

Sentaurus™ Device Monte Carlo User Guide

Version T-2022.03, March 2022

SYNOPSYS®

Copyright and Proprietary Information Notice

© 2022 Synopsys, Inc. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <https://www.synopsys.com/company/legal/trademarks-brands.html>.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Contents

Conventions	16
Customer Support	16
Acknowledgments	17

Part I: Single-Particle Device Monte Carlo

1. Simulation Procedure	19
Initial Sentaurus Device Simulation	19
Single-Particle Device Monte Carlo Simulation	23
Screen Output of Simulation	27
Results of Simulation	29
Ramping of Boundary Conditions	30
Parallelization	33
Parallel Simulation Procedure	33
Parallel Simulations and Sentaurus Workbench	34
Licensing	34
2. Input Specification	35
File Section	35
Math Section	35
Solve Section	36
Monte Carlo Post-Solve	36
Plot Section	36
MonteCarlo Section	38
Coordinate System	48
3. Physical and Numeric Models	49
Models for Band Structure and Scattering Mechanisms	49

Contents

Surface Roughness Scattering	53
Fermi Statistics	54
Arbitrary Lattice Temperature.	57
Trajectory Calculation	57
Self-Consistent Single-Particle Approach	59
Gathering Statistics	60
Estimating Currents	61
Drain Current	61
Substrate Current.	61
Contact Resistance	62
Averages and Statistical Error of the Currents.	62
Considering Traps	63
Quantum Correction.	64
Effective Quantum Correction	64
Density-Gradient Monte Carlo	66
Visualizing Energy Distributions.	66
Visualizing Valley Occupations	67
Averages Over Forward- and Backward-Flying Carriers	67
Ballistic Transport.	68
References.	70
<hr/> 4. Strained Silicon	73
Bulk Properties.	73
NMOSFETs	80
On-Current Interpretation.	82
PMOSFETs	84
Arbitrary Stress	86
References.	87
<hr/> 5. Surface and Channel Orientations	90
Changing Surface and Channel Orientations	90
TCAD Sentaurus Tutorial: Simulation Projects	92

Contents

References	93
<hr/>	
6. Stress-Dependent Built-in Analytic Band Structures	94
Stress Engineering	94
Visualizing Band Structures	95
Hole Band Structure	96
Electron Band Structure	97
References	101
<hr/>	
7. Electron and Hole Transport	103
Electron and Hole Transport in Strained SiGe	103
Hole Band Structure	103
Scattering Mechanisms	105
Simulation Procedure and Results	106
Electron Transport	107
Electron Transport in InGaAs	108
Features of Electron Transport	108
References	110
<hr/>	
8. Mobility Reduction in High-k Gate Stacks	112
Introduction	112
Soft-Optical Phonon Scattering	113
High-k Mobility in the Presence of an Interfacial Oxide	114
High-k Mobility in the Absence of an Interfacial Oxide	116
Remote Coulomb Scattering	117
References	118
<hr/>	
9. Example: NMOS Transistor	119
Simulation of an NMOS Transistor	119

Contents

Part II: Band Structure and Mobility Calculation

10. Using Sentaurus Band Structure	123
Introduction to Sentaurus Band Structure	123
Bulk Band Structure	123
Subband Structure and Inversion Mobility	124
TCAD Sentaurus Tutorial: Tcl Interface	124
Starting Sentaurus Band Structure	124
Command-Line Options	125
EPM Band-Structure Tutorial	127
Creating a Silicon Crystal	127
Computing Band-Structure Data	127
Inspecting the Results	128
Finding the Band Minimum	130
Computing Effective Masses	130
Parallelization	132
Applying Strain	132
Biaxial Strain	133
Uniaxial Strain	133
Strain Tensor From Stress Tensor in Principal-Axis System	134
Generating Band-Structure Tables From k-Vector Files	134
Taking Advantage of Band-Structure Symmetries	135
File Formats for Storing Band Data	136
File Format \$SHORT_FORMAT	136
File Format \$LONG_FORMAT	139
Creating Band Data for Sentaurus Device Monte Carlo	139
Creating Band Data for Garand MC	140
Analytic Band-Structure Models	140
Complex Band Structures	141
Calculating Electron Subbands and Mobility	142
Loading a 1D Device Structure	142
Specifying Top-Level Parameters of Physics Command	142
Performing an Initial Solve	143
Defining a Nonlocal Line	143
Specifying a Schrödinger Solver	143

Contents

Specifying a Mobility Calculator	144
Performing a Self-Consistent Solve With the Schrödinger Equation	144
Computing the Mobility	144
Saving Models Over the Device Structure	145
Saving Models Over 2D k-Space	145
Ramping the Bias	145
11. Empirical Pseudopotential Method	147
Introduction to Pseudopotentials	147
Empirical Pseudopotentials	149
Using the Crystal Symmetry	149
Nonlocal Corrections to the Pseudopotential	151
Spin-Orbit Coupling	153
Evaluating Radial Overlap Integrals	155
Normalizing the Spin-Orbit Term	156
Strained Materials	157
Bulk Strain and Internal Strain	157
Empirical Pseudopotential Method for Strained Materials	158
Local Pseudopotential in Strained Crystals	158
Cubic Spline Interpolation of Local Pseudopotential	159
Friedel Interpolation Formula for Local Pseudopotential	160
Plane-Wave Normalization in Strained Crystals	160
Alloys and the Virtual Crystal Approximation	160
EPM Model Defaults	161
Group Velocities and Effective Masses	162
References	164
12. Analytic Bands for Bulk Crystals	165
Conduction Bands	165
Two-Band $k \cdot p$ Model	166
Ellipsoidal Model	166
Nonparabolicity Model	168
Model Parameters	168
Valence Bands	169
Theory	169
Luttinger–Kohn or Bir–Pikus Hamiltonian	170
Model Parameters	171

Contents

References	172
<hr/>	
13. Subband and Mobility Calculations	173
Device Structure	173
Regions and Materials	173
Contacts	174
Nonlocal Lines and Nonlocal Areas	174
Device Coordinate Axes	174
Axes for 1D Devices	174
Axes for 2D Devices	175
Axes for Wurtzite Crystal Semiconductors	176
Files Used by Subband and Mobility Calculations	176
Input Files	177
Output Files	177
Poisson Equation	178
Boundary Conditions	179
Electric Field	179
Convergence	179
Valley Models	180
Common Syntax	180
Automatic Generation of Bulk Density and Band-Edge Models	181
Ellipsoidal Valleys	181
ConstantEllipsoid Valley Model	181
2kpEllipsoid Valley Model	182
Valleys Based on $k \cdot p$ Theory	183
The 6kpValley Model	184
The 3kpValley Model	184
The 8kpValley Model	185
The 2kpValley Model	186
Default Valley Models in Silicon	187
Electrostatic Models	188
Permittivity	188
Relaxed Band Models	188
Quasi-Fermi Levels	189
Carrier Density	190
Bulk Versus Confined Carrier Density	190
Fermi–Dirac Bulk Density Model	191
Bulk Density Model Based on an Ellipsoidal Valley	192
Bulk Hole Density Model Based on 6kp	192

Contents

Multivalley Bulk Density Model	193
Default Models for Carrier Density	193
Maxwell–Boltzmann Statistics	193
Doping Concentration	194
Net Density	195
Interface Charge	195
Fixed Interface Charge	196
Interface Traps	196
Visualization of Interface Charge	198
Effective Field for Universal Mobility	199
Strain	200
Polarization-Induced Charge	200
Visualization of Polarization-Induced Charges	202
Calculating Subbands	203
Polar Grid for 2D k-Space	203
Grid for 1D k-Space	204
Ladders	204
Dynamically Varying Subband Numbers	204
Calculating the Confined Carrier Density	205
Calculating the Thermal Injection Velocity	206
Calculating the Transport Mass	206
Automatically Created Models for Visualization and Extraction	207
Parabolic Schrödinger Solver	208
Dispersion	209
Surface Orientation and Effective Masses	210
Using the Parabolic Schrödinger Solver	211
Confined $k \cdot p$ Schrödinger Solvers	211
Arbitrary Surface Orientations	211
Reordering the Subband Dispersion	212
Confined Six-Band $k \cdot p$ Schrödinger Solver	213
Confined Three-Band $k \cdot p$ Schrödinger Solver	217
Confined Two-Band $k \cdot p$ Schrödinger Solver	217
Confined Eight-Band $k \cdot p$ Schrödinger Solver	219
Handling Subband Degeneracies	223
Interface Potential Spike	223
Visualizing the Interface Potential Spike	224
Using Sentaurus Band Structure as an External Schrödinger Solver for Sentaurus Device	224
Calculating Mobility	226
Kubo–Greenwood Formalism	226
Inverse Momentum Relaxation Time	227

Contents

Linear Boltzmann Transport Equation	227
Mobility Calculators	228
k-Space Grid	228
Using Mobility Calculators	228
Visualization Quantities	229
Coulomb Green's Function	229
Spatial Mobility Profile	230
Scattering Models	231
Common Parameters	231
Valleys	231
Transition Type	231
Momentum Relaxation Factor	232
Screening	233
Scattering Model Availability by Device Dimension	236
Phonon Scattering	237
Acoustic Phonon Scattering	237
Ohashi Acoustic Phonon Scattering	239
Inelastic Phonon Scattering	240
Evaluating the Wavefunction Overlap Form-Factor	241
Tensor Acoustic Phonon Scattering	241
Tensor Optical Phonon Scattering	243
Polar-Optical Phonon Scattering	244
Remote Soft-Optical Phonon Scattering	245
Alloy Disorder Scattering	247
Surface Roughness Scattering	248
Power-Spectral Density Function	248
Common Parameters	249
Surface Roughness Scattering From Parabolic Schrödinger for 1D	249
Surface Roughness Scattering From 6kp Schrödinger for 1D	250
Isotropic Prange–Nee Surface Roughness Scattering From 6kp Schrödinger for 1D	250
Surface Roughness Scattering for Two Dimensions	251
Coulomb Scattering	252
Common Parameters	253
Speeding Up the Matrix Element Calculation	253
Usage	254
Default Scattering Models	254
Material Database	256
Applying Materials to a Loaded Device	258
Viewing the Mapping of Materials to a Loaded Device	259
Viewing Material Parameters	260
Defining Binary Alloy Materials	261

Contents

Defining Interpolated Parameters	261
Parameter Interpolation Table	262
Interpolation of Material-Specific Scattering Models	263
References	265
14. Sentaurus Band Structure/Tcl Commands	268
Overview of Commands and Syntax	268
Notational Conventions of Syntax Description	270
Positional Arguments	270
Named Arguments	270
Object-Oriented Tcl Commands	271
Classes for the Band-Structure Calculation	271
Material Parameter Files	272
Global Settings	273
sBandGet and sBandSet	273
EPM::Crystal Objects	277
Creating Standard and Custom EPM::Crystal Objects	277
compute_SiGe_a0	278
Crystal	278
DiamondCrystal	279
GermaniumCrystal	280
SiGeCrystal	280
SiliconCrystal	281
StrainedCrystal	282
Using EPM::Crystal Objects	283
<EPM::Crystal> addAtom	283
<EPM::Crystal> apply	284
<EPM::Crystal> computeBandstructure	285
<EPM::Crystal> computeDOS	287
<EPM::Crystal> destroy	288
<EPM::Crystal> get	288
<EPM::Crystal> scaleKvector	289
<EPM::Crystal> set	289
<EPM::Crystal> status	290
<EPM::Crystal> unscaleKvector	291
EPM::AtomicSpecies Objects	291
Creating and Loading EPM::AtomicSpecies Objects	291
AtomicSpecies	292
requireAtomicSpecies	296
Using EPM::AtomicSpecies Objects	296

Contents

<EPM::AtomicSpecies> destroy	297
<EPM::AtomicSpecies> dynamicType	297
<EPM::AtomicSpecies> get	297
<EPM::AtomicSpecies> rename	299
<EPM::AtomicSpecies> set	299
<EPM::AtomicSpecies> status	300
AnalyticBandSolver Objects	301
Creating AnalyticBandSolver Objects	301
AnalyticBandSolver	301
AnalyticBandSolverFromSpecies	301
Using AnalyticBandSolver Objects	302
<AnalyticBandSolver> apply	303
<AnalyticBandSolver> computeBandstructure	303
<AnalyticBandSolver> computeDOS	304
<AnalyticBandSolver> copy	306
<AnalyticBandSolver> destroy	306
<AnalyticBandSolver> get	306
<AnalyticBandSolver> set	306
<AnalyticBandSolver> status	308
Elasticity Objects	308
Creating Elasticity Objects	308
Elasticity	308
Using Elasticity Objects	309
<Elasticity> biaxialStrain	310
<Elasticity> biaxialStrainRatio	311
<Elasticity> copy	311
<Elasticity> destroy	311
<Elasticity> dynamicType	312
<Elasticity> status	312
<Elasticity> strainFromStress	312
<Elasticity> uniaxialStrain	313
Band-Structure Operations	314
determineSymmetry	314
Creating Input Files	314
createGarandMCFFiles	314
createMonteCarloFiles	315
Finding Conduction Band Valley Positions	316
findBandMinimum	316
findBandMinima	318
Container Objects for Band-Structure Results	319

Contents

bandstructure_t	319
Using <bandstructure_t> Objects	320
groupVelocity_t	321
Using <groupVelocity_t> Objects	321
inverseMass_t	322
Using <inverseMass_t> Objects	323
3D Vector/Matrix Auxiliary Commands	323
crossProduct	323
invertMatrix	324
matrixProduct	324
matrixVectorProduct	325
scalarProduct	325
solveLinearEquation	326
transposeMatrix	326
unitVec	327
vectorAdd	327
vectorDivide	328
vectorLength	328
vectorMultiply	329
vectorSubtract	329
Tcl Support	330
Support for Complex Numbers	330
imaginaryPart	330
realPart	330
Support for Named Argument Parsing	331
check_args	331
get_arg	332
get_known_args	333
get_unknown_args	334
has_arg	334
init_arg	335
parse_args	337
Support for Type Checking	338
check_type	338
common_type	339
get_type	339
General Utilities	340
import_array	340
tee	341
Material Database	342

Contents

CreateMaterial and ModifyMaterial	342
CloneMaterial	345
ListMaterials	346
RemoveMaterial	346
AddValley and ModifyValley	346
RemoveValley	348
AddScatteringModel and ModifyScatteringModel	349
RemoveScatteringModel	351
ApplyMaterial	351
PrintParameters	352
ListMaterialMap	354
Subband and Mobility Calculations	354
AddToFile	355
ComputeMass	355
ComputeMobility	356
ComputeVinj.	357
Dopant	358
Extract	359
GetLast.	360
LoadDevice	360
Material	361
Math	362
Physics	365
Physics for Top-Level Parameters	365
Physics for Contacts	367
Physics for Electrostatic Models	368
Physics for Valley Models	369
Physics for Scattering Models	371
Physics for Trap Models	374
Physics for Interface Potential Spike Models	377
Physics of Nonlocal Lines or Nonlocal Areas	379
Save	383
SaveDitProfile	385
SaveK.	385
Solve	386
References	389

About This Guide

The *Sentaurus™ Device Monte Carlo User Guide* consists of the module for Monte Carlo simulation – single-particle device Monte Carlo – as well as the band-structure and mobility calculator Sentaurus Band Structure.

In the *single-particle* device Monte Carlo simulator, one particle (electron or hole) after another is simulated in the device as it travels from one contact to another until good statistics for the charge density are obtained. Then, the nonlinear Poisson equation is solved. This procedure is iterated until convergence. The result is a self-consistent steady-state solution of the Boltzmann and Poisson equations. The tool can simulate process-simulated devices with unstructured grids. It permits you to consider silicon, germanium, or SiGe band structures under arbitrary stress conditions as well as arbitrary crystallographic surface and channel orientations, as well as InGaAs for electrons. It also permits users to import externally generated band structure tables. Alternatively, built-in analytic band models can be used for a given strain tensor. The simulation results have been found to be in good agreement with measured on-currents in the nanoscale regime, and the tool has been applied to advanced strained-silicon devices at the scaling limit of CMOS. Arbitrary lattice temperatures and mobility reduction in high- κ gate stacks can be considered. The single-particle device Monte Carlo simulator can be run in parallelized mode featuring a high speedup with the number of cores used. One-dimensional, 2D, and 3D devices can be simulated.

Sentaurus Band Structure can be used to compute tables of full-band structure under arbitrary strain for use by Monte Carlo simulation. In addition, the mobility calculator feature can be used to compute the subband structure and inversion layer mobility of 1D device structures under arbitrary strain and device orientation.

The *Sentaurus™ Device Monte Carlo User Guide* is divided into the following parts:

- Part I discusses aspects of single-particle device Monte Carlo simulation.
- Part II discusses aspects of Sentaurus Band Structure.

For additional information, see:

- The TCAD Sentaurus release notes, available on the Synopsys SolvNetPlus support site (see [Accessing SolvNetPlus on page 16](#))
- Documentation available on the SolvNetPlus support site

About This Guide

Conventions

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Bold text	Identifies a selectable icon, button, menu, or tab. It also indicates the name of a field or an option.
Courier font	Identifies text that is displayed on the screen or that you must type. It identifies the names of files, directories, paths, parameters, keywords, and variables.
<i>Italicized text</i>	Used for emphasis, the titles of books and journals, and non-English words. It also identifies components of an equation or a formula, a placeholder, or an identifier.

Customer Support

Customer support is available through the Synopsys SolvNetPlus support site and by contacting the Synopsys support center.

Accessing SolvNetPlus

The SolvNetPlus support site includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The site also gives you access to a wide range of Synopsys online services, which include downloading software, viewing documentation, and entering a call to the Support Center.

To access the SolvNetPlus site:

1. Go to <https://solvnetplus.synopsys.com>.
 2. Enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register.)
-

Contacting Synopsys Support

If you have problems, questions, or suggestions, you can contact Synopsys support in the following ways:

- Go to the Synopsys [Global Support Centers](#) site on www.synopsys.com. There you can find email addresses and telephone numbers for Synopsys support centers throughout the world.

About This Guide

Acknowledgments

- Go to either the Synopsys SolvNetPlus site or the Synopsys Global Support Centers site and open a case (Synopsys user name and password required).

Contacting Your Local TCAD Support Team Directly

Send an email message to:

- support-tcad-us@synopsys.com from within North America and South America
- support-tcad-eu@synopsys.com from within Europe
- support-tcad-ap@synopsys.com from within Asia Pacific (China, Taiwan, Singapore, Malaysia, India, Australia)
- support-tcad-kr@synopsys.com from Korea
- support-tcad-jp@synopsys.com from Japan

Acknowledgments

The single-particle device Monte Carlo simulator was codeveloped by Integrated Systems Laboratory of ETH Zurich in the joint research project HYBRID with financial support by the Swiss funding agency CTI.

Part I: Single-Particle Device Monte Carlo

This part of the *Sentaurus™ Device Monte Carlo User Guide* contains the following chapters:

- [Chapter 1, Simulation Procedure](#)
- [Chapter 2, Input Specification](#)
- [Chapter 3, Physical and Numeric Models](#)
- [Chapter 4, Strained Silicon](#)
- [Chapter 5, Surface and Channel Orientations](#)
- [Chapter 6, Stress-Dependent Built-in Analytic Band Structures](#)
- [Chapter 7, Electron and Hole Transport](#)
- [Chapter 8, Mobility Reduction in High-k Gate Stacks](#)
- [Chapter 9, Example: NMOS Transistor](#)

1

Simulation Procedure

This chapter explains how to perform a single-particle device Monte Carlo simulation. For details about Sentaurus Device simulations and the input syntax, see the Sentaurus™ Device User Guide.

A Monte Carlo simulation has two main steps. First, a drift-diffusion simulation is run. Second, based on this, a Monte Carlo simulation is performed. Typically, the Monte Carlo method is applied to a window that excludes the polysilicon region, but covers most of the device, including almost all parts of the gate oxide.

The drift-diffusion simulation yields an electric field that is used in the initial frozen-field simulation of the iteration between a Monte Carlo transport simulation and the Poisson equation. The drift-diffusion simulation also predicts the density distribution of electrons and holes. The tool integrates these density distributions over the entire Monte Carlo window and simulates only the carrier type with the greater integral of the density. Furthermore, the tool assumes that the integrated density of the predominant carrier type is predicted correctly by the drift-diffusion simulation. Consequently, compared to a drift-diffusion simulation, a Monte Carlo simulation changes only the shape of the density distribution, but not the total charge in the Monte Carlo window.

Initial Sentaurus Device Simulation

The command file of the drift-diffusion simulation `drift_new.cmd` is:

```
File {
    grid      = "n5_msh.tdr"
    current   = "drift_new"
    output    = "drift_new"
    plot      = "drift_new"
    save      = "drift_new"
    param     = "nmos"
}
Plot {
    eVelocity/Vector
    eCurrent/Vector
    hCurrent/Vector
```

Chapter 1: Simulation Procedure

Initial Sentaurus Device Simulation

```
ElectricField/Vector
eDensity hDensity potential
ConductionBandEnergy ValenceBandEnergy
GradConductionBand GradValenceBand
}
Electrode {
    { name=gate voltage=1.2 barrier=0.06 }
    { name=bulk voltage=0 }
    { name=source voltage=0 }
    { name=drain voltage=0.0 }
}
Physics {
    mobility (
        highfieldsaturation
        Enormal
        PhuMob
    )
    EffectiveIntrinsicDensity ( Slotboom NoFermi )
}
Math {
    method=blocked
    submethod=pardiso
    wallclock
    Extrapolate
    Derivatives
    RelErrControl
    Digits=5
    ErRef(electron)=1.e10
    ErRef(hole)=1.e10
    Notdamped=50
    Iterations=50
}
Solve {
    ----- solution at initial conditions
    Poisson
    coupled {poisson electron hole}

    ----- ramp
    Quasistationary ( InitialStep=0.001 MinStep=1.0e-5 MaxStep=0.1
    goal { name=drain voltage=1.2 }
    ) {
        coupled {poisson electron hole}
    }
}
```

The 0.1 μm NMOS transistor of the example is simulated.

The `File` section contains all of the files needed for the simulation. Grid and doping information are read from the file `n5_msh.tdr`.

Chapter 1: Simulation Procedure

Initial Sentaurus Device Simulation

The results are written to the following files:

- `drift_new_des.plt` (terminal currents)
- `drift_new_des.tdr` (plot file of all quantities defined in the `Plot` section for visualization with Sentaurus Visual)
- `drift_new_des.sav` (save file from which the Monte Carlo simulation is started)
- `drift_new_des.log` (textual output)

File extensions such as `.tdr` do not have to appear in the command file because the program adds them automatically if they are missing. For more information about the predefined extensions, see the *Sentaurus™ Device User Guide*.

The parameter file `nmos.par` contains the parameter values for the various physical models specified in the `Physics` section and is given by:

```
Bandgap
{ * Eg = Eg0 + dEg0 + alpha Tpar^2 / (beta + Tpar) -
  alpha T^2 / (beta + T)
  Chi0 = 4.05                      # [eV]
  Eg0 = 1.12                        # [eV]
  dEg0(Slotboom) = 0.0e+00          # [eV]
  alpha = 0.00e+00                   # [eV K^-1]
  beta = 0.00e+00                    # [K]
  Tpar = 300.0000e+00                # [K]
}
eDOSMass {
  Formula = 2                      # [1]
  Nc300   = 2.97101e+19             # [cm^-3]
}
hDOSMass {
  Formula = 2                      # [1]
  Nv300   = 2.2400e+19              # [cm^-3]
}
ConstantMobility:
{ * mu_const = mumax (T/T0)^(-Exponent)
  mumax = 1.423e+03 , 476.070     # [cm^2/(Vs)]
}
HighFieldDependence:
{ * Caughey-Thomas model:
  * mu_highfield = mu_lowfield / ( 1 + (mu_lowfield E / vsat)^beta )
  * ^1/beta * beta = beta0 (T/T0)^betaexp.
  beta0 = 1.13417 , 1.213 # [1]

  * Formula1 for saturation velocity:
  * vsat = vsat0 (T/T0)^(-Vsatexp)
  * (Parameter Vsat_Formula has to be not equal to 2):
    vsat0 = 1.0200e+07 , 8.3700e+06 # [1]
}
PhuMob:
{ * Philips Unified Mobility Model:
```

Chapter 1: Simulation Procedure

Initial Sentaurus Device Simulation

```
mumax_As = 1.423e+03      # [cm^2/Vs]
mumin_As = 55.9            # [cm^2/Vs]
mumax_P = 1.423e+03        # [cm^2/Vs]
mumax_B = 476.070          # [cm^2/Vs]
}NormalMob
{ * mu_Enorm^(-1) = mu_ac^(-1) + mu_sr^(-1) with:
  * mu_ac = B / Enorm + C (T/T0)^(-1) (N/N0)^lambda / Enorm^(1/3) )
  * mu_sr^-1 = Enorm^(A+alpha*n/N^nu) / delta + Enorm^3 / eta
  * EnormalDependence is added with factor exp(-l/l_crit), where l is
  * the distance to nearest point of DES_c_Si/DES_c_SiO2 interface.
  * Factor   * is equal to 1 if l_crit > 100.
    B      = 3.6100e+07 , 1.5100e+07      # [cm/s]
    C      = 4.0e+08 , 4.1800e+03      # [cm^5/3/(sV^2/3)]
    N0     = 1 , 1                      # [cm^-3]
    lambda = -0.2399 , 0.0119          # [1]
    delta   = 3.5800e+18 , 4.1000e+15    # [V/s]
    A       = 2.58 , 2.18                # [1]
    alpha   = 6.8500e-21 , 7.8200e-21    # [1]
    nu      = 0.0767 , 0.123             # [1]
    eta     = 5.8200e+30 , 2.0546e+30    # [V^2/cm*s]
    l_crit  = 1.0000e-06 , 1.0000e-06    # [cm]
}
```

Note:

The single electron in the simulation carries the whole electron charge as obtained by integrating the electron density of Sentaurus Device over the whole device. Therefore, the physical models of the Sentaurus Device simulation should be as consistent as possible with the Monte Carlo model. For example, the effective densities-of-states N_c and N_v must correspond to the values resulting from the full band structure, and Boltzmann statistics should be used as in the Monte Carlo simulation. The parameters of Sentaurus Device listed above ensure this consistency for the simulation example.

In the Plot section, the quantities to be viewed in a visualization tool, as a function of position, are defined.

In the Electrode section, the initial voltage at the gate contact is defined to be 1.2 V and 0.0 V at all other contacts. When the polysilicon region is also included in the simulation, as in the example, the value in the barrier variable only serves to take into account the threshold shift due to quantum effects.

In the Physics section, all the adjustable physical models can be defined.

Note:

For a meaningful comparison of the simulation results at high drain voltages between Sentaurus Device and single-particle device Monte Carlo, the drain currents should coincide at low drain voltages. Since the surface mobility models for Sentaurus Device and single-particle device Monte Carlo are different, the parameters of Sentaurus Device for this model (keyword Enormal in the Physics

Chapter 1: Simulation Procedure

Single-Particle Device Monte Carlo Simulation

section, and `ENormalMob` in the parameter file `nmos.par`) should be adjusted at gate voltage=supply voltage, and low drain voltage (for example, 0.05 V or 0.1 V), so that the drain currents of Sentaurus Device and single-particle device Monte Carlo are approximately the same (see the previous note).

In the `Math` section, some parameters of the numeric methods used are specified. Extrapolation is used in the quasistationary simulation. The variables for the initial condition for a given quasistationary step are computed using an extrapolation from the previous step.

In the `Solve` section, the equations that Sentaurus Device must solve and how they are to be solved are defined. In this example, first, there is an independent solution of the Poisson equation. This is followed by a self-consistent solution of three equations: the Poisson equation, electron continuity equation, and hole continuity equation. This gives the solution for the specified initial conditions. Then, there is a quasistationary and simultaneous calculation of the same three equations, which ramps the voltage of the drain contact from 0 V to 1.2 V. Only at the end of the quasistationary calculation are the output files `drift_new_des.xxx` generated.

The simulation is run by typing:

```
sdevice drift_new
```

Single-Particle Device Monte Carlo Simulation

When the program finishes, the save file `drift_new_des.sav` is created, which corresponds to a voltage of 1.2 V at the drain contact. This solution is used to start a Monte Carlo simulation using the command file `mc_new.cmd`.

```
File {
    grid      = "n5_msh.tdr"
    current   = "mc_new"
    output    = "mc_new"
    plot      = "mc_new"
    load      = "drift_new"
    param     = "nmos"
    MonteCarloOut = "mc_new"
}
MonteCarlo {
    CurrentErrorBar = 2.5
    MinCurrentComput = 19
    DrainContact = 1      # No. of drain contact in .tdr (count from 0)
    useTestFunction
    SelfConsistent(FrozenQF)
    SurfScattRatio = 0.85
    Window = Rectangle [(-0.225,-0.00201) (0.225, 4.25)]
    FinalTime = 4.0e-6    # Simulation time until stationary state
    Plot { Range= (0,40.e-6) intervals=100 }    # Total simulation time
}
Plot {
```

Chapter 1: Simulation Procedure

Single-Particle Device Monte Carlo Simulation

```
MCFIELD/VECTOR
eMCDENSITY hMCDENSITY
eMCENERGY hMCENERGY
eMCVELOCITY/VECTOR hMCVELOCITY/VECTOR
eMCavalanche hMCavalanche
eMCCURRENT/VECTOR hMCCURRENT/VECTOR
}
Electrode {
    { name=gate voltage=1.2 barrier=0.06 }      # quantum threshold shift
    { name=bulk voltage=0 }                      # bulk
    { name=source voltage=0 }                    # source
    { name=drain voltage=1.2 }                  # drain
}
Thermode {
    { name=gate temperature=300 }                # gate
    { name=bulk temperature=300 }                # bulk
    { name=source temperature=300 }              # source
    { name=drain temperature=300 }               # drain
}
Physics {
    mobility (
        highfieldsaturation
        Enormal
        PhuMob
    )
    EffectiveIntrinsicDensity ( Slotboom NoFermi )
}
Math {
    method=pardiso
    wallclock
    Extrapolate
    Derivatives
    RelErrControl
    Digits=5
    ErRef(electron)=1.e10
    ErRef(hole)=1.e10
    Notdamped=50
    Iterations=50
    currentweighting
}
Solve {
    coupled {poisson electron hole}
    montecarlo
}
```

In the `File` section, the names of the output files are changed and three new entries are added. The file `drift_new_des.sav` is specified to be loaded at the beginning of the simulation and is used as initial condition for the simulation. The keyword `MonteCarloOut` determines the prefix of the Monte Carlo output files with the results for the currents.

In the `Electrode` section, a voltage of 1.2 V at the drain contact is specified, which corresponds to the voltage used in the save file `drift_new_des.sav`.

Chapter 1: Simulation Procedure

Single-Particle Device Monte Carlo Simulation

In the `Math` section, the keyword `currentweighting` is added. This keyword activates the computation of the drain current by using the test function method, and the keyword `useTestFunction` must be present in the Sentaurus Device Monte Carlo command file if the drain current is to be estimated by the test function method. Otherwise, the drain current is calculated by direct particle counting.

To take advantage of modern multicore machines, the tool supports parallelization of particles through the keyword `NumberOfSolverThreads` in the `Math` section of the Sentaurus Device command file (see *Sentaurus™ Device User Guide*, Table 211).

In the `MonteCarlo` section, the number of a contact in the file `n5_msh.tdr` can be given following the keyword `DrainContact`. For this contact, the tool computes the current and its statistical error. Despite the name `DrainContact`, any contact can be specified. In practice, always select a contact with a comparatively high current; otherwise, the statistical error of the computed current will be very high.

Note:

The numbering of the contact names in the grid file begins with zero.

Then, a Monte Carlo `window` must be defined. The window is a rectangle with edges parallel to the axes and should consist of the entire MOSFET, excluding the polysilicon region, and including the major part of the gate oxide. The gate oxide is included so that the change of the oxide field is considered during self-consistent Monte Carlo simulations.

For self-consistent simulations, it is necessary to consider that a certain number of iterations are required to reach self-consistency between the carrier distribution and electrostatic potential. Only after this time does the simulation fluctuate around the stationary solution and the gathering of statistics begins. The tool cannot determine the simulation time required to reach the steady state automatically; it must be specified explicitly by the keyword `FinalTime`.

In contrast, the maximum total simulation time is given in the `Plot` keyword as the second number in the `Range` interval. Finally, `intervals` specifies the number of intervals into which the maximum total simulation time is divided. After each interval, an estimation for drain and substrate currents is performed, and a plot file for visualization of the internal variables is generated (for example, `mc_new_000001_des.tdr` after the first interval). Before reaching the steady state, the estimates for the internal variables in the plot file correspond to the last interval only. After that time, cumulative expectation values are displayed.

In the example, `FinalTime = 4 μs`, maximum total simulation time = `40 μs`, and `intervals = 100`. This means that the interval for one iteration lasts `0.4 μs` and cumulative averaging begins after ten iterations. The estimates of the internal variables in the plot files beginning with the eleventh interval correspond to averages over all but the initial ten intervals. For example, the variables in `mc_new_000024_des.tdr` result from averaging over 14 intervals.

For solving the Poisson equation at the end of each simulation time interval, the tool always uses the carrier density distribution computed in that particular interval, rather than the

Chapter 1: Simulation Procedure

Single-Particle Device Monte Carlo Simulation

density accumulated over multiple intervals. As a consequence of this approach, the number of Newton iterations needed to solve the Poisson equation and the initial residual of the Poisson equation do not systematically reduce further when the stationary state is reached.

If the keywords `CurrentErrorBar` and `MinCurrentComput` are assigned a value, the simulation can stop before the end of the maximum total simulation time. In this case, the simulation ends when the 'relative error' of the drain current is smaller than `CurrentErrorBar` and, at least, `MinCurrentComput` iterations (and, therefore, current computations) have been performed after the stationary state is reached.

Self-consistent simulations are activated by the keyword `SelfConsistent`. For stability reasons, the nonlinear Poisson equation must be used, which is specified by the option `FrozenQF` in parentheses following `SelfConsistent`.

Finally, surface roughness scattering is modeled in the tool by a combination of specular and diffusive scattering. The ratio of specular scattering is given by the keyword `SurfScattRatio`. The default is 85% specular scattering, that is, 15% diffusive scattering. The parameter `SurfScattRatio` is used to adjust the drain current of the Monte Carlo simulation to measurements (typically, if the gate voltage is equal to the supply voltage and, at a low drain voltage of, for example, 0.05 V or 0.1 V).

Note:

If the simulation interval for one iteration is too short to give a reasonable estimation of the density, then the accuracy of the simulation result is jeopardized. For example, this is the case if there are *holes* inside the inversion channel when viewing the file `mc_new_000001_des.tdr`. In general, the simulation time for one interval depends on the device, and the figure must be established empirically. Several time intervals must be tested to ensure that the chosen time interval is long enough. Of course, the aim is to select a time interval as small as possible because this minimizes the time for reaching the stationary state.

The number of iterations required to reach the stationary state is also a figure to be established empirically. However, this is not critical since the average value of the current always converges versus the true value for a long-enough simulation time.

For too few iterations, some 'nonstationary' values are considered in the averaging procedure, therefore, increasing the simulation time for reaching a good current estimation. For too many iterations, some 'stationary' values are omitted in the averaging procedure, which also increases the simulation time for reaching a good current estimation. In addition, for a 'reasonable error estimation,' averaging must not begin before the stationary state is reached (see [Estimating Currents on page 61](#)).

In the `Solve` section, the drift-diffusion solution is recomputed for reasons of consistency. The save file that is loaded contains only the electrostatic potential, carrier densities, and lattice temperature. Quantities such as the carrier velocities are computed from these basic variables only.

Chapter 1: Simulation Procedure

Single-Particle Device Monte Carlo Simulation

In the Plot section, the quantities that are computed by the Monte Carlo simulation and can be visualized with Sentaurus Visual are defined.

Screen Output of Simulation

The Monte Carlo simulation is run by typing:

```
sdevice mc_new
```

Sentaurus Device produces detailed output during each simulation about the specifications chosen and the convergence properties of the run. Some specifications from the MonteCarlo section can be found at the beginning of the output file.

The number of window elements and boundary elements in which carriers are injected are shown immediately before the Monte Carlo simulation begins:

```
=====
Starting solve of next problem:
MonteCarlo
=====
Number of window elements : 8063
Number of boundary elements : 66
*****
```

Then, the tool starts to read the input files, such as band-structure information. This can take some time because the band structure table is large. After some calculations regarding, for example, the scattering rates, some band structure data is printed that is needed, for example, as input for Sentaurus Device simulations and that changes under stress and for modified surface orientation:

```
Effective DOS: electron hole 2.872067300E+19 2.061158400E+19
Band-gap electronegativity (unstrained Si: 1.120 4.05) 1.120 4.050
electron: m_quant m_dos gamma 0.892 1.096 3.684
hole: m_quant m_dos gamma 0.266 0.878 9.897
eminx, eminy, eminz 0.000 0.000 0.000
emqx, emqy, emqz 0.196 0.196 0.892
hmin0, hmin1, hmin2 0.043 0.000 0.000
hmq0, hmq1, hmq2 0.233 0.212 0.266
```

Then, the main simulation parameters are printed:

```
Now: Number of electrons = 1
Now: Number of holes = 0
done.
Simulation time per frozen-field iteration = 4.000000000000000E-007
Simulation time until stationary state = 4.000000000000000E-006
Maximum total simulation time = 4.000000000000000E-005
```

Chapter 1: Simulation Procedure

Single-Particle Device Monte Carlo Simulation

The tool calculates whether to simulate an electron or a hole based on their total charge in the Monte Carlo window. Since an NMOSFET is under consideration, electrons are more numerous and, therefore, an electron is simulated.

After some checks, the main Monte Carlo routine is entered:

```
MC time: 0.0000e+00s: Writing plot 'mc_new_000000_des.tdr'... done.  
done.  
Entering propagation routine  
  
Total simulation time (micro sec) = 7.177531000056912E-002  
Number of particles propagated = 100000  
Mean energy of injected particles (eV) = 5.2868258E-02  
Mean propagation time per trajectory (ps) = 0.717753097065775  
  
Total simulation time (micro sec) = 0.144085571838629  
Number of particles propagated = 200000  
Mean energy of injected particles (eV) = 5.2735962E-02  
Mean propagation time per trajectory (ps) = 0.720427856242271  
  
Total simulation time (micro sec) = 0.214427214289264  
Number of particles propagated = 300000  
Mean energy of injected particles (eV) = 5.2720539E-02  
Mean propagation time per trajectory (ps) = 0.714757378036568  
  
Total simulation time (micro sec) = 0.285396316309597  
Number of particles propagated = 400000  
Mean energy of injected particles (eV) = 5.2756481E-02  
Mean propagation time per trajectory (ps) = 0.713490787851533  
  
Total simulation time (micro sec) = 0.355100399215986  
Number of particles propagated = 500000  
Mean energy of injected particles (eV) = 5.2747753E-02  
Mean propagation time per trajectory (ps) = 0.710200795522990  
Cumulative simulation time: .4000000E+06 psec, Particle number: 1  
Leaving propagation routine  
Number of propagated particles = 561614  
Simulation time (micro sec) = 0.4000000000000000
```

After the first iteration is completed, as shown above, the nonlinear Poisson equation is solved:

```
Computing poisson-equation  
using Bank/Rose nonlinear solver.
```

Iteration	Rhs	factor	step	error	#inner	#iterative	time

0	1.97e+01						
1	2.69e+01	1.00e+00	2.28e-01	1.62e+02	0	1	0.25
2	5.68e+00	1.00e+00	4.69e-02	3.38e+01	0	1	0.40
3	4.36e-01	1.00e+00	1.04e-02	7.18e+00	0	1	0.57
4	3.04e-03	1.00e+00	6.62e-04	4.31e-01	0	1	0.73
Finished, because...							

Chapter 1: Simulation Procedure

Single-Particle Device Monte Carlo Simulation

```
Error smaller than 1 ( 0.430914 ).  
  
Accumulated (wallclock) times:  
Rhs time:      0.07 s  
Jacobian time: 0.08 s  
Solve time:    0.53 s  
Total time:    0.78 s  
  
gamma (impurity scattering) = 1.0569807E+15  
Completion status: 1.0000 %  
Writing Monte Carlo output files.  
Preparing data sets for visualization.  
  
Computing test functions ... done.  
contact   voltage   electron current   hole current   conduction current  
source     0.000e+00   -3.574e-04       6.020e-89      -3.574e-04  
drain      1.200e+00   3.603e-04       1.098e-89      3.603e-04  
gate       1.200e+00   0.000e+00       0.000e+00      0.000e+00  
bulk       0.000e+00   -2.882e-06      -7.118e-89     -2.882e-06  
  
Integrated generation rates:   window   whole device  
avalanche [A]        3.375e-10   0.000e+00  
total G-R [A]        3.375e-10   0.000e+00  
  
MC time: 4.0000e-07s: Writing plot 'mc_new_000001_des.tdr'... done.  
done.  
Entering propagation routine
```

Then, the first plot file `mc_new_000001_des.tdr` is generated and the propagation routine is entered for the second iteration. This entire procedure is repeated until the end of the simulation.

Results of Simulation

Plot files for visualizing internal variables, such as density, electron temperature, and velocity, are stored in the files `mc_new_000024_des.tdr` and so on, and can be viewed using Sentaurus Visual.

The files with the simulation results for the currents have the suffixes `_time.plt` and `_average.plt`. They contain estimates of the current at the contact that was specified by the keyword `DrainContact` (called `MCdrain` in the `.plt` file) as well as the integral of the impact ionization rate over the entire Monte Carlo window (called `MCsubstrate`) (see [Equation 17 on page 61](#)).

In the file with the suffix `_time.plt`, the current estimates, which correspond only to one iteration interval, are stored as a function of the simulation time. Here, it is possible to deduce the time after which the simulation has reached the stationary state and the current begins to fluctuate around its average value.

Chapter 1: Simulation Procedure

Ramping of Boundary Conditions

In contrast, the file with the suffix `_average.plt` contains the cumulative averages over the current values stored in the file with the suffix `_time.plt`. They are plotted as a function of the number of iterations after reaching the stationary state. From this construction, it follows that the fluctuations of the cumulative averages diminish over the course of the simulation time. These cumulative averages represent the final simulation result. In addition, the ‘relative errors’ of these averages can be extracted from the file with the suffix `_average.plt`.

For details about the definition of these quantities, see [Averages and Statistical Error of the Currents on page 62](#).

However, these errors only represent a reasonable criterion for stopping the simulation when the gathering of statistics begins after reaching the stationary state (see [Estimating Currents on page 61](#)). The results contained in the files with the suffixes `_time.plt` and `_average.plt` can be viewed by using the visualization tool Inspect.

Ramping of Boundary Conditions

Instead of running one Sentaurus Device Monte Carlo process per bias point, you can combine calculations for multiple bias points in a single Sentaurus Device Monte Carlo run. In conjunction with parallelization (see [Parallelization on page 33](#)), this provides a simple way to obtain Monte Carlo-based I–V curves with minimal effort. For illustration, you will add the calculation of an I_d – V_d curve to the MCpFinFET example in the Applications Library. This example can be found in the `$STROOT/tcad/$STRELEASE/Applications_Library/GettingStarted/sdevice/MCpFinFET` directory.

This project contains a number of Monte Carlo nodes, each of which calculates data for a single bias point. Ramping the drain voltage can be switched on in the command file of the Sentaurus Device instance called `SPMC`:

1. Change the `Solve` section to:

```
Solve {
    Coupled {poisson hole}
    NewCurrentPrefix="n@node @_MCRamping_"
        # start new plot file for MC ramp
    QuasiStationary(
        Goal{name="Drain" voltage=-0.05} maxstep=0.077 doZero) {
            Plugin(iterations=0 breakOnFailure) {
                Coupled {poisson hole} #total charge for MC charge scaling
                Save(filePrefix="n@node @_beforeMC")
                    # write DD results to file
                Montecarlo # perform Monte Carlo simulation at bias point
                # Plot statements for plotting MC results go here.
                Load(filePrefix="n@node @_beforeMC") # restore DD results
            }
        }
}
```

Chapter 1: Simulation Procedure

Ramping of Boundary Conditions

2. Add the following line to the MonteCarlo section:

```
-InternalCurrentPlot
```

This deactivates writing of data points from the internal Boltzmann–Poisson iteration of the self-consistent Monte Carlo solver. Without `-InternalCurrentPlot`, the current plot file would contain a cloud of individual Monte Carlo current samples in addition to the converged Monte Carlo results.

The new `Solve` section uses a `Quasistationary` statement to ramp the gate bias. At each bias point, a plug-in loop is evaluated. In the options section of the `PlugIn` statement, `iterations=0` means that each entry of the plug-in loop is evaluated exactly once (no convergence checking), and `breakOnFailure` aborts the plug-in loop if one of its equations fails to converge.

The first entry in the body of the plug-in loop requests a coupled solution of the Poisson equation and the current continuity equation for holes. This prepares the initial solution for the `MonteCarlo` simulation that forms the third entry of the plug-in loop. Before calling `MonteCarlo`, the simulator status is written to a `.sav` file. At the end of the `Plugin` loop, this file is reloaded into the simulator. This procedure serves to eliminate convergence issues that otherwise could arise from the fact that a Monte Carlo simulation is not necessarily a good starting point for a drift-diffusion simulation with the inherent assumption of local equilibrium.

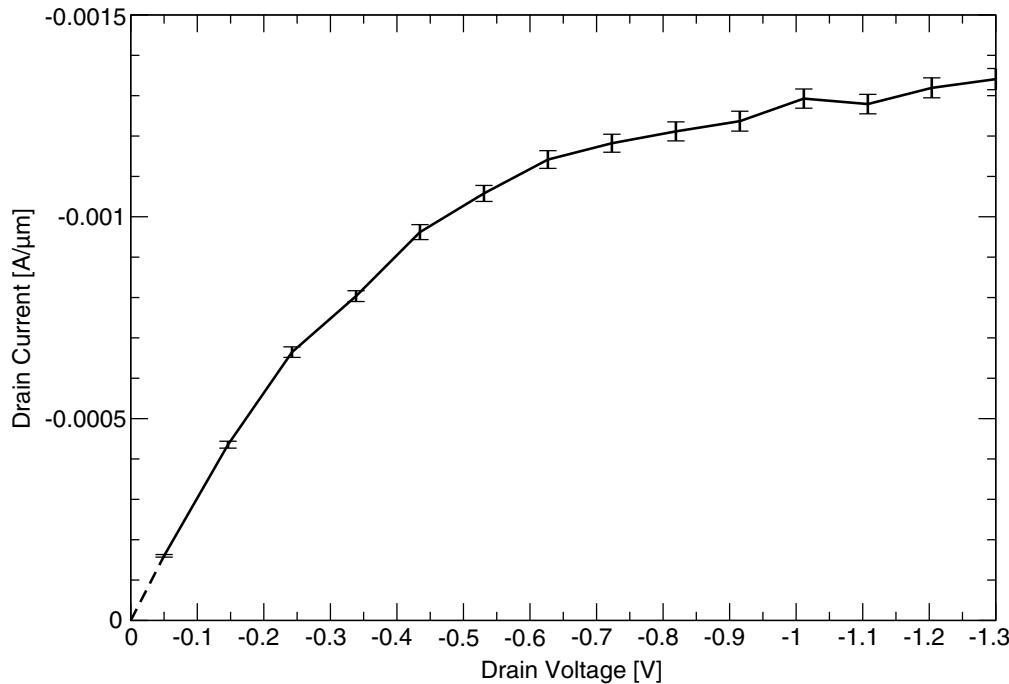
The `NewCurrentPrefix` statement sets the name of the current plot output file for the I_d – V_d curve. For each electrode, the Monte Carlo current and its 2σ absolute error are stored under the names `MCTotalCurrent` and `MCTotalCurrentError`, respectively. In addition, there are `TotalCurrent` values, which are the currents obtained from the drift-diffusion (`Coupled {poisson hole}`) simulation at the end of the plug-in loop. The I_d – V_d curve from the Monte Carlo simulation (with error bars obtained by adding or subtracting `MCTotalCurrent` and `MCTotalCurrentError`) is shown in [Figure 1](#).

Note that no Monte Carlo simulation is performed at zero drain bias, that is, the current is zero, of course, but the Monte Carlo simulation is ill suited to produce this result. The relative error criterion `CurrentErrorBar` becomes meaningless; the simulation runs all the way to the maximum-allowed time (the upper bound of the `Plot` range in the `MonteCarlo` section); and the final result is pure Monte Carlo noise.

Chapter 1: Simulation Procedure

Ramping of Boundary Conditions

Figure 1 I_d - V_d curve from Monte Carlo simulation



Note:

Single-particle device Monte Carlo always takes the total number of charge carriers from the previous simulation result. Therefore, the Quasistationary ramping statement must include a statement for calculating an appropriate starting solution. Putting only the MonteCarlo section inside the ramping statement never updates the total carrier number:

- Typically, the starting solution is produced using either `Coupled{poisson hole}` or `Coupled{poisson electron}`.
- Monte Carlo results far from equilibrium might be an inadequate starting solution for a drift-diffusion calculation (local equilibrium assumed). To avoid possible convergence issues, it is recommended to use Save/Load commands to restore solution variables to their pre-Monte Carlo values before performing the next bias step. Monte Carlo results can be plotted by inserting a `Plot` statement between the `MonteCarlo` and `Load` statements in the `Solve` section.
- For I_d - V_g curves, it is recommended to ramp down the gate voltage from the fully switched-on state towards the threshold voltage. In deep subthreshold, the accuracy of the Monte Carlo results can be improved using a fixed total sampling time by increasing the length of the individual Monte Carlo sampling intervals (that is, by reducing the number of intervals).
- Do not use the test function-based current evaluation (`useTestFunction`) for Monte Carlo currents in the subthreshold.

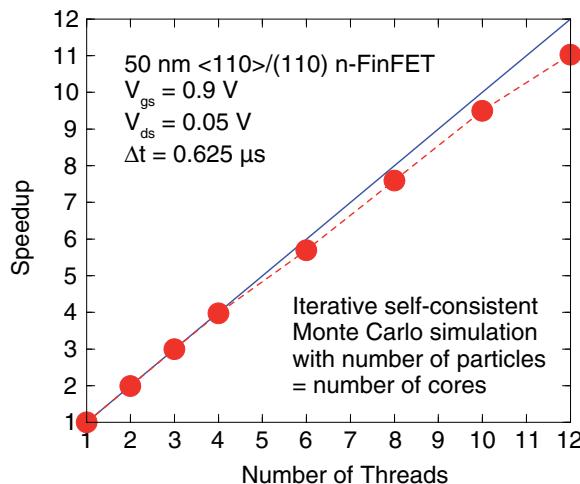
Parallelization

Single-particle device Monte Carlo is based on the propagation of independent particles through the device. Therefore, it is well suited for parallelization and makes good use of modern multicore computers.

Parallel Simulation Procedure

Parallelization of single-particle device Monte Carlo uses several threads of execution, each of which propagates its own particle through the device. Independent propagation of $\#threads$ particles for a time $T_{total}/\#threads$ leads to the same sample statistics as propagating a single particle for the full time T_{total} . If Monte Carlo sampling dominates the total execution time of a simulation, this leads to a near linear speedup (see [Figure 2](#)).

Figure 2 Parallel speedup of a typical Monte Carlo simulation



You activate parallel Monte Carlo simulation by specifying multiple solver threads in the `Math` section of the command file:

```
Math { NumberOfSolverThreads=integer }
```

The number of threads can be a fixed number (for example, `NumberOfSolverThreads=8`), or you can instruct Sentaurus Device to autodetect the number of CPU cores in your system and use one thread per core by specifying `NumberOfSolverThreads=maximum`.

Alternatively, you can use the keyword `NumberOfThreads`, but this has the side effect of setting not only the number of solver threads (the number of threads used by Monte Carlo), but also the number of assembly threads.

To take full advantage of parallelization, the wallclock time between subsequent Poisson updates must not be too small; otherwise, file writing and general bookkeeping can degrade

Chapter 1: Simulation Procedure

Parallelization

parallel performance. As a general rule, the sampling time between Poisson updates should be several seconds (wallclock time) or more. In addition, at the onset of a Monte Carlo simulation, certain setup tasks (for example, reading band-structure files) must be performed. In very short simulations, this can be a substantial part of the total runtime; therefore, very short Monte Carlo simulations will not see the full parallel speedup.

Parallel Simulations and Sentaurus Workbench

When scheduling parallel jobs using Sentaurus Workbench, be sure to properly configure the number of threads in the Tool Properties dialog box.

See *Sentaurus™ Workbench User Guide*, Configuring the Execution of Jobs, for more information.

Licensing

Parallel Sentaurus Device Monte Carlo is based on the Sentaurus Device parallel feature and uses the same licensing. For every four threads, you need one parallel license.

For details, see *Sentaurus™ Device User Guide*, Parallelization.

2

Input Specification

This chapter explains the Monte Carlo-specific parts of the command file of Sentaurus Device. For details about the command file, see the Sentaurus™ Device User Guide.

A typical input file for a Monte Carlo simulation is shown in [Chapter 1 on page 19](#). Changes must be made in the `File`, `Math`, `Solve`, and `Plot` sections and the `MonteCarlo` section itself. The next sections describe possible modifications.

File Section

File names for the simulation are specified in this section. Each keyword uses a predefined file extension. If the extension is omitted, it is appended automatically. The following keywords can be defined for Monte Carlo purposes:

- `MonteCarloOut` defines the prefix of the Monte Carlo output files for the current computations.
- `MonteCarloPath` defines the path of the directory where Sentaurus Device Monte Carlo finds the data files that describe input data such as the band structure. By default, `MonteCarloPath` points to an installation-specific directory that contains the data files for silicon.

Math Section

In addition to the mathematical models described in the *Sentaurus™ Device User Guide*, the keyword `currentweighting`, which is associated with terminal currents, is introduced in this section. This keyword defines the domain integration technique to be used in the evaluation of the drain current.

Parallelization of Monte Carlo simulations is controlled by the `NumberOfSolverThreads` keyword (see [Parallelization on page 33](#)).

Chapter 2: Input Specification

Solve Section

Solve Section

This section defines the equations that Sentaurus Device must solve and how Sentaurus Device solves them. You have great flexibility as to which equations are solved and the methods used (see the *Sentaurus™ Device User Guide*).

Monte Carlo Post-Solve

To calculate a Monte Carlo post-solve, `MonteCarlo` can be specified at any point in the `Solve` section like a partial differential equation. Of course, `MonteCarlo` cannot be coupled to any of the partial differential equations. However, it can be used in quasistationary simulations.

Plot Section

In the `Plot` section, you select the variables that are to be saved in the plot file. [Table 1](#) lists the Monte Carlo-specific keywords with the corresponding keywords in drift-diffusion simulations.

Table 1 Monte Carlo-specific keywords in Plot section

Keyword	Corresponding keyword in drift-diffusion simulations
<code>eMCavalanche</code> , <code>hMCavalanche</code>	The electron and hole parts of the averaged impact ionization rate that correspond to <code>eAvalanche</code> and <code>hAvalanche</code> .
<code>eMCCurrent</code> , <code>hMCCurrent</code>	The conduction current densities that correspond to <code>eCurrent</code> and <code>hCurrent</code> .
<code>eMCCurrentBackward</code> , <code>hMCCurrentBackward</code>	The conduction current densities of backward-flying carriers, that is, having a negative scalar product between group velocity and <code>InjectionDirection</code> .
<code>eMCCurrentForward</code> , <code>hMCCurrentForward</code>	The conduction current densities of forward-flying carriers, that is, having a positive scalar product between group velocity and <code>InjectionDirection</code> .
<code>eMCDensity</code> , <code>hMCDensity</code>	The carrier densities that correspond to <code>eDensity</code> and <code>hDensity</code> .
<code>eMCDensityBackward</code> , <code>hMCDensityBackward</code>	The carrier densities of backward-flying carriers, that is, having a negative scalar product between group velocity and <code>InjectionDirection</code> .

Chapter 2: Input Specification

Plot Section

Table 1 Monte Carlo-specific keywords in Plot section (Continued)

Keyword	Corresponding keyword in drift-diffusion simulations
eMCDensityForward, hMCDensityForward	The carrier densities of forward-flying carriers, that is, having a positive scalar product between group velocity and <code>InjectionDirection</code> .
eMCenergy, hMCenergy	The average carrier energies that correspond, in the case of hydrodynamic simulations, to <code>eTemperature</code> and <code>hTemperature</code> .
eMCValleyDeltaX	Occupation of the Δ -valley along the k_x -axis in the crystallographic coordinate system.
eMCValleyDeltaY	Occupation of the Δ -valley along the k_y -axis in the crystallographic coordinate system.
eMCValleyDeltaZ	Occupation of the Δ -valley along the k_z -axis in the crystallographic coordinate system.
eMCValleyGamma	Occupation of the Γ -valley.
eMCValleyLMinusMinus	Occupation of the L-valley pair along the $(-1,-1,1)$ direction in the Brillouin zone.
eMCValleyLMinusPlus	Occupation of the L-valley pair along the $(-1,1,1)$ direction in the Brillouin zone.
eMCValleyLPlusMinus	Occupation of the L-valley pair along the $(1,-1,1)$ direction in the Brillouin zone.
eMCValleyLPlusPlus	Occupation of the L-valley pair along the $(1,1,1)$ direction in the Brillouin zone.
eMCVelocity, hMCVelocity	The carrier velocities that correspond to <code>eVelocity</code> and <code>hVelocity</code> .
eMCVelocityBackward, hMCVelocityBackward	The carrier velocities of backward-flying carriers, that is, having a negative scalar product between group velocity and <code>InjectionDirection</code> .
eMCVelocityForward, hMCVelocityForward	The carrier velocities of forward-flying carriers, that is, having a positive scalar product between group velocity and <code>InjectionDirection</code> .
hMCBandHeavyHole	Occupation of the heavy-hole band.
hMCBandLightHole	Occupation of the light-hole band.

Chapter 2: Input Specification

MonteCarlo Section

Table 1 Monte Carlo-specific keywords in Plot section (Continued)

Keyword	Corresponding keyword in drift-diffusion simulations
hMCBandSplitOff	Occupation of the split-off band.
MCFIELD	The driving field that corresponds to GradConductionBand (when electrons are simulated) or GradValenceBand (when holes are simulated).

MonteCarlo Section

The parameters for the interface to the Monte Carlo simulation are defined in this section.

Note:

Some quantities such as the band gap are taken from the parameter file.

Therefore, for a Monte Carlo simulation of a strained silicon device, specify in the parameter file the correct values that correspond to the strain level considered.

Table 2 Parameter keywords in MonteCarlo section

Keyword	Explanation
AlloyFactor = float	Prefactor of the scattering rate for alloy scattering. It is used to switch off alloy scattering by setting AlloyFactor=0.0 or to adjust the hole mobility in SiGe. Default: 1.0
BetaExponentialSurfaceRoughness = float	Exponential in the exponential power spectrum (unitless). Default: 1.0
CarrierType = autodetect electrons holes	Selects the carrier type used for Monte Carlo propagation. Supported values are: <ul style="list-style-type: none">• autodetect (default): The carrier type is determined by integration of the electron and hole densities over the volume of the Monte Carlo window. Whichever carrier type is present in the Monte Carlo window in larger numbers is selected for propagation.• electrons: Propagate electrons.• holes: Propagate holes.
ChannelDirection = (integer,integer,integer)	This specifies the crystallographic orientation of the channel using a three-dimensional vector for the crystallographic direction. Default: ChannelDirection = (1 1 0)

Chapter 2: Input Specification

MonteCarlo Section

Table 2 Parameter keywords in MonteCarlo section (Continued)

Keyword	Explanation
CorrelationLength = float	Correlation length for normal field-dependent surface roughness scattering (in nm). Default: 1.49
CreateDOSFile = integer	This writes a file containing the density-of-states of the Monte Carlo band structure if CreateDOSFile=1 is specified. The default is CreateDOSFile=0, that is, no file is written.
CrystAngle = float	This specifies the crystallographic orientation of the channel by giving the angle (in degrees) by which the channel direction is rotated around the (001) substrate orientation with respect to CrystDirection=100 (see the keyword above). For example, CrystAngle=45 corresponds to CrystDirection=110, but other values allow for different channel orientations around the (001) substrate direction.
CrystDirection = integer	This specifies the crystallographic orientation of the channel, that is, the crystallographic direction parallel to the interface between silicon and SiO ₂ , which essentially corresponds to the direction of the current flow in the channel. The default is CrystDirection=110, which is the direction used in current technology. Alternatively, CrystDirection=100 can be specified. In 2D device simulation, the specified crystallographic direction is associated with the x-axis, which usually corresponds to the channel direction in a MOSFET. Note that this direction does <i>not</i> correspond to the substrate orientation, that is, the crystallographic direction perpendicular to the interface between silicon and SiO ₂ , which is always in the (001) direction.
CurrentErrorBar = float	This specifies the 'relative error' (in percent) of the drain current, below which the simulation is stopped (if the number of iterations after steady state is, at the same time, greater than MinCurrentComput). If absent, the simulation runs until the end of the specified, maximum simulation time.
DopingtoSelectSDforFermi = float	This selects all elements in the device where doping is equal to or higher than the specified value. In these elements, the maximum carrier density is set to the doping density for the computation of the Fermi energy.

Chapter 2: Input Specification

MonteCarlo Section

Table 2 Parameter keywords in MonteCarlo section (Continued)

Keyword	Explanation
DrainContact = <i>string</i> or: DrainContact = <i>integer</i>	This selects the contact for which current is computed and to which the CurrentErrorBar criterion is applied (unless WithSubstrateError is specified). It is recommended to specify the contact by name (for example, DrainContact="drain"). Alternatively, the contact can be specified by its integer contact number: Starting with zero, the contact number counts the contacts in the order of their regions in the (.tdr) file and can be queried using Sentaurus Data Explorer (tdx -info).
EnergyDistributionPosition1 = <i>vector</i>	This specifies that the energy distribution for the specified position vector must be stored in a PLT file with the suffix _endist.plt.
EnergyDistributionPosition2 = <i>vector</i>	This specifies that the energy distribution for the specified position vector must be stored in a PLT file with the suffix _endist.plt.
EnergyDistributionPosition3 = <i>vector</i>	This specifies that the energy distribution for the specified position vector must be stored in a PLT file with the suffix _endist.plt.
EnergyDistributionPosition4 = <i>vector</i>	This specifies that the energy distribution for the specified position vector must be stored in a PLT file with the suffix _endist.plt.
EnergyDistributionPosition5 = <i>vector</i>	This specifies that the energy distribution for the specified position vector must be stored in a PLT file with the suffix _endist.plt.
EnergyDistributionPosition6 = <i>vector</i>	This specifies that the energy distribution for the specified position vector must be stored in a PLT file with the suffix _endist.plt.
EnergyDistributionPosition7 = <i>vector</i>	This specifies that the energy distribution for the specified position vector must be stored in a PLT file with the suffix _endist.plt.
EnergyDistributionPosition8 = <i>vector</i>	This specifies that the energy distribution for the specified position vector must be stored in a PLT file with the suffix _endist.plt.

Chapter 2: Input Specification

MonteCarlo Section

Table 2 Parameter keywords in MonteCarlo section (Continued)

Keyword	Explanation
EnergyDistributionPosition9 = vector	This specifies that the energy distribution for the specified position vector must be stored in a PLT file with the suffix <code>_endist.plt</code> .
FinalTime = float	This is the simulation time after which the steady state is assumed to be reached. The gathering of cumulative averages begins only after <code>FinalTime</code> .
HighKEpsilonInf = float	Optical dielectric constant (in ϵ_0) of the high- κ gate oxide. The default is 5.03, which corresponds to HfO ₂ .
HighKEpsilonInt = float	Intermediate dielectric constant (in ϵ_0) of the high- κ gate oxide. The default is 6.58, which corresponds to HfO ₂ .
HighKEpsilonZero = float	Static dielectric constant (in ϵ_0) of the high- κ gate oxide. The default is 22.0, which corresponds to HfO ₂ .
HighKModus = integer	This activates soft-optical phonon scattering. Use <code>HighKModus=2</code> if an interfacial oxide is present between the high- κ gate oxide and the semiconductor channel. Use <code>HighKModus=1</code> in the absence of an interfacial oxide; it considers two instead of one transverse-optical phonon modes of the high- κ gate oxide. The default is <code>HighKModus=0</code> , that is, soft-optical phonon scattering is switched off.
HOmegaTO1 = float	Lowest-energy transverse-optical phonon energy (in meV) of the high- κ gate oxide. The default is 12.40, which corresponds to HfO ₂ .
HOmegaTO2 = float	Second lowest-energy transverse-optical phonon energy (in meV) of the high- κ gate oxide. The default is 48.35, which corresponds to HfO ₂ .
IIFactor = float	Prefactor of the scattering rate for impact ionization. It is used to switch off impact ionization by setting <code>IIFactor=0.0</code> or to adjust, for example, the substrate current. Default: 1.0
ImpFactor = float	Prefactor of the scattering rate for impurity scattering. It is used to switch off impurity scattering by setting <code>ImpFactor=0.0</code> . Default: 1.0

Chapter 2: Input Specification

MonteCarlo Section

Table 2 Parameter keywords in MonteCarlo section (Continued)

Keyword	Explanation
<code>ImpWindowtoSelectSDforFermi</code> <code>-ImpWindowtoSelectSDforFermi</code>	This specifies that, inside <code>WindowImpFactor1</code> and <code>WindowImpFactor2</code> , the maximum carrier density is set to the doping density for the computation of the Fermi energy. Default: <code>-ImpWindowtoSelectSDforFermi</code>
<code>InjectionDirection = (float,float,float)</code>	This activates the gathering of separate statistics for forward- and backward-flying carriers. At the same time, this specifies a direction vector in device coordinates, which is used to determine whether a carrier is forward flying or backward flying. If the scalar product between <code>InjectionDirection</code> and the group velocity of the carrier is positive, the carrier is considered to be forward flying; in the opposite case, the carrier is considered to be backward flying.
<code>InternalCurrentPlot</code> <code>-InternalCurrentPlot</code>	This controls whether data points for the current plot file are added for each internal iteration of the self-consistency procedure. By default, the <code>InternalCurrentPlot</code> flag is switched on. To include only the final results of the self-consistency iteration (for example, for bias ramping), the flag can be deactivated by using <code>-InternalCurrentPlot</code> .
<code>Intervals = integer</code>	This specifies the number of intervals at which plot files are to be written within the given range. <code>end</code> divided by <code>Intervals</code> also determines the simulation time for each frozen-field iteration.
<code>IsInGaAs</code>	This specifies that InGaAs is simulated instead of the default material SiGe.
<code>KVecEnd = (float,float,float)</code>	The end wavevector for visualization of the analytic band energies (activated using <code>MCStrain</code>) along a line from <code>KVecStart</code> .
<code>KVecStart = (float,float,float)</code>	The start wavevector for visualization of the analytic band energies (activated using <code>MCStrain</code>) along a line until <code>KVecEnd</code> .
<code>MasettiCalibration</code> <code>-MasettiCalibration</code>	This specifies that a doping-dependent calibration to the Masetti mobility measurements is used (this is the default). For deactivation, that is, to use uncalibrated impurity scattering, specify <code>-MasettiCalibration</code> .

Chapter 2: Input Specification

MonteCarlo Section

Table 2 Parameter keywords in MonteCarlo section (Continued)

Keyword	Explanation
<code>MCl_crit = float</code>	This specifies the decay length in the exponential prefactor of the surface roughness scattering rate (in cm). The default is 100.1, that is, there is no effect.
<code>MCStrain= (float,float,float, float,float,float)</code>	Components of the symmetric strain tensor, which determine the analytic band structures used in the simulation. The values ϵ_{xx} , ϵ_{yy} , ϵ_{zz} , ϵ_{yx} , ϵ_{xz} , ϵ_{xy} must be given in the coordinate system where the three axes are aligned with the principal axes of the three pairs of ellipsoids in the first conduction band.
<code>MinCurrentComput = float</code>	This specifies the minimum number of iterations after steady state that are performed irrespective of whether the ‘relative error’ of the drain current is less than <code>CurrentErrorBar</code> .
<code>NintHighK = float</code>	Density of interface charges between the high- κ dielectric and the interfacial oxide (in cm^{-2}), entering the expression for the inverse microscopic relaxation time of remote Coulomb scattering (RCS). Default: 0.0
<code>Normal2OxideDirection = (integer,integer,integer)</code>	This specifies the crystallographic surface orientation, that is, the direction perpendicular to the interface between the gate oxide and channel, using a three-dimensional vector for the crystallographic direction. In bulk MOSFETs, this corresponds to the substrate orientation. The default is <code>Normal2OxideDirection = (0 0 1)</code> . <code>Normal2OxideDirection</code> must be perpendicular to <code>ChannelDirection</code> .
<code>OxideEpsilonInf = float</code>	Optical dielectric constant (in ϵ_0) of the interfacial oxide