capture cross section	σ_{crit}	critical capture cross section	cm^2

To enable threshold voltage shift evaluation or the impact of the ionizing radiation dose on the device functionality, Charon allows the positive charge trapped on the interface and volume traps in the insulator to be frozen after a radiation dose and be used as fixed charge for a bias sweep. For instance, if a radiation dose is applied to the gate oxide at zero bias and then removed, by sweeping the gate bias the $I_d - V_g$ curve can be obtained and the threshold voltage shift can be evaluated.

To freeze the trapped charge after an irradiation dose the user must specify in the general section of the Kimpton model

```
start Material Block {materialBlockName}
  start tid
    start kimpton model
      ...
      freeze traps at voltage = {V} on {contact}
      ...

    end
  end
end
```

where V is the bias voltage on *contact* such that for voltages greater than V, trapped charge in the insulator becomes fixed charge for a voltage sweep on *contact*.

15.   SOLVERS

The solvers in Charon are supplied by the Trilinos set of solver libraries. Because the drift-diffusion equations are nonlinear and must be solved iteratively, a nonlinear Newton solver implemented in the NOX package of Trilinos is used. Within each Newton iteration, a matrix equation set must be solved for the Newton update. This linear problem is solved using various iterative solvers, and preconditioners for those solvers, all of which are also contained within Trilinos. For the current release of Charon the preferred suite of linear solvers is based around the Trilinos package Tpetra. The minimum necessary specification of solver settings for a given problem type is

```
start solver block
  start tpetra block
    problem type is {problemType}
  end tpetra block
end solver block
```

Currently supported problem types are summarized in table 15-1.

Table 15-1 Available problem types for Tpetra solver settings

Problem Type	Description
steady state	A steady state problem with no transient, parametric, or boundary variations
constrained steady state	A steady state problem with some type of constraint placed on it. Examples include a resistor attached to a contact or a specified constant current at a contact.
parameter sweep	An automated simulation of a sequence of steady state problems obtained by sweeping some parameter. Examples include increasing the applied potential on a contact or increasing the concentration of a dopant.
constrained parameter sweep	A combination of a problem with a constraint and a parameter sweep.
transient	Typical transient simulation. For example, a sinusoidal voltage being applied to a contact.

When possible it is best to leave the settings at their default values for a given problem type, as shown above. General syntax for the settings block is

```
start tpetra block
   (see table 15-2 for more information)
   problem type is {problemType}
   verbosity level is low / medium / high / extreme
   (see equation 175 for details on WRMS)
   start nonlinear solver wrms block
     test is on / off
     relative tolerance = {relTolerance}
```

```
    absolute tolerance = {absTolerance}
    alpha = {alphaVal}
    beta = {betaVal}
  end nonlinear solver wrms block
  maximum number nonlinear iterations = {maxIters}
  nonlinear residual norm convergence = {normF}
  linear solver is aztecOO / belos
  linear solver tolerance = {tolerance}
  preconditioner is ilut / riluk / schwarz
  (for transient simulations)
  final time = {fTime}
  time step size is fixed at {deltaT} / variable
  (for variable time stepping)
  initial step size = {initDeltaT}
  minimum step size = {minDeltaT}
  maximum step size = {maxDeltaT}
end tpetra block
```

Table 15-2 Available problem types for Tpetra solver settings

Solver setting	Description
verbosity level	How much screen output should be performed giving details about solver performance.
maximum number nonlinear iterations	An integer specifying how many Newton steps to take during a nonlinear solve. If the number is reached without achieving convergence of all the other specified nonlinear convergence tests the nonlinear solver step is marked as a failure (Default: 30).
nonlinear residual norm convergence	The value of the nonlinear residual at which to consider the test converged. Note there may be other tests such that even if this test is converged the nonlinear solver iteration itself may not be considered converged. (see eq. 173) (Default: 1.0e-4).
WRMS convergence test details	
relative tolerance	ε_{rel} in eq. 175
absolute tolerance	ε_{abs} in eq. 175
beta	The achieved linear solver tolerance must be less than *beta* for convergence. (see eq. 176) (Default: 1.0).
alpha	If a line-search based solver is used the step size must be greater than *alpha* for convergence. (see eq. 177) (Default: 0.0)
Variable time stepping settings	
initial step size	Initial value of time step.
maximum step size	The maximum size of the time step.
minimum step size	The minimum size of the time step.
final time	Final value of time at which to stop the simulation.

The stopping criteria for Newton iterations determines when the nonlinear iterations are declared converged. The first convergence criterion is that the norm of the residual be smaller than a specified value,

$$||\mathbf{F}|| < \varepsilon_{normF} \tag{173}$$

where $||\mathbf{F}||$ is the norm of the residual.

The second criterion is that a *WRMS* norm is satisfied,

$$||\delta x^k||_{wrms} < \varepsilon_{wrms} \tag{174}$$

The *WRMS* norm is:

$$||\delta x^k||_{wrms} := \sqrt{\frac{1}{N}\Sigma_{i=1}^N \left(\frac{x_i^k - x_i^{k-1}}{\varepsilon_{rel}|x_i^{k-1}| + \varepsilon_{abs}}\right)^2} \tag{175}$$

In this equation, x^k is the vector of degrees of freedom in the system at the kth nonlinear iteration. x_i^k is the ith component of that vector, N is the number of degrees of freedom in the system and ε_{rel} and ε_{abs} are user-specified relative and absolute error tolerances. Two additional parameters can also be specified.

Even when the linear solver mentioned above doesn't satisfy its solver tolerance, Charon will still try to use that solution as a Newton update. If the linear solver isn't converging to a sufficient degree, the *WRMS* stopping criterion can cause stagnation in nonlinear iterations. This can be mitigated by specifying the β parameter for the *WRMS* criterion such that,

$$\eta < \beta \tag{176}$$

where η is the norm of the residual of the linear solver. Setting $\beta = 1$ removes this criterion.

Sometimes when Newton iteration updates are very small, the *WRMS* norm will detect stagnation in the solver process. This can be mitigated when the line search method is used by requiring the fraction of the Newton update as determined by the line search to be greater than a specified value, α.

$$\lambda > \alpha \tag{177}$$

The quantity λ is the fraction of the Newton step calculated by the line search method. Setting $\alpha = 0$ eliminates this criterion.

16.   CHARON INPUT FILE REFERENCE

The following sections are a reference to the various models in Charon and how they're called out in the input file. All of the input file lines are case insensitive except where the input is a variable. In the tables that describe the syntax, these are bracketed in the main syntax line and *italicized* in every case. The top cell in each of the tables represents a single line—indeed, it *must occupy* a single line in the input file with no line breaks.

Occasionally, the syntax may required that at least one option be specified–On/Off for example. In the **Required** column of each table, it will indicate when an option is required. This does not mean that each option is required, only that at least one of the options is required. For example, the line

```
unscale variables is on
```

could equally be specified

```
unscale variables is off
```

This a case where "is on" and "is off" represent two different options for the interpreter line and at least one of them must be specified.

16.1.   Import State File

File I/O is a critical part of any Charon execution. At a minimum, Charon must import a file that contains a mesh. It can also include a state, i.e. a full set of potential, carrier concentration, temperature, etc at a given parameter set. This file is called a state file even when the state is null—in other words, contains only a mesh. The mesh is a necessary part of any simulation employing finite element or finite volume methods as Charon does. The state will be used as an initial guess of the solution being sought in the present simulation.

Often times, a state file can contain multiple states. For example, the state file may have been produced by a previous simulation that swept over a range of contact voltage. If states were calculated at ten different values of the voltage in the previous simulation, then there will be ten states in the file that it produced numbered 0-9. By default, Charon will import the final state in the file.

The input file line to import the state file into the present simulation exists at the root level of the interpreter input file.

Table 16-1 Import state file syntax

Import State File {*filename*} [at Index {*index*}]		
Option	Description	Required
at Index	Specifies that a specific state will be used as an initial guess	No
Variables	Description	Default
filename	Specifies the name of the state file	*none*
index	integer index of the state file to use.	-1

16.2.   Output Parameters

The data generated by Charon execution are written to output files of two different types. The primary output file is an exodus formatted record of all states generated during execution plus other user-requested output. Text files are generated for tabulated output of current as a function of time for transient simulations and as a function of parameter for parameter sweeps.

User control of how the output is configured is done in the output parameters block. The block sits at the root level and is formatted in the input file as:

```
start output parameters
      output specifications
end output parameters
```

The output specifications are individual lines contained in the block and are described in the remainder of this section.

Table 16-2 Output state file syntax

Output State File {*filename*}		
Option	Description	Required
	None	
Variables	Description	Default
filename	Specifies the name of the exodus formatted state file for output	*none*

By default, degree-of-freedom variables (potential, carrier concentration, temperature) are written to the exodus file at node locations in the underlying mesh. The global variables of contact current are also written to the exodus file by default. Other incidental variables are not written by default, but can be requested. The variables can be requested as output in two different ways. One is at nodal locations just as the degree of freedom variables are recorded. The other is as a cell average quantity. Some quantities, particularly those that are gradients of other variables are not well defined at nodes. These must be output as cell averaged quantities.

In the output variables lines, any number of variables may be specified. This requires a special form of input. When specifying the variables, the full section must be in s̈. And the variable

names must be comma separated. As in all interpreter variables, the output variable names are case sensitive.

Table 16-3 Output nodal variables syntax

Output nodal variables in {*regionName*} for *{variableNames}*		
Option	Description	Required
	None	
Variables	Description	Default
regionName	This is the geometry block where the variable is defined	*None*
variableNames	Names of the variable to be output to the exodus file	*None*

Table 16-4 Output cell averaged variables syntax

Output cell average variables in {*regionName*} for [scalar *{variableNames}* [vector *{variableNames}*]]		
Option	Description	Required
scalar	The variables are all scalars	Yes
vector	The variables are all vectors	Yes
Variables	Description	Default
regionName	This is the geometry block where the variable is defined	*None*
variableNames	Names of the variable to be output to the exodus file	*None*

Any simulation that produces multiple states are either transient or they are sweeps of some parameter in the model. As stated earlier, the current at each contact can be printed as a function of time or parameter value to a text file. The name of the file to which these values are printed can be specified by the appropriate line as described in Tables 16-5 and 16-6.

Table 16-5 Specify filename of tabulated currents for transient simulations

Output tabulated transient currents to {*filename*}		
Option	Description	Required
	None	
Variables	Description	Default
filename	Name of the file where currents in a transient simulation are printed	*None*

Finally, because the scaling of the drift-diffusion equations is somewhat unwieldy, the solution of them can be numerically difficult. For this reason, the variables of the equations are scaled by characteristic quantities. They are also written to the exodus file as scaled quantities. If the user wishes unscaled quantities in the exodus file, they can request it in the output parameters block.

Table 16-6 Specify filename of tabulated currents for parameter sweep simulations

Output tabulated parameter currents to {*filename*}		
Option	Description	Required
	None	
Variables	Description	Default
filename	Name of the file where currents in a parameter sweep simulation are printed	*currents-loca.dat*

Table 16-7 How to request unscaled quantities in the exodus file.

Unscale Variables [is on [is off]]		
Option	Description	Required
is on	Turn unscaling on	Yes
is off	Turn unscaling off	Yes-Default
Variables	Description	Default
	None	

16.3.   Physics Block

This section describes input lines that appear at the level of the physics block. Any description of physics in a semiconductor or insulating region of a device simulation must have a physics block associated with it. Any one physics block can be associated with multiple regions of a device. For instance, two semiconducting regions may be separated by an insulator. The physics block will contain a description of all physical mechanisms represented in the simulation for that region. Because there may be multiple physics blocks, each one must have a unique name—no spaces please—to differentiate it from the others. The syntax to create a physics block follows:

```
begin phsyics block {physicsBlockName}
      geometry block is {geometryBlockName}
      material model is {materialModelName}
      standard discretization type is {discType}
      discretization method is {discMethod}
      {other physics descriptions}
end physics block {physicsBlockName}
```

16.3.1.   Geometry Block

One of the elements that must always be in a physics block is the geometry block specification using the `geometry block is {geometryBlockName}` line. This line ties the physics block to at least one region of the device as it was created and named during the meshing stage of creating a simulation. Table 16-8 describes the line in the physics block that specifies the geometry block.

In an instance where there are multiple regions the have identical physics descriptions, this line may be repeated for each named region,

geometry block is *geometryBlockName*		
Option	Description	Required
	None	
Variables	Description	Default
geometryBlockName	Name that corresponds to a geometry block specified in the meshing step.	*None*

Table 16-8 Specificy the geometry block to which these physics apply.

16.3.2. Material Block

Each phsyics block must be accompanied by a material model block located elsewhere in the input file. This is done using the `material model is {materialModelName}` line. As in the case of the physics block, there can be a plurality of material blocks each with a unique name. The phsyics block and material block are linked by specifiying the name of the material model name in the appropriate physics block. Only one material block may be specified for each physics block and it is done so in a single line inside the phsyics block as described in Table 16-9.

material model is *materialModelName*		
Option	Description	Required
	None	
Variables	Description	Default
materialModelName	Name that corresponds to a named material model block	*None*

Table 16-9 Specify the material model that accompanies a phsyics block.

16.3.3. Discretization

The transport models and discretization methods available in Charon are described in Chapter 1. They are specified using the lines of `standard discretization type is {discType}` and `discretization method is {discMethod}`. Available `discType` and `discMethod` values are listed in Table 1-2.

16.4. Material Block

The material block is a complement to the physics block. Between the two, they give a complete description of the material properties and physical mechanisms of the device in the simulation.

The material block contains all properties associated with the material contained in its associated physics block which in turn is associated with the geometry block specified in the physics block. Because there can be multiple materials in any one simulation, the material block must be uniquely named. The syntax to create a material block is

```
begin material block {materialBlockName}
      {material descriptions}
end material block {materialBlockName}
```

where materialBlockName is the unique name of the material model.

16.4.1.   *Material Name*

In some instances, common materials have properties pre-programmed into Charon, e.g. silicon and Gallium Arsenide. To associate those properties with the material block, the name of the material is specified in the material block. Note that the material name, specified in the material name line is case sensitive. The syntax is described in Table 16-10.

Table 16-10 Specificy the name of a material.

Material name is *materialName*		
Option	Description	Required
	None	
Variables	Description	Default
materialName	Specify the name of the known material for default properties	*None*

16.4.2.   *Relative Permittivity*

Relative permittivity is a material parameter that must be specified for all materials in a Charon simulation. For some materials that have properties already programmed into Charon, there will be an associated default value. In practice, it's a good idea to specifiy the value explicitly and it can always be specified as a value to supercede the default value. The relative permittivity is specified in the material block in a single line in the syntax descritbed in Table 16-11.

16.5.   Carrier Recombination

Charon presently supports four types of carrier recombination models. They are Schockley-Reed-Hall (SRH) in both mid-gap and generic forms, direct and auger recombination. For details on how each is implemented, refer to section 8. To enable any type of recombination, entries are required in both the physics blocks associated with the region where recombination takes place and in the accompanying material model block. In the physics block, individual types of recombination are toggled on and off,

Table 16-11 Specificy the relative permittivity of a material.

Relative Permittivity = *relativePermittivity*		
Option	Description	Required
	None	
Variables	Description	Default
relativePermittivity	Numerical value of the relative permittivity of the material	*default for known material, 0 otherwise*

```
start physics block {physicsBlockName}
    <recombinationType> is on/off
end physics block
```

recombination toggle end physics block Parameters for each type of recombination are specified in the relevant material model block. The following sections tabulate the input lines required to toggle recombination on and off and set relevant parameters.

16.5.1. *Shockley-Reed-Hall Recombination*

SRH recombination is enabled for mid-gap traps or generic traps wherein the trap energy may be specified for any energy level in the band gap. The mid-gap recombination model is toggled on and off in the physics block via,

Table 16-12 Mid-gap SRH recombination toggle

srh recombination [is on [is off]]		
Option	Description	Required
is on	Toggle mid-gap recombination on	Yes
is off	Toggle mid-gap recombination off	Yes
Variables	Description	Default
	None	

Parameters for mid-gap SRH recombination are set in the carrier lifetime block inside the material model block,

```
start Material Block {materialBlockName}
   start Carrier Lifetime Block
      <set parameters>
   end Carrier Lifetime Block
end Material Block
```

For mid-gap SRH traps, the carrier lifetimes may be constant or concentration or temperature dependent. If the carrier lifetimes are concentration dependent.

Table 16-13 Mid-gap SRH constant carrierlifetime

carrierType lifetime is constant = *lifetime*		
Option	Description	Required
	None	
Variables	Description	Default
carrierType	Specified as either elctron or hole	None
lifetime	Numerical value of the lifetime in seconds	Material Dependent

Table 16-14 Concentration dependent SRH carrier lifetime

carrierType lifetime is concentration dependent [with Nsrh = *NsrhParameter*]		
Option	Description	Required
with Nsrh	Use a specified the parameter value for the concentration dependent recombination	No
Variables	Description	Default
CarrierType	Specified as either elctron or hole	None
Nsrh	Numerical value of the parameter for concentration dependent lifetime	Material Dependent

Table 16-15 Concentration dependent SRH carrier lifetime tau_0 value

electron tau0 = *tau0*		
Option	Description	Required
	None	
Variables	Description	Default
tau0	Electron Tau_0	Material dependent

Table 16-16 Electron lifetime temperature dependence

electron lifetime is temperature dependent		
Option	Description	Required
	None	
Variables	Description	Default
	None	

To specify that the mid-gap recombination model is temperature dependendent,

The generic SRH trap model is toggled with

Table 16-17 Generic SRH recombination toggle

trap srh [is on [is off]]		
Option	Description	Required
is on	Toggle generic recombination on	Yes
is off	Toggle generic recombination off	Yes
Variables	Description	Default
	None	

16.5.2. *Radiative (Direct) Recombination*

Radiative, or direct, recombination for direct bandgap semiconductors is the simplest of the recombination models. It can be fully employed with the usual on/off toggle directive in the physics block and a specification of a single coefficient in the accompanying material block.

Table 16-18 Direct recombination toggle

radiative recombination [is on [is off]]		
Option	Description	Required
is on	Toggle radiative recombination on in physics block	Yes
is off	Toggle radiative recombination off in physics block	Yes
Variables	Description	Default
	None	

Parameters for mid-gap SRH recombination are set in the carrier lifetime block inside the material model block,

```
start Material Block {materialBlockName}
   Radiative recombination coefficient = {coefficient}
end Material Block
```

Table 16-19 Radiative recombination coefficient

Radiative recombination coefficient = *coefficient*		
Option	Description	Required
	None	
Variables	Description	Default
coefficient	coefficient for radiative recombination	None

16.5.3. *Auger Ionization/Recombination*

The Auger model can account for both recombination and generation of charge carrier pairs. The details of the model are described in Section 8.3. In addition to toggling the model on as described

above, coefficients for the Auger model may be defined in the material model block as follows.

```
start Material Block materialBlockName
   start Auger Recombination Parameters
      Auger Coefficient electron/hole = augerCoefficient
      Generation on/off
   end Auger Recombination Parameters
end Material Block materialBlockName
```

16.6.   Variable Voltage at a Boundary

In order to perform current-voltage sweeps, or IV sweeps, the voltage at a contact must be varied over the desired range. In Charon this capability is implemented using the Library of Continuation Algorithms (LOCA) library. A specific example of how to use this capability was covered in section 3.5.1. More generally, to use this capability the first step is to modify the boundary condition associated with the contact over which voltage will be varied. The syntax for doing this is

```
bc is ohmic for {contact name} on {material name} swept from {start
   voltage} to {stop voltage}
```

Next a section for controlling how the voltage will be swept needs to be added. The syntax for that is

```
start sweep options
  continuation method is arc length / natural
  predictor method is constant / secant / tangent
  (One parameter per line)
  (see table 16-20 for available parameters)
end sweep options
```

It is important to note that because of the use of LOCA for voltage stepping within Charon there are a wide range of simulations possible. For example, while generally an IV sweep proceeds in a fairly straightforward way from the *start voltage* to the *stop voltage*, with LOCA that doesn't necessarily have to be the case.

Consider figure 16-1. The red line might represent the results of a typical IV sweep on, for example, a semiconductor resistor. The simulation starts at V_{start} and proceeds in a straightforward manner to V_{stop}. In this case V_{start} is the minimum value that the voltage will ever attain and so there is no need to set *min value* in the sweep parameters to anything other than *start voltage*, and Charon will default to setting *min value* to *start voltage* if *min value* isn't set in the input file.

Now consider a simulation in which the I-V characteristics are more complex, as represented by the green line in figure 16-1. In this case V_{start} will not be the minimum voltage. Charon is capable of performing such a simulation but you must set *min value* in the sweep parameters to something other than V_{start} since not doing so will cause the simulation to terminate once V curves back to V_{start}. In this situation the sweep option *max value* is redundant since the simulation will terminate if the voltage reaches either *stop voltage* or *max value*. For IV simulations in the reverse direction the situation is mirrored and the value of the sweep option *max value* becomes relevant in the same way *min value* is relevant in the forward case..

Table 16-20 Parameters used to control voltage sweeping.

Input file	Description	Default
initial step size	initial voltage increment	1.0
maximum number of steps	If the number of voltage steps taken reaches this value then the simulation is terminated.	*none*
minimum step size	If the voltage step size is reduced to this value due to step failures then the simulation is terminated.	$1.0x10^{-12}$
maximum step size	The voltage step size will never exceed this value.	$1.0x10^{12}$
step size aggressiveness	Floating point value between 0 and 1 that represents how aggressive the code should be about changing the step size.	1.0
successful step increase factor	If the code determines that a larger voltage step is possible it will increase the size by this factor.	1.26
failed step reduction factor	If the voltage step attempt fails then reduce the step size by this factor and try again.	0.5
min value	If the voltage value falls below this value then the simulation is terminated. See the text in section 16.6 for how this can be different than the starting value in the *bc* line.	*start voltage* or *stop voltage*
max value	If the voltage exceeds this value then the simulation is terminated. See the text in section 16.6 for how this can be different than the ending value in the *bc* line.	*start voltage* or *stop voltage*

16.7.   Doping

Doping is specified in the relevant `material block` section of the input file. In general doping is specified via a set of analytic functions within the area of the associated material block. Functions are additive, unless otherwise noted, so each additional function that is specified adds to the previous ones that were specified for that material block.

Additionally a file may be specified which consists of a list of points and a value for doping at that point. Note that due to interpolations that are required for file doping specifications this can negatively impact the performanace of the code.

Each of the following subsections gives information on the syntax for each specific doping function available.

16.7.1.   Uniform Doping

Uniform doping places a constant value of doping over the specified area. You can specify uniform doping via
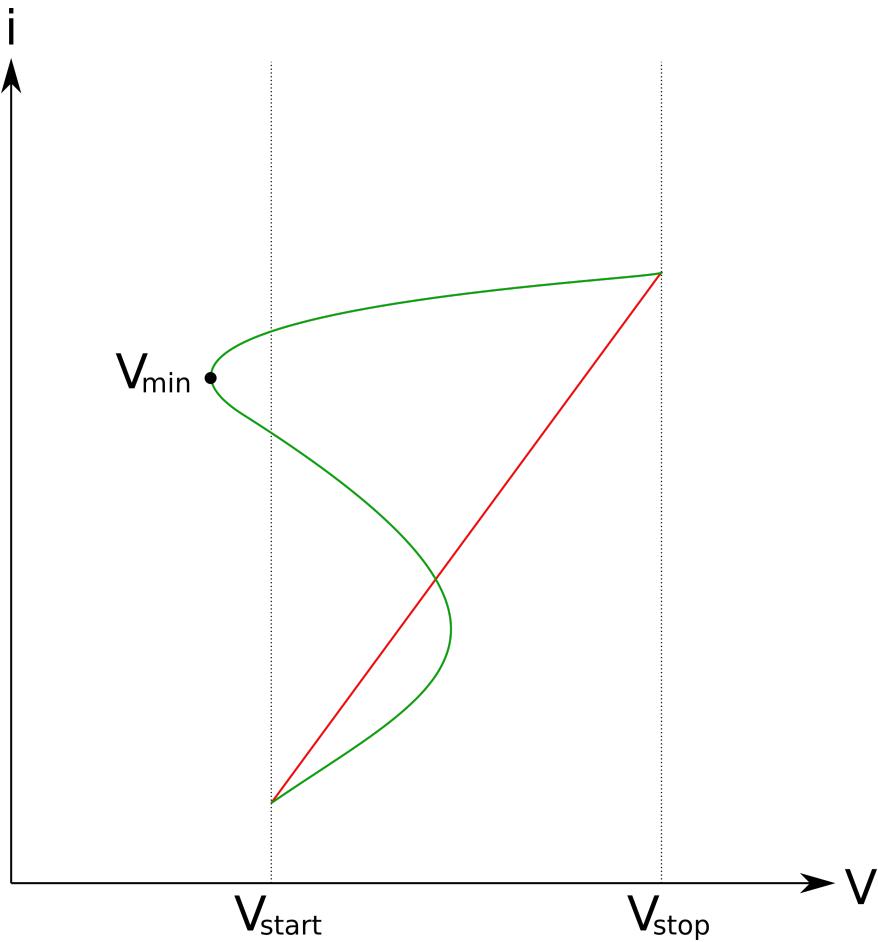
Figure 16-1 Example of a non-typical IV sweep possible in Charon.

```
start material block {materialBlockName}
  ...
  start uniform doping named Function{i}
    spatial range is {minimum} to {maximum} in x / y / z
    type is donor / acceptor
    concentration = {value in #/cm² }
    gauss decay with start position at {pos} and decay width at {width} in +x / -x / +y
        / -y / +z / -z
  end
  ...
end
```

Where the *{i}* in `Function{i}` is replaced with an integer starting with 1 and going to the number of doping functions in the specific material block. You can use a `spatial range` specification for each coordinate direction when necessary. If a particular direction isn't specified then the doping will be applied for the entire block in that coordinate direction. Optionally, a Gaussian decay can be specified using the `gauss decay` line along one or multiple axises. The Gaussian decay factor is defined as $exp(-(x-x_0)^2/(2\sigma^2))$ with x_0 given by `pos` and σ given by `width`. The doping along the Gaussian decay axis is the uniform doping multiplied by the Gaussian decay factor.

16.7.2.   Step Junction

You can use step doping for a doping profile that varies as a step function within the specified material with the specified parameters. In Charon the syntax for the step function is

```
start material block {materialBlockName}
  ...
  start step junction doping
    acceptor concentration = {N_A}
    donor concentration = {N_D}
    junction location = {coordinate value}
    direction is x / y / z
    dopant order is pn / np
  end
  ...
end
```

All of the statements shown above must be included for a valid step junction specification.

16.7.3.   Gaussian Doping

You can use Gaussian doping for a doping profile that varies as a Gaussian function within the specified material with the specified parameters. In Charon the form of the Gaussian function, assuming a variation in the *x* direction, is:

$$ N = \{N_{\max}\} \exp\left[-\ln\left(\frac{\{N_{\max}\}}{\{N_{\min}\}}\right) \left(\frac{x - \{peak\}}{\{width\}}\right)^2 \right] \tag{178} $$

The syntax for Gaussian doping is

```
  start material block {materialBlockName}
    ...
    start gauss doping named Function{i}
      concentration range is {N_{min}} to {N_{max}} in x / y / z
      spatial range is {minimum} to {maximum} in x / y / z
      gradient width = {width} in x / y / z
      peak location = {peak} in x / y / z
      direction is Positive / Negative / Both in x / y / z
      type is donor / acceptor
    end
    ...
  end
```

Where the *{i}* in Function{i} is replaced with an integer starting with 1 and going to the number of doping functions in the specific material block. If a spatial range specification isn't made for a particular direction then the doping will be applied over the entire material block in which it is specified.

167

16.7.4. Modified Gaussian Doping

You can use *MGauss* (modified Gaussian) doping for a doping profile that varies as a Gaussian function within the specified material with the specified parameters. In Charon the form of the MGauss function, assuming a variation in the *x* direction, is:

$$
N = \begin{cases}
\exp\left[-\frac{\{N_{\max}\}}{\{N_{\min}\}}\left(\frac{x-\{min\}}{\{width\}}\right)^2\right] & x < \{min\}, N_{\min} > 0 \text{ and } \textit{erfc} \text{ is not set} \\[2ex]
\exp\left[-\left(\frac{x-\{min\}}{\{width\}}\right)^2\right] & x < \{min\}, N_{\min} \leq 0 \text{ and } \textit{erfc} \text{ is not set} \\[2ex]
\exp\left[-\frac{\{N_{\max}\}}{\{N_{\min}\}}\left(\frac{x-\{max\}}{\{width\}}\right)^2\right] & x > \{max\}, N_{\min} > 0 \text{ and } \textit{erfc} \text{ is not set} \\[2ex]
\exp\left[-\left(\frac{x-\{max\}}{\{width\}}\right)^2\right] & x > \{max\}, N_{\min} \leq 0 \text{ and } \textit{erfc} \text{ is not set} \\[2ex]
\frac{1}{2}\left[\text{erfc}\left(\frac{x-\{max\}}{\{width\}}\right) - \text{erfc}\left(\frac{x-\{min\}}{\{width\}}\right)\right] & \text{if } \textit{erfc} \text{ is set}
\end{cases}
\tag{179}
$$

The syntax for MGauss doping is

```
start material block {materialBlockName}
  ...
  start mgauss doping named Function{i}
    concentration range is {N_min} to {N_max} in x / y / z
    spatial range is {min} to {max} in x / y / z
    gradient width = {width} in x / y / z
    direction is Positive / Negative / Both in x / y / z
    erfc in x / y / z
    type is donor / acceptor
  end
  ...
end
```

Where the *{i}* in `Function{i}` is replaced with an integer starting with 1 and going to the number of doping functions in the specific material block. If a `spatial range` specification isn't made for a particular direction then the doping will be applied over the entire material block in which it is specified.

16.7.5. Doping from File

Besides specifying analytic doping profiles, Charon can also use doping from doping files. A doping file is a text file that contains 1D, 2D, or 3D profile. 1D doping file contains two-column data: the first column is the coodinate value in the unit of μm, and the second column is the doping value in the unit of cm^{-3}. 2D doping file contains three-columnn data: the first column is the x-coordinate value, the second column is the y-coordinate value, and the third column is the doping value. 3D doping file contains four-columnn data: the first column is the x-coordinate value, the second column is the y-coordinate value, the third column is the z-coordinate value, and the fourth column is the doping value. Doping from a 1D doping file is linearly interpolated

to a mesh grid in Charon simulation. Doping from a 2D/3D doping file is mapped to a mesh grid using the nearest-neighbor values.

2D and 3D doping files can be specified using the following line syntax:

```
start material block {materialBlockName}
  ...
  read doping from {filename} for acceptor / donor in 2d / 3d with buffer {bufferValue}
  ...
end
```

where `buffer` is to increase the doping domain extracted from a 2D or 3D doping file by an amount of `bufferValue` in x, y (for 2D), and z (for 3D) directions.

The syntax for specifying 1D doping file is

```
start material block {materialBlockName}
  ...
  start file1d doping named Function{i}
    file is {filename}
    type is donor / acceptor
    direction is x / y / z
    buffer = {value}
    spatial range is {min} to {max} in x / y / z
    gauss decay with start position at {pos} and decay width at {width} in +x / -x / +y
        / -y / +z / -z
  end
  ...
end
```

Here `direction` specifies the axis along which the 1D doping from a file is applied, and `buffer` is to expand the doping domain by `value` along the `direction` axis. `spatial range` is to specify the domain where a 1D doping is applied. Along the `direction` axis, the maximum and minimum coordinate values are extracted from the 1D doping file, and are compared to the `min` and `max` values given in the `spatial range`; the smaller window of the two cases is used for placing the doping. Along other axises (i.e., not the `direction` axis), a Gaussian decay can be specified using the `gauss decay` line optionally. The Gaussian decay factor is defined as $exp(-(x-x_0)^2/(2\sigma^2))$ with x_0 given by `pos` and σ given by `width`. The doping along the Gaussian decay axis is the doping from a 1D doping file multiplied by the Gaussian decay factor.

An example of specifying Gaussian decays for doping read from 1D doping file and a uniform doping is given at `tcad-charon/test/nightlyTests/capability/pndiode.nlp.gauss-decay.inp`. In addition, multiple 1D and 2D doping files can be read in a single Charon simulation. An example of this is located at `tcad-charon/test/nightlyTests/capability/pndiode.nlp.multi-files.inp`.

169

17. CHARON ANALYSIS WITH DAKOTA (BETA FEATURE)

A beta version of scripts is included with this version of Charon that facilitates the use of the Dakota [47] tool with Charon. This addition to the Charon suite is presently in a very early stage and robustness is not guaranteed. Later versions will be more complete. This section describes these python scripts and how to use them, but aside from very simple examples, does not describe the use of Dakota. The user is referred to Dakota documentation for that instruction. Moreover, the Dakota application code must be separately installed. The user is referred to Dakota documentation for instructions to do the installation.

Dakota is used in conjunction with Charon simulations through driver scripts. Details of how to use those scripts is described later in this section. The basic work flow, with Dakota as the primary application driver is shown graphically in Figure 17-1.
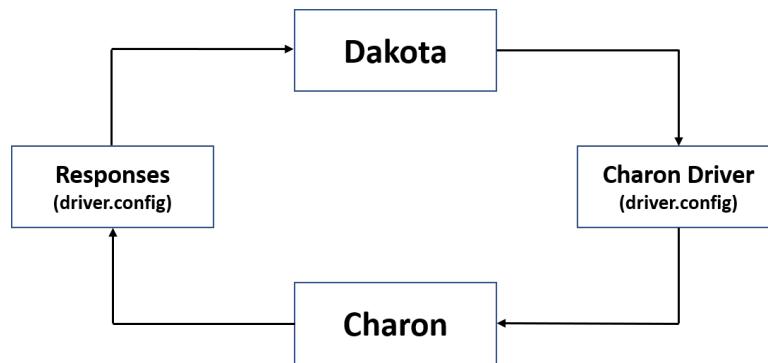

Figure 17-1 Diagram of the work flow of a Charon simulation with Dakota.

Dakota will execute the Charon simulations with Dakota controlled parameters through the Charon driver as configured by a simulation specific `driver.config` file. Also defined in the `driver.config` file are specifics of the responses that Dakota will expect back for its continued analysis. Details of how to write the `driver.config` file will be described in a later section.

17.1. Building with Dakota Support

In order to use the Charon driver scripts with Dakota, they must be built with Charon. By including the following line in the opts file (see Section 2)

```
tcad-charon_ENABLE_DAKOTA_DRIVERS:BOOL=ON
```

the relevant scripts will be compiled and installed with the Charon installation.

The main driver script, `charonDriver.py`, will be installed in the `<charonInstall>/bin` directory where the Charon executable is likewise installed. This script will be called by Dakota to execute Charon or may be called on the command line to get a limited amount of help with its use. The Charon driver is described in Section 17.2.

Selected response modules will also be installed in the `bin` directory along with the Charon driver and Charon executable. Response modules will be described in Section 17.3. Remaining support scripts will be installed in an adjacent location which the primary driver can access.

17.2.　　The Charon Driver and its Configuration for Dakota

The Charon driver, `charonDriver.py` is the main script that executes Charon for Dakota according to a simulation specific configuration file that will be created by the user. As will be seen in Section 17.4, `charonDriver.py` will need to be specified in a configuration for Dakota for the driver to be used.

The Charon driver may be executed directly from the command line in order to obtain some help in configuring a Dakota run. Note that it will always expect at least one argument. The scripts have a set of responses that are available to use with Dakota. Any Dakota simulation will always expect at least one response. To see the list of available responses, execute:

```
% charonDriver.py --listresponses

Response  0  is  thresholdVoltage
Response  1  is  currentAtVoltage
Response  2  is  betaGain
Response  3  is  IVCurve
Response  4  is  compositeFunction
```

An enumerated list of available responses will be produced. For assistance in configuring any one response, help is available through the driver. For example, one response is `thresholdVoltage`. To obtain configuration help for `thresholdVoltage`, execute:

```
% charonDriver.py --help thresholdVoltage

reponse thresholdVoltage argument1=value1 argument2=value2...

This response returns the threshold voltage.

method -- <slope>,<current> If the method is set to slope, the
function will calculate the slope of the IV curve and use it's maximum
to interpolate the threshold voltage.  If method is not specified or
is specified as current, a current must be specified.


current -- Set a numerical value of the contact current at which the
threshold voltage is calculated.  This is an alternative to the slope
method.
```

```
filename -- Name of the file where the currents and voltages are
expect to be found.



 target -- The target is used for calibration.  It is the expected
 value of the threshold voltage.  When the target is specified, a
 residual of the expected and calculated values is returned.



 icap -- Numerical cap on the current at which the slope method can be
 used to calculate the threshold voltage



weighted -- When the residual is calculated with a target value,
setting this to yes or no will opt into weighting the residual by the
expected value.  The default is to weight the residual.



responsename -- The default name of the response is thresholdVoltage.
However, if differnt threshold voltages are to be calculated such as
weighted and unweighted or at multiple current values, a unique name
must be given to each reponse.
```

This assistance is available for all the responses. Details of how this configuration assistance is to be used will be described in 17.4.

When Dakota calls on Charon to produce responses for its analysis, the `driver.config` file will configure the driver to execute Charon in specific ways. Indeed, it can be configured to run Charon multiple times if needed to produce a single more multiple responses. The configure script has three sections: executeMethods, executeProcs and response. An example configuration file is:

```
executeMethods  app=charon  template=mosfet.nlp.template  options=decompose
executeMethods  app=charon  template=mosfet.drain.template
executeMethods  app=charon  template=mosfet.gate.template

executeProcs    64

response  thresholdVoltage data=single filename=gateSweep.dat\
                                    current=2E-5 responsename=tV1
response  currentAtVoltage filename=gateSweep.dat voltage = 1.3
```

In this configuration, each time Dakota invokes the driver to obtain responses from Charon, the driver will execute Charon three times to produce the required data. In this instance, the driver

will run a nonlinear Poisson initial simulation with the option to do a domain decomposition of the state file, then a drain sweep and finally a gate sweep. Each instance will be executed on 64 processors as per the `executeProcs` line. Note that any options available to the interpreter may be specified in the `options=` section of the `executeMethods` line. The `response` line will be described in the following section.

The application line, `app=`, of the `executeMethods` line will call out the application to be executed by the driver. This will almost always be `charon`, but `cubit` can also be executed for users with access to the cubit mesh generation tool. The `template=` section of the `executeMethods` will be the main Charon input file. It must always be suffixed as `.template` for the driver to process it. Any other input files specified in the main Charon input file through `/include` statements should be left unchanged. The driver will initially consolidate the template file plus any included files into a single input file and the suffix modified to `.inp`. Special variables that Dakota will modify for specifying parameters for a specific simulation will need to be specified in the template and include files. A simple example how this is done is described in Section 17.4.

17.3.   Charon Response Modules

In the present version, a specific set of responses is available through the driver to provide data to Dakota for analysis. In future version, a facility will be included for user-defined scripts to be executed so that customized responses may be created. Because this feature is rapidly evolving, the reader is directed to use the help features of the driver to obtain help on the configuration of specific responses.

The responses that will be calculated are called out in the `response` section of the `driver.config` file. The configuration presented in the previous section directs that two responses be computed: `thresholdVoltage` and `currentAtVoltage`. Dakota will expect and request responses with specific names which are specified in the Dakota input file. The Charon driver will name these responses. By default, the name of the computed response will be identical to the name of the response itself. For example, in the previous section, the computed response for the `currentAtVoltage` response will likewise be named `currentAtVoltage`. However, a custom name may be given to any computed response with the `responsename=` option on the `response` line. In the previous example, the `thresholdVoltage` response will be named `tv1`.

In some cases, it may be required for custom names to be specified. For example, if the current at two gate voltages is required, the configuration may be

```
response   thresholdVoltage data=single filename=gateSweep.dat\
                                   current=2E-5 responsename=tV1
response   currentAtVoltage filename=gateSweep.dat voltage = 1.3\
                                   responsename=current1
response   currentAtVoltage filename=gateSweep.dat voltage = 0.5\
                                   responsename=current2
```

where three responses are computed for the threshold voltage and the currents at gate voltages of 0.5V and 1.3V. The three response names that will be specified in the Dakota input file will be `'tv1' 'current1' 'current2'`.

In the present version, two of the responses may be called on manually from the command line. These are independent of a Dakota simulation and may be used for any Charon simulation of the appropriate type. The two responses are `thresholdVoltage` and `currentAtVoltage`. The syntax of calling on a response from the command line differs from that in the configuration file for the Charon driver. Specific syntax and options may be got by calling the response with the `--help` argument. For example,

```
% thresholdVoltage --help
usage: thresholdVoltage [-h] [-m METHOD] [-c CURRENT] [-f FILENAME]
                        [--icap ICAP]

optional arguments:
  -h, --help            show this help message and exit
  -m METHOD, --method METHOD
                        method to use to compute Vt (slope, current)
  -c CURRENT, --current CURRENT
                        current at which to compute Vt if method is current.
  -f FILENAME, --filename FILENAME
                        File name that holds I-V data to compute Vt.
  --icap ICAP           If method is slope, it will limit the evaluation to a
                        current no higher than specified
```

For example, to compute threshold voltage, V_t of a MOSFET with the threshold defined to be at $33\mu A/cm$ from a Charon generated I-V file named `gateSweep.dat`,

```
% thresholdVoltage -m current -f gateSweep.dat -c 3.3e-5

------------------------------------------------------------------
Response:    thresholdVoltage
------------------------------------------------------------------
value:                       0.6204534252298233
target:                      N/A
------------------------------------------------------------------

Vt =  0.6204534252298233  at  3.3e-05  amps
```

17.4.   A Simple Dakota/Charon Example

A simple example in which Dakota is called on to sample a MOSFET simulation with different values of the work function of the gate contact is provided here. There are many ways in which a

Dakota-driven simulation may be organized, but a recommended organization is presented in this section. A root directory, say `MOSFETSimulation` is created for the simulation and its results. In that directory is place a single file that configures the Dakota run. That file may be `dakotaSamples.in`, for example,

```
environment
  tabular_data
    tabular_data_file = 'lhs.dat'

method
  sampling
    sample_type lhs
    samples = 20
    seed = 1337

variables
  normal_uncertain = 2
    means         = 4.25
    std_deviations = 0.5
    descriptors    'workFunction'

interface
  analysis_drivers = 'charonDriver.py'
    fork
    asynchronous
    evaluation_concurrency 1
    link_files = 'template_dir/*'
    work_directory named 'workingDir'
    directory_save

responses
  response_functions = 3
    descriptors      = 'tv1' 'current1' 'current2'
  no_gradients
  no_hessians
```

The reader is referred to Dakota documentation for the details of this and any other Dakota input file. But for the sake of this example, Dakota will produce a distribution of the responses, `'tv1'` `'current1'` `'current2'` for 20 Latin Hypercube samples of the gate work function with a normal distribution with a mean of 4.5 and a standard deviation of 0.5.

Note that the `analysis_drivers` is called out to be `'charonDriver.py'`. This must always be for Charon's driver scripts to be used. The recommended organization for Dakota simulations is to create a `template_dir` directory under the root `MOSFETSimulation` directory that contains all of the

relevant Charon files—that is, all of the `.inp` files, the `.template` files and the mesh files—to include the `driver.config` file.

The `driver.config` located in the `template_dir` directory for this example is

```
executeMethods  app=charon  template=mosfet.nlp.template  options=decompose
executeMethods  app=charon  template=mosfet.drain.template
executeMethods  app=charon  template=mosfet.gate.template

executeProcs    64

response  thresholdVoltage data=single filename=gateSweep.dat\
                                    current=2E-5 responsename=tV1
response  currentAtVoltage filename=gateSweep.dat voltage = 1.3\
                                    responsename=current1
response  currentAtVoltage filename=gateSweep.dat voltage = 0.5\
                                    responsename=current2
```

The response names in `driver.config` *must* match the `descriptors` specified in the `responses` section of the Dakota input file for Dakota and Charon to communicate properly.

In each of the Charon input files, a boundary condition for the gate contact is specified (refer to Section 16 for details). In this case, the boundary condition line should be specified as

```
BC is contact on insulator for <contact> on <block> with \
                            work function {workFunction} <voltage>
```

where `<contact>`, `<block>` and `voltage` are specified as per usual for a Charon simulation. The `workFunction` variable must be bracketed and spelled with case sensitivity just as it is specified in the `descriptors` line in the `variables` section of the Dakota input file. When Dakota calls the Charon driver, it will provide a specific value for the work function. The driver will create the input file appropriately with the specified value before Charon is executed.

When Dakota is executed with the input file shown above, it will call the Charon driver once for each evaluation of responses at a given value of the work function. Each of these evaluations will be executed in a newly created subdirectory under the root called `workingDir.x` where $1 \leq x \leq 20$. Once the Dakota execution has completed, the results will be tabulated in the file `lhs.dat` in the root directory as specified by `tabular_data_file` in the `environment` section of the Dakota input file.

REFERENCES

[1] X. Gao, E. Nielsen, R. P. Muller, R. W. Young, A. G. Salinger, N. C. Bishop, M. P. Lilly, and M. S. Carroll. Quantum computer aided design simulation and optimization of semiconductor quantum dots. *Journal of Applied Physics*, 114(164302), 2013.

[2] D. Schroeder, T. Ostermann, and O. Kalz. Comparison of transport models for the simulation of degenerate semiconductors. *Semiconductor Science and Technology*, 9:364–369, 1994.

[3] M. S. Adler. Accurate calculations of the forward drop and power dissipation in thyristors. *IEEE Transactions on Electron Devices*, ED-25(1):16–22, 1978.

[4] G. K. Wachutka. Rigorous thermodynamic treatment of heat generation and conduction in semiconductor device modeling. *IEEE Transactions on Computer Aided Design*, 9(11):1141–1149, 1990.

[5] U. Lindefelt. Heat generation in semiconductor devices. *Journal of Applied Physics*, 75(942), 1994.

[6] J. E. Parrott. Thermodynamic theory of transport processes in semiconductors. *IEEE Transactions on Electron Devices*, 43(5):809–826, 1996.

[7] D. B. Strukov, J. L. Borghetti, and R. S. Williams. Coupled ionic and electronic transport model of thin-film semiconductor memristive behavior. *Small*, 5(9):1058–1063, 2009.

[8] X. Gao, D. Mamaluy, P. R. Mickel, and M. Marinella. Three-dimensional fully-coupled electrical and thermal transport model of dynamic switching in oxide memristors. *ECS Transactions*, 69(5):183–193, 2015.

[9] X. Gao, D. Mamaluy, E. C. Cyr, and M. J. Marinella. Comprehensive assessment of oxide memristors as post-cmos memory and logical devices. *ECS Transactions*, 72(3):49–58, 2016.

[10] A. N. Brooks and T. J.R. Hughes. Streamline upwind petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32(1-3):199–259, 1982.

[11] P. Bochev, K. Peterson, and X. Gao. A new control volume finite element method for the stable and accurate solution of the drift-diffusion equations on general unstructured grids. *Computer Methods in Applied Mechanics and Engineering*, 254:126–145, 2013.

[12] P. Bochev and K. Peterson. A parameter-free stabilized finite element method for scalar advection-diffusion problems. *Cent. Eur. J. Math*, 11(8):1458–1477, 2013.

[13] H.A. Bethe and Massachusetts Institute of Technology. Radiation Laboratory. *Theory of the Boundary Layer of Crystal Rectifiers*. Report (Massachusetts Institute of Technology. Radiation Laboratory). Radiation Laboratory, Massachusetts Institute of Technology, 1942.

[14] R. Tsu and L. Esaki. Tunneling in a finite superlattice. *Applied Physics Letters*, 22(11):562–564, 1973.

[15] J. W. Slotboom. The *pn*-product in silicon. *Solid-State Electronics*, 20:279–283, 1977.

[16] D.B.M. Klaassen, J.W. Slotboom, and H.C. [de Graaff]. Unified apparent bandgap narrowing in n- and p-type silicon. *Solid-State Electronics*, 35(2):125–129, 1992.

[17] E.S. Harmon, M.R. Melloch, and M.S. Lundstrom. Effective bandgap shrinkage in gaas. *Applied Physics Letters*, 64(4):502–504, January 1994.

[18] C. Persson, U. Lindefelt, and B. E. Sernelius. Band gap narrowing in n-type and p-type 3c-, 2h-, 4h-, 6h-sic, and si. *Journal of Applied Physics*, 86(8):4419–4427, 1999.

[19] N.D. Arora, J.R. Hauser, and D.J. Roulston. Electron and hole mobilities in silicon as a function of concentration and temperature. *IEEE Trans. Electron Devices*, 29:292–295, 1982.

[20] J.D. Albrecht et al. Electron transport characteristics of GaN for high temperature device modeling. *Journal of Applied Physics*, 83(9):4777–4781, May 1998.

[21] M. Farahmand et al. Monte Carlo Simulation of Electron Transport in the III-Nitride Wurtzite Phase Materials System: Binaries and Ternaries. *IEEE Transactions on Electron Devices*, 48(3):535–542, March 2001.

[22] D. B. M. Klaassen. A unified mobility model for device simulation i. *Solid-State Electronics*, pages 953–959, 1992.

[23] D. B. M. Klaassen. A unified mobility model for device simulation ii. *Solid-State Electronics*, pages 961–967, 1992.

[24] D.M. Caughey and R.E. Thomas. Carrier mobilities in silicon empirically related to doping and field. *Proceedings of the IEEE*, 55(12):2192–2193, December 1967.

[25] S.M. Sze and Kwok K. Ng. *Physics of Semiconductor Devices*. John Wiley and Sons, Inc., Hoboken, New Jersey, 2007.

[26] A. Schenk. A model for the field and temperature dependence of shockley-read-hall lifetimes in silicon. *Solid-State Electronics*, 35:1585–1596, 1992.

[27] X. Gao, B. Kerr, and A. Huang. Analytic band-to-trap tunneling model including band offset for heterojunction devices. *Journal of Applied Physics*, 125, 054503, 2018.

[28] C. R. Crowell and S. M. Sze. Temperature dependence of avalanche multiplication in semiconductors. *Applied Physics Letters*, 9(6):242–244, 1966.

[29] G. A. Baraff. Distribution functions and ionization rates for hot electrons in semiconductors. *Phys. Rev.*, 128:2507–2517, Dec 1962.

[30] E.O. Kane. Theory of tunneling. *Journal of Applied Physics*, 32(1):83–91, 1961.

[31] E.O. Kane. Zener tunneling in semiconductors. *Journal of Physics and Chemistry of Solids*, 12:181–188, 1959.

[32] G.A.M. Hurkx, D.B.M. Klaassen, and M.P.G. Knuvers. A new recombination model for device simulation including tunneling. *IEEE Transactions on Electron Devices*, 39(2):331–338, 1992.

[33] A. Schenk. Rigorous theory and simplified model of the band-to-band tunneling in silicon. *Solid-State Electronics*, 36(1):19–34, 1993.

[34] H.Y. Wong, D. Dolgos, L. Smith, and R.V. Mickevicius. Modified hurkx band-to-band-tunneling model for accurate and robust tcad simulations. *Microelectronics Reliability*, 104(113552):1–4, 2020.

[35] A. Wettstein, A. Schenk, and W. Fichtner. Quantum device-simulation with the density-gradient model on unstructured grids. *IEEE Transactions on Electron Devices*, 48(2):279–284, 2001.

[36] C. M. Wu and E. S. Yang. Carrier transport across heterojunction interfaces. *Solid-State Electronics*, 22:241–248, 1979.

[37] K. Horio and H. Yanai. Numerical modeling of heterojunctions including the thermionic emission mechanism at the heterojunction interface. *IEEE Transactions on Electron Devices*, 37(4):1093–1098, 1990.

[38] K. Yang, J. R. East, and G. I. Haddad. Numerical modeling of abrupt heterojunctions using a thermionic-field emission boundary condition. *Solid-State Electronics*, 36(3):321–330, 1993.

[39] D. Bednarczyk and J. Bednarczyk. The approximation of the fermi-dirac integral f(1/2)(eta). *Physics Letters*, 64A(4):409–410, January 1978.

[40] N.G. Nilsson. An accurate approximation of the generalized einstein relation for degenerate semiconductors. *Physica Status Solidi (a)*, pages K75–K78, 1973.

[41] Thomas I. Seidman. Time-dependent solutions of a nonlinear system arising in semiconductor theorem, ii: Boundedness and periodicity. *IMA Preprint Series 80*, July 1984.

[42] B. Troyanovsky. Frequency domain algorithms for simulating large signal distortion in semiconductor devices. *Ph.D. Thesis*, Nov. 1997.

[43] A. Kurganov and J. Rauch. The order of accuracy of quadrature formulae for periodic functions. *Advances in Phase Space Analysis of Partial Differential Equations: In Honor of Ferruccio Colombini's 60th Birthday*, 2009.

[44] J. C. Pedro and N. B. Carvalho. Efficient harmonic balance computation of microwave circuitsâĂŹ response to multi-tone spectra. *29th European Microwave Conference Proc.*, vol. I, Oct. 1999.

[45] S.E. Laux. Techniques for small-signal analysis of semiconductor devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 4(4):472–481, 1985.

[46] William R. Wampler and Samual M. Myers. Model for transport and reaction of defects and carriers within displacement cascades in gallium arsenide. *Journal of Applied Physics*, 117(045707), 2015.

[47] Dakota. Internet URL https://dakota.sandia.gov.

[48] X. Gao, A. Huang, and B. Kerr. Efficient band-to-trap tunneling model including heterojunction band offset. *ECS Transactions*, 80:1005–1015, 2017.

[49] X. Gao, B. Kerr, A. Huang, G. Hennigan, L. Musson, and M. Negoita. Analytic band-to-trap tunneling model including electric field and band offset enhancement. *Proceedings of International Conference on Simulation of Semiconductor Processes and Devices 2018*, 2018.

APPENDIX A. HISTORICAL PERSPECTIVE

APPENDIX B. BAND-TO-TRAP TUNNELING MODELS

In this appendix, the supported band-to-trap tunneling models are described in some level of details. Charon supports four variation forms of the Schenk band-to-trap tunneling model [26] and also a new model designed for heterojunction devices [27].

The first form is the Schenk model in the high temperature limit, where the electron field enhancement factor is given by

$$
\begin{aligned}
g_n(T,F) \;=\; & \left(1 + \frac{2E_R k_B T}{(\hbar\Theta)^{3/2}\sqrt{E_t - E_0}}\right)^{-1/2} \frac{E_{act}^0 + E_t}{k_B T}\left(\frac{\hbar\Theta}{E_t + E_R}\right)^{3/2} \\
& \times \; \exp\left[\frac{E_{act}^0 - E_{act}(F)}{k_B T}\right] \exp\left(\frac{E_t - E_0}{k_B T}\right) \exp\left[-\frac{4}{3}\left(\frac{E_t - E_0}{\hbar\Theta}\right)^{3/2}\right].
\end{aligned}
\tag{180}
$$

$E_R = S\hbar\omega_0$, the lattice relaxation energy, with S being the Huang-Rhys factor and $\hbar\omega_0$ being the effective optical phonon energy. E_0 is the optimum transition energy and given by

$$
E_0 = 2\sqrt{E_F}\left[\sqrt{E_F + E_t + E_R} - \sqrt{E_F}\right] - E_R, \;\; E_F = \frac{(2E_R k_B T)^2}{(\hbar\Theta)^3}, \;\; \hbar\Theta = \left(\frac{\hbar^2 q^2 F^2}{2m_n^*}\right)^{1/3}.
\tag{181}
$$

Here q is the elemental charge, F is the local electric field, m_n^* is the electron effective mass. $\hbar\Theta$ is known as the electrooptical energy. $E_{act}^0 = (E_t - E_R)^2/(4E_R)$, the activation energy for capturing an electron from the conduction band edge. $E_{act} = (E_0 - E_R)^2/(4E_R)$, the field-dependent activation energy. The first model is enabled by specifying `model is hightemp approx` in the trap srh block.

The second form is the Schenk model in the low temperature limit, where the electron field enhancement factor is given by

$$
\begin{aligned}
g_n(T,F) \;=\; & \left[1 + \frac{(\hbar\Theta)^{3/2}\sqrt{(E_t - E_0)}}{E_0 \hbar\omega_0}\right]^{-1/2} \frac{(\hbar\Theta)^{3/4}(E_t - E_0)^{1/4}}{2\sqrt{E_t E_0}}\left(\frac{\hbar\Theta}{k_B T}\right)^{3/2} \\
& \times \; \exp\left[-\frac{E_t - E_0}{\hbar\omega_0} + \frac{\hbar\omega_0 - k_B T}{2\hbar\omega_0} + \frac{E_t + 0.5 k_B T}{\hbar\omega_0}\ln\left(\frac{E_t}{E_R}\right) - \frac{E_0}{\hbar\omega_0}\ln\left(\frac{E_0}{E_R}\right)\right] \\
& \times \; \exp\left(\frac{E_t - E_0}{k_B T}\right) \exp\left[-\frac{4}{3}\left(\frac{E_t - E_0}{\hbar\Theta}\right)^{3/2}\right].
\end{aligned}
\tag{182}
$$

The E_R, E_0, and $\hbar\Theta$ parameters have the same definitions as in Eq. (180). The second model is enabled by specifying `model is lowtemp approx` in the trap srh block.

The above two g_n expressions were obtained from the Schenk model in the high and low temperature limits, respectively. Without any temperature approximation, it can be shown [27] that the general g_n formula from the Schenk model takes the form of

$$g_n(T,F) = \frac{\int_0^{E_t} \rho_n(E) I_{E/\hbar\omega_0}(z) \exp\left(\frac{-E}{2k_BT}\right) dE}{\int_{E_t}^{\infty} \rho_n^{F=0}(E) I_{E/\hbar\omega_0}(z) \exp\left(\frac{-E}{2k_BT}\right) dE}. \tag{183}$$

The modified Bessel function for large orders (i.e., $l = E/\hbar\omega_0 \gg 1$, which generally holds for deep-level traps that are relevant for band-to-trap tunneling) is equal to

$$I_l(z) = \frac{1}{\sqrt{2\pi}} \frac{1}{(l^2+z^2)^{1/4}} \exp\left(\sqrt{l^2+z^2}\right) \exp\left[-l\ln\left(\frac{l}{z} + \sqrt{1+\frac{l^2}{z^2}}\right)\right], \tag{184}$$

with

$$z = 2S\sqrt{f_B(f_B+1)}, \quad f_B = \left[\exp\left(\frac{\hbar\omega_0}{k_BT}\right) - 1\right]^{-1}. \tag{185}$$

f_B is the Bose distribution function. The zero-field density of states (DOS), $\rho_n^{F=0}(E)$, is given by

$$\rho_n^{F=0}(E) = \frac{1}{2\pi^2} \left(\frac{2m_n^*}{\hbar^2}\right)^{3/2} \sqrt{E-E_t}. \tag{186}$$

The field dependent density of states, $\rho_n(E)$, in the Schenk model was derived based on the constant field assumption. It is given by

$$\rho_n(E) = \frac{1}{2\pi} \left(\frac{2m_n^*}{\hbar^2}\right)^{3/2} \sqrt{\hbar\Theta} \mathcal{F}\left(\frac{E_t-E}{\hbar\Theta}\right), \quad \mathcal{F}(y) = A_i'^2(y) - yA_i^2(y). \tag{187}$$

$\mathcal{F}(y)$ is known as the electrooptical function. $A_i(\cdot)$ is the Airy function of the first kind. $A_i'(\cdot)$ is the first derivative of the Airy function. For large positive arguments, the electrooptical function $\mathcal{F}$ takes the asymptotic form of

$$\mathcal{F}\left(\frac{E_t-E}{\hbar\Theta}\right) = \frac{1}{8\pi} \frac{\hbar\Theta}{E_t-E} \exp\left[-\frac{4}{3}\left(\frac{E_t-E}{\hbar\Theta}\right)^{3/2}\right]. \tag{188}$$

To use the general Schenk model in Eq. (183) with the constant field DOS in Eq. (187), it is required to specify `model is constant field` in the trap srh block. To use the Schenk model with the asymptotic DOS in Eq. (188), we need to replace `constant field` with `asymptotic field` in the input file.

The four Schenk models described above can be used for modeling band-to-trap tunneling in homojunction devices under different conditions. They provide different accuracies and require different computation time. That is, the `hightemp approx` and `lowtemp approx` models are faster than `asymptotic field`, which is faster than `constant field`, in terms of calculation time. However,

the `constant field` model has the least number of assumptions, so it is the most accurate among the four.

Figure 1(a) shows the electron field enhancement factor for the gold traps in silicon at 300 K. The legend texts in the following figures have the following correspondence: *Schenk HighTemp* corresponds to `hightemp approx`, *Schenk LowTemp* corresponds to `lowtemp approx`, *Schenk AsymConstFDOS* corresponds to `asymptotic field`, *Schenk ConstFDOS* corresponds to `constant field`, and *Schenk NewDOS* corresponds to `new`. The parameters used for the calculation are $\hbar\omega_0 = 0.068$ eV, $S = 3.5$, $E_t = 0.55$ eV, $m_n^* = \sqrt{0.92 \times 0.19} = 0.42$, and $T = 300$ K. For the gold traps in silicon, we can make three observations: (i) the *Schenk LowTemp* and *Schenk AsymConstFDOS* models produce the same electron enhancement factors for the whole range of electric fields; (ii) the results from the two models are very close to the more accurate *Schenk ConstFDOS* model, except at the low field regime where the field factors are small anyway; (iii) the results from *Schenk HighTemp* model show somewhat larger errors when compared to other three models. Figure 1(b) shows the electron field enhancement factor for the E5 traps in GaAs at 300 K. The parameters used for this calculation are $\hbar\omega_0 = 0.02$ eV, $S = 12.2$, $E_t = 0.66$ eV, $m_n^* = 0.067$, and $T = 300$ K. For the E5 traps in GaAs at 300 K, we can see that the *Schenk LowTemp* model is broken down, while other three Schenk models produce good results.


(a) Gold traps in Si   **(b)** E5 traps in GaAs
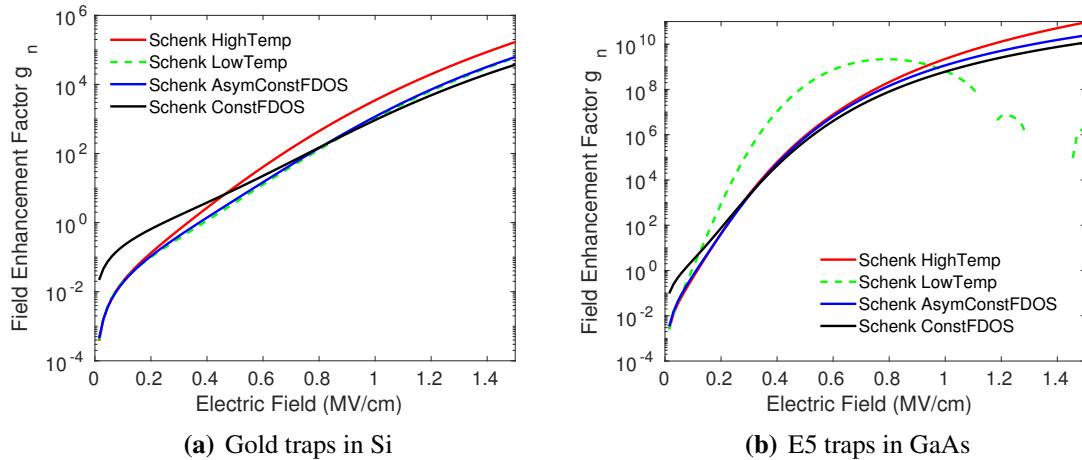
Figure B-1 Electron field enhancement factor for traps at 300 K as a function of electric field computed using the four Schenk models.

The Schenk model with the constant field DOS and its approximated versions were derived for silicon devices. Therefore, the models may be adequate for modeling band-to-trap tunneling in homojunction devices. However, they are unsuitable for modeling heterojunction structures, since they do not capture the effect of heterojunction band offset. This effect can be very strong near the heterojunction. As seen from the typical band profile of an $In_{0.5}Ga_{0.5}P/GaAs$ NP^+N heterojunction bipolar transistor (HBT) in figure B-2, the tunneling of holes in the base to traps in the emitter is determined by the valence band profile. And it is evident that the valence band deviates significantly from the constant field assumption due to the large band offset.

To address the limitation of the constant field assumption, we have developed an analytic DOS model [48, 49, 27] that includes both the electric field and band offset effects. The new model
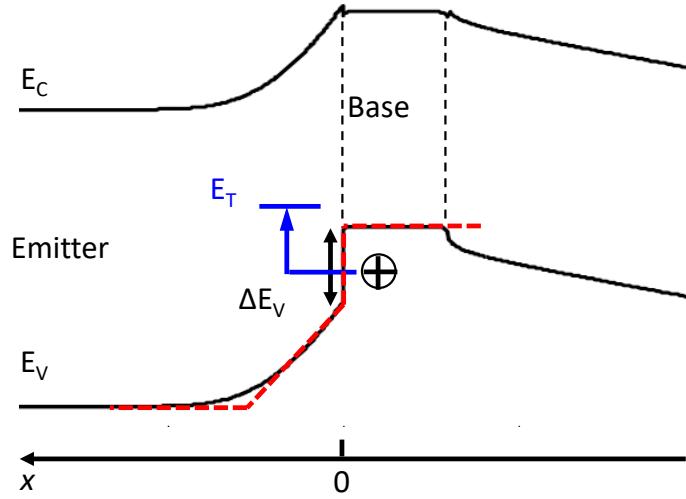
Figure B-2 Typical band profile in the emitter and base regions of an In$_{0.5}$Ga$_{0.5}$P/GaAs NP$^+$N HBT. Here E$_C$ is the conduction band, E$_V$ is the valence band, ΔE$_V$ is the valence band offset, and E$_T$ indicates the trap location. The circle with a plus represents a hole, and the blue arrow denotes the hole-to-trap tunneling path. The red dashed curve is a linearized potential to approximate the actual valence band.

allows us to accurately simulate the band-to-trap tunneling near a heterojunction. For the typical valence band in figure B-2, the actual potential profile is first linearized with the band offset included, as shown by the red dashed curve. Then the DOS for the linearized potential is given by

$$\rho_n(x,E) = \frac{1}{2\pi^2}\left(\frac{2m_n^*}{\hbar^2}\right)^{\frac{3}{2}}\int_0^{E^*}\frac{|A_i(\alpha x + \beta)|^2}{|A_i(\beta)|^2 + \frac{\hbar\Theta}{E_x}|A_i'(\beta)|^2}\frac{dE_x}{\sqrt{E_x}}.\tag{189}$$

$E^* = qFx + V_0 - E_t + E$, where q is the elemental charge, F is the local electric field in the emitter side, x is the positive distance (in the emitter side) from the heterojunction, V_0 is the band offset (e.g., ΔE$_V$ in figure B-2), and $0 \leq E \leq E_t$. $A_i(\cdot)$ is the Airy function of the first kind, and $A_i'(\cdot)$ is the first derivative of the Airy function. α, β, and $\hbar\Theta$, are given by

$$\alpha = \left(\frac{2m_n^*qF}{\hbar^2}\right)^{\frac{1}{3}}, \quad \beta = \frac{V_0 - E_x}{\hbar\Theta}, \quad \hbar\Theta = \left(\frac{\hbar^2q^2F^2}{2m_n^*}\right)^{\frac{1}{3}}.\tag{190}$$

For the approximated valence potential in figure B-2 (red dashed curve), the *Schenk NewDOS* model was used to compute the electron enhancement factor at different locations (in the emitter side) from the heterojunction. The results are plotted in figure B-3 and also compared to those computed using the *Schenk ConstFDOS* model. The parameters used for the calculation are the same as those for the E5 traps in GaAs. At 15 and 20 nm locations, the *Schenk NewDOS* model produces nearly the same results as the *Schenk ConstFDOS* model. However, at 5 and 10 nm locations, the *Schenk NewDOS* model predicts much larger field factors than the *Schenk ConstFDOS* model because of the strong band offset enhancement.
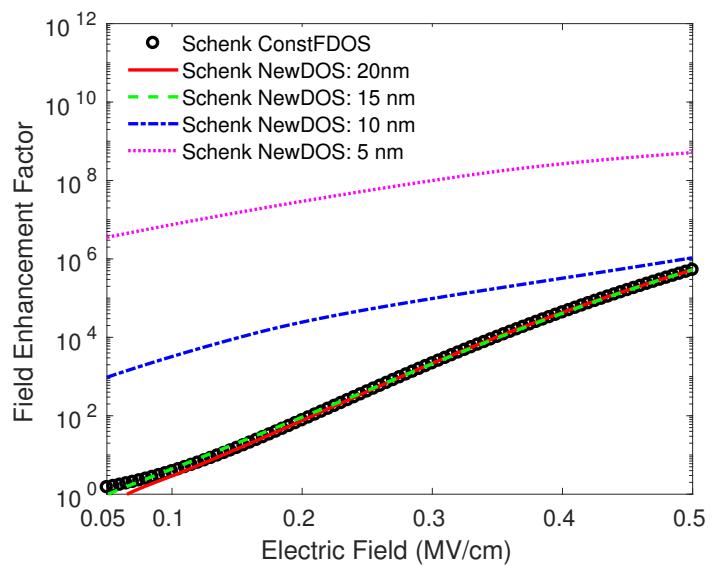
Figure B-3 Electron field enhancement factor for the approximated potential in figure B-2 at 300 K as a function of electric field computed using the *Schenk ConstFDOS* and *Schenk NewDOS* models. The calculations were done for four different locations that are 5, 10, 15, and 20 nm in the emitter away from the heterojunction. The trap parameters are the same as those for the E5 traps in GaAs.

APPENDIX C. DERIVATION OF HETEROJUNCTION MODELS

In this appendix the thermionic emission (TE) and local tunneling (LT) models given in Sec. 12 are derived in detail for electrons. The same derivation is valid for holes.
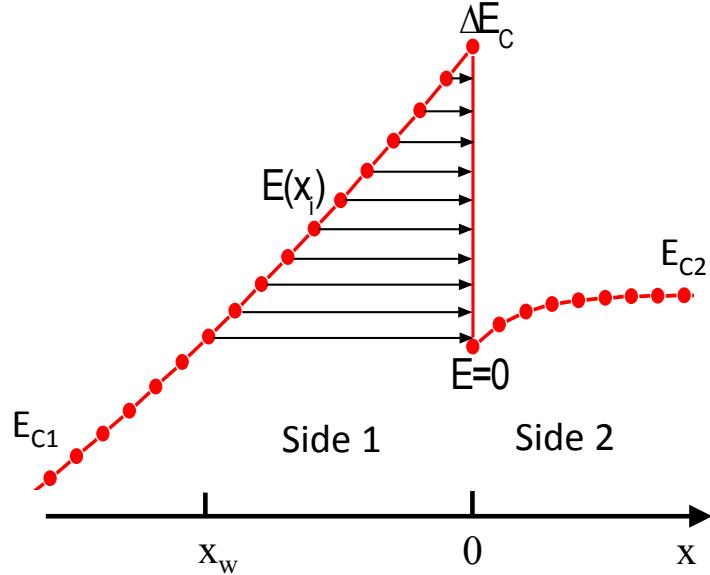

Figure C-1 Example of conduction band diagram illustrating the carrier transport across a heterojunction.

The transport of carriers from one side of a heterojunction to the other is characterized by the quantum mechanical transmission coefficient. According to [36], the electron current per unit area from side 1 to side 2 of the heterojunction in the x-direction illustrated in Fig. C-1 is given by

$$J_{1->2} = \frac{-2q}{V} \sum_{\mathbf{k}} v_x T(E, k_y, k_z) f_1(E)[1 - f_2(E)],$$

where 2 is due to spin degeneracy, q is the elemental charge, V is a normalization volume, v_x is the carrier velocity in the x-direction, and $\mathbf{k}$ is the three-dimensional wave vector which contains three components of k_x, k_y, and k_z. E is the kinetic energy of the carrier and given by (assuming isotropic electron effective mass and parabolic conduction band dispersion)

$$E = \frac{\hbar^2}{2m_n^*} |\mathbf{k}|^2 = \frac{\hbar^2}{2m_n^*} \left(k_x^2 + k_y^2 + k_z^2 \right).$$

In addition, $T(E, k_y, k_z)$ is the quantum mechanical transmission coefficient of the carrier across the junction, $f_1(E)$ and $f_2(E)$ are the probability of the carrier occupancy at the energy E in side 1 and side 2, respectively.

The net electron current across the heterojunction is the difference of the two current components flowing in opposite directions, i.e.,

$$
\begin{aligned}
J_n &= J_{1->2} - J_{2->1} \\
&= \frac{-2q}{V} \sum_{\mathbf{k}} v_x T(E, k_y, k_z) f_1(E)[1 - f_2(E)] + \frac{2q}{V} \sum_{\mathbf{k}} v_x T(E, k_y, k_z) f_2(E)[1 - f_1(E)] \\
&= \frac{-2q}{V} \sum_{\mathbf{k}} v_x T(E, k_y, k_z)[f_1(E) - f_2(E)] \\
&= \frac{-2q}{V} \frac{V}{(2\pi)^3} \int d^3 \mathbf{k} \, v_x T(E, k_y, k_z)[f_1(E) - f_2(E)] \quad \text{use the sum-to-integral rule} \\
&= \frac{-2q}{V} \frac{V}{(2\pi)^3} \int \int \int dk_x dk_y dk_z \left(\frac{\hbar k_x}{m_n^*}\right) T(E, k_y, k_z)[f_1(E) - f_2(E)]
\end{aligned}
$$

The last equality uses the relation that $v_x = p_x/m_n^* = \hbar k_x/m_n^*$. To perform the integration over the wave vector components we rewrite the integration over the carrier kinetic energy by noting that

$$
E = E_{||} + E_x, \quad \text{where } E_{||} = \frac{\hbar^2}{2m_n^*} k_{||}^2 = \frac{\hbar^2}{2m_n^*} \left(k_y^2 + k_z^2\right), \quad E_x = \frac{\hbar^2}{2m_n^*} k_x^2.
$$

From the above, the derivatives of the wave vector components can be expressed as

$$
dk_x = \frac{m_n^*}{\hbar^2 k_x} dE_x, \quad dk_y dk_z = d^2 k_{||} = 2\pi k_{||} dk_{||} = \frac{2\pi m_n^*}{\hbar^2} dE_{||}.
$$

Then J_n becomes

$$
\begin{aligned}
J_n &= \frac{-2q}{(2\pi)^3} \frac{1}{\hbar} \frac{2\pi m_n^*}{\hbar^2} \int dE_{||} \int dE_x T(E_{||}, E_x)[f_1(E) - f_2(E)] \\
&= \frac{-qm_n^*}{2\pi^2 \hbar^3} \int dE_{||} \int dE_x T(E_{||}, E_x)[f_1(E) - f_2(E)].
\end{aligned}
$$

To further simplify the expression of J_n we need to assume Boltzmann statistics for the carrier occupation, i.e.,

$$
\begin{aligned}
f_1(E) &= \exp\left(\frac{E_{Fn1} - E_{||} - E_x}{k_B T}\right), \\
f_2(E) &= \exp\left(\frac{E_{Fn2} - E_{||} - E_x}{k_B T}\right).
\end{aligned}
$$

Moreover, we note that there is no barrier in the y and z directions at the junction, hence the transmission coefficient is 1 in the two directions, i.e., $T(E_{||}, E_x) = 1 \times T(E_x) = T(E_x)$. Then J_n becomes

$$
\begin{aligned}
J_n &= \frac{-qm_n^*}{2\pi^2 \hbar^3} \int_0^{+\infty} dE_{||} \exp\left(\frac{-E_{||}}{k_B T}\right) \int_{E_{min}}^{+\infty} dE_x T(E_x) \left[\exp\left(\frac{E_{Fn1} - E_x}{k_B T}\right) - \exp\left(\frac{E_{Fn2} - E_x}{k_B T}\right)\right] \\
&= \frac{-qm_n^* k_B T}{2\pi^2 \hbar^3} \int_{E_{min}}^{+\infty} dE_x T(E_x) \left[\exp\left(\frac{E_{Fn1} - E_x}{k_B T}\right) - \exp\left(\frac{E_{Fn2} - E_x}{k_B T}\right)\right] \\
&= \frac{-A_n^* T}{k_B} \int_{E_{min}}^{+\infty} dE_x T(E_x) \left[\exp\left(\frac{E_{Fn1} - E_x}{k_B T}\right) - \exp\left(\frac{E_{Fn2} - E_x}{k_B T}\right)\right],
\end{aligned}
$$

where A_n^* is the electron Richardson constant and defined as

$$A_n^* = \frac{qm_n^* k_B^2}{2\pi^2 \hbar^3} = \frac{4\pi qm_n^* k_B^2}{h^3}.$$

To perform the integration over the energy E_x in computing J_n, the energy lower bound E_{min} must be determined. In general, E_{min} can be any value between 0 and ΔE_C. Figure C-2 provides an example of conduction band diagram where $E_{min} > 0$, while the band diagram in Fig. C-1 has $E_{min} = 0$. It is noted that $T(E_x) = 1$ for $E_x > \Delta E_C$. Next, let us write $J_n = J_{n1} - J_{n2}$. Then we
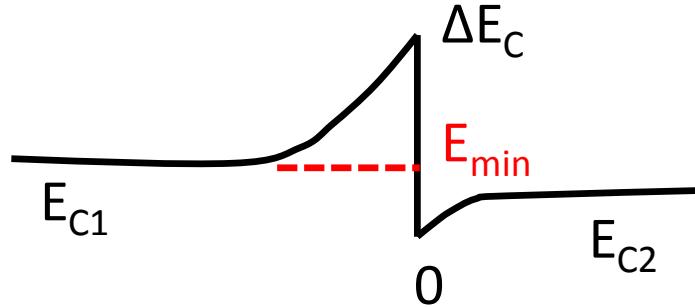

Figure C-2 Example of conduction band diagram illustrating the case of $E_{min} > 0$.

have

$$
\begin{aligned}
J_{n1} &= \frac{-A_n^* T}{k_B} \int_{E_{min}}^{\Delta E_C} dE_x T(E_x) \exp\left(\frac{E_{Fn1} - E_x}{k_B T}\right) - \frac{A_n^* T}{k_B} \int_{\Delta E_C}^{+\infty} dE_x \exp\left(\frac{E_{Fn1} - E_x}{k_B T}\right) \\
&= \frac{-A_n^* T}{k_B} \int_{E_{min}}^{\Delta E_C} dE_x T(E_x) \exp\left(\frac{E_{Fn1} - E_x}{k_B T}\right) - A_n^* T^2 \exp\left(\frac{E_{Fn1} - \Delta E_C}{k_B T}\right) \\
&= -A_n^* T^2 \exp\left(\frac{E_{Fn1} - \Delta E_C}{k_B T}\right) \left[1 + \frac{\exp\left(\frac{\Delta E_C}{k_B T}\right)}{k_B T} \int_{E_{min}}^{\Delta E_C} dE_x T(E_x) \exp\left(\frac{-E_x}{k_B T}\right) \right] \\
&= -A_n^* T^2 \exp\left(\frac{E_{Fn1} - \Delta E_C}{k_B T}\right) (1 + \delta_n),
\end{aligned}
$$

with δ_n equal to

$$\delta_n = \frac{\exp\left(\frac{\Delta E_C}{k_B T}\right)}{k_B T} \int_{E_{min}}^{\Delta E_C} dE_x T(E_x) \exp\left(\frac{-E_x}{k_B T}\right).$$

Similarly, J_{n2} is given by

$$J_{n2} = -A_n^* T^2 \exp\left(\frac{E_{Fn2} - \Delta E_C}{k_B T}\right) (1 + \delta_n).$$

Therefore, J_n is simplified to

$$J_n = J_{n1} - J_{n2} = -A_n^* T^2 \left[\exp\left(\frac{E_{Fn1} - \Delta E_C}{k_B T}\right) - \exp\left(\frac{E_{Fn2} - \Delta E_C}{k_B T}\right) \right] (1 + \delta_n).$$

188

From the conduction band diagrams in Figs. C-1 and C-2, it is seen that
$\Delta E_C = E_{C1} - E_{C2} = E_{C1} - 0 = E_{C1} > 0$ at the junction without loss of generality. Finally, J_n
across the heterojunction can be written as

$$
\begin{aligned}
J_n &= -A_n^* T^2 \left[\exp\left(\frac{E_{Fn1} - E_{C1}}{k_B T} \right) - \exp\left(\frac{E_{Fn2} - E_{C2}}{k_B T} \right) \exp\left(\frac{-\Delta E_C}{k_B T} \right) \right] (1 + \delta_n) \\
&= J_{TE,n}(1 + \delta_n),
\end{aligned}
$$

where $J_{TE,n}$ is the net electron thermionic emission current density and equal to

$$
J_{TE,n} = -A_n^* T^2 \left[\exp\left(\frac{E_{Fn1} - E_{C1}}{k_B T} \right) - \exp\left(\frac{E_{Fn2} - E_{C2}}{k_B T} \right) \exp\left(\frac{-\Delta E_C}{k_B T} \right) \right]. \tag{191}
$$

Next, in order to compute the tunneling contribution, i.e., δ_n, we focus on the conduction band
diagram in Fig. C-1 where $E_{min} = 0$, since the NPN HBT device of interest has a similar
conduction band profile. Applying the Wentzel-Kramers-Brillouin (WKB) approximation to the
tunneling, the transmission coefficient $T(E_x)$ for $0 \leq E_x \leq \Delta E_C$ is equal to

$$
T(E_x) = \exp\left[-2 \int_{x_w}^0 dx \sqrt{\frac{2m_{nt}^*}{\hbar^2}} \left[E_C(x) - E_x \right]^{1/2} \right].
$$

By approximating the conduction band in Fig. C-1 to a triangular shape, the conduction band
$E_C(x)$ for tunneling is given by

$$
E_C(x) = \frac{E_x - \Delta E_C}{x_w} x + \Delta E_C = q\xi x + \Delta E_C \ \text{ with } \ \xi = \frac{\nabla E_C}{q} = \frac{E_x - \Delta E_C}{q x_w},
$$

where ξ is the electric field in the tunneling source side (e.g., side 1 in Fig. C-1). Substituting
$E_C(x)$ into the expression of $T(E_x)$, we obtain

$$
\begin{aligned}
T(E_x) &= \exp\left[-2 \int_{x_w}^0 dx \sqrt{\frac{2m_{nt}^*}{\hbar^2}} \left(q\xi x + \Delta E_C - E_x \right)^{1/2} \right] \\
&= \exp\left[-2 \sqrt{\frac{2m_{nt}^*}{\hbar^2}} \frac{2}{3q\xi} (\Delta E_C - E_x)^{3/2} \right] \\
&= \exp\left[\frac{-8\pi}{3hq\xi} \sqrt{2m_{nt}^*} (\Delta E_C - E_x)^{3/2} \right].
\end{aligned}
$$

Given the above $T(E_x)$, δ_n can be rewritten as

$$
\begin{aligned}
\delta_n &= \frac{1}{k_B T} \int_0^{\Delta E_C} dE_x T(E_x) \exp\left(\frac{-E_x}{k_B T} \right) \exp\left(\frac{\Delta E_C}{k_B T} \right) \\
&= \frac{1}{k_B T} \int_0^{\Delta E_C} dE_x \exp\left[\frac{-8\pi}{3hq\xi} \sqrt{2m_{nt}^*} (\Delta E_C - E_x)^{3/2} \right] \exp\left(\frac{\Delta E_C - E_x}{k_B T} \right).
\end{aligned}
$$

189

Let $u = \frac{\Delta E_C - E_x}{k_B T}$, then $du = \frac{-1}{k_B T} dE_x$. δ_n is further simplified to

$$
\begin{aligned}
\delta_n &= \frac{1}{k_B T} \int_{\frac{\Delta E_C}{k_B T}}^{0} (-k_B T) \exp\left[\frac{-8\pi}{3hq\xi}\sqrt{2m_{nt}^*}(k_B T)^{3/2} u^{3/2}\right] \exp(u)\, du \\
&= \int_{0}^{\frac{\Delta E_C}{k_B T}} \exp\left[u - \left(\frac{u}{u_0}\right)^{3/2}\right] du,
\end{aligned}
\tag{192}
$$

where u_0 is given by

$$
u_0 = \frac{1}{k_B T}\left(\frac{3hq\xi}{8\pi\sqrt{2m_{nt}^*}}\right)^{2/3}.
$$

If the band diagram in Fig. C-1 is flipped in the x direction, ΔE_C will become negative, but Eq. (192) is still valid except using the absolute value of ΔE_C. It is noted that tunneling process is non-local by nature. However, by using the WKB approximation and the triangular barrier assumption, the quantum mechanical transmission coefficient $T(E_x)$ for tunneling depends only on the local electric field, hence the tunneling model given by Eq. (192) is called the local tunneling model.

APPENDIX D. CHARON INPUT FILE FOR A PN STEP JUNCTION DIODE EXAMPLE

This appendix contains input file listings for examples given in the document.

D.1. NLP input file for PN step junction diode in Section 3.4

.

Listing 1 Input file for NLP simulation of the PN step junction diode.

```
 1  # Name of the input exodus file, geometry only
 2  import state file pndiode.exo
 3
 4  # Output exodus file for results of this simulation
 5  start output parameters
 6     output state file pndiode.nlp.exo
 7  end output parameters
 8
 9  start physics block semiconductor
10
11     # geometry block name IS case sensitive, note however that Cubit
12     # often downcases names prior to output so it is recommended
13     # that in Cubit you use all lower case for naming entities to
14     # avoid confusion.
15     geometry block is si
16
17     # The type of physics to be solved, in this case a nonlinear
18     # Poisson, or nlp, simulation will be performed
19     standard discretization type is nlp
20
21     # The name of the material model IS case sensitive. The name is
22     # used as a key for the associated material block, also contained
23     # in this input file.
24     material model is siliconParameter
25
26  end physics block semiconductor
27
28  # The material block where most material parameters for this
29  # simulation are set. It is specified in the physics block by it's
30  # name, siliconParameter.
31  start material block siliconParameter
32
33     # Material name IS case sensitive
34     material name is Silicon
35
36     # Simple, scalar, material property
37     relative permittivity = 11.9
38
39     # The doping for the diode
40     start step junction doping
41        acceptor concentration = 1e16
```

191

```
42       donor concentration = 1e16
43       junction location = 0.5
44       dopant order is PN
45       direction is x
46    end step junction doping
47
48  end material block siliconParameter
49
50  # Boundary condition specifications. Required, but must be set to zero
51  # for the NLP simulation
52  bc is ohmic for anode on si fixed at 0
53  bc is ohmic for cathode on si fixed at 0
54
55  # Initial conditions for the ELECTRIC_POTENTIAL will come from an
56  # equilibrium approximation
57  initial conditions for ELECTRIC_POTENTIAL in si is equilibrium potential
58
59  # Section to specify the solvers, nonlinear and linear, for the
60  # problem. Most of the settings are encapsulated in "solver packs" for
61  # convenience.
62  start solver block
63     use solver pack 1
64  end solver block
```

D.2. I-V sweep input file for PN step junction diode in Section 3.5

Listing 2 Input file for I-V sweep of the PN step junction diode.

```
 1  # Name of the input exodus file, which includes the results for
 2  # ELECTRIC_POTENTIAL as obtained from a previous NLP simulation
 3  import state file pndiode.nlp.exo at index 1
 4
 5  # Output exodus file for results of this simulation
 6  start output parameters
 7    output state file pndiode.dd.iv.exo
 8  end output parameters
 9
10  start physics block Semiconductor
11
12     # geometry block name IS case sensitive, note however that Cubit
13     # often downcases names prior to output so it is recommended that
14     # in Cubit you use all lower case for naming entities to avoid
15     # confusion.
16     geometry block is si
17
18     # The type of physics to be solved, in this case a nonlinear
19     # Poisson, or nlp, simulation will be performed
20     standard discretization type is drift diffusion gfem
21
22     # The name of the material model IS case sensitive. The name is
23     # used as a key for the associated material block, also contained
24     # in this input file.
25     material model is siliconParameter
```

```
26
27    # Turn on Schokley-Reed-Hall recombination
28    srh recombination is on
29
30  end physics block
31
32  # The material block where most material parameters for this
33  # simulation are set. It is specified in the physics block by it's
34  # name, siliconParameter.
35  start material block siliconParameter
36
37    # Material name IS case sensitive
38    material name is Silicon
39    relative permittivity = 11.9
40
41    start step junction doping
42       acceptor concentration =1.0e16
43       donor concentration =1.0e16
44       junction location = 0.5
45       dopant order is PN
46       direction is x
47       end step junction doping
48
49  end material block siliconParameter
50
51  # This is taken from the NLP file. Note that it is read from "index 1"
52  # as specified in the input file specification. In this case there is
53  # only a single result in that file.
54  initial conditions for ELECTRIC_POTENTIAL in si is exodus file
55
56  # The NLP simulation does not include a solution for the carrier
57  # densities, therefore some other type of estimate, in this case an
58  # equilibrium calculation, is used to obtain an initial guess for
59  # the carrier densities.
60  initial conditions for ELECTRON_DENSITY in si is equilibrium density
61  initial conditions for HOLE_DENSITY in si is equilibrium density
62
63  # Boundary condtions at the contacts.
64  bc is ohmic for cathode on si fixed at 0.0
65  bc is ohmic for anode on si swept from 0.0 to 1.0
66
67  # Sweep parameter controls
68  start sweep options
69     initial step size = 0.02
70     minimum step size = 0.02
71     maximum step size = 0.02
72  end sweep options
73
74  # Use a straightforward solver pack for this simulation.
75  start solver block
76    use solver pack 1
77  end solver block
```

INDEX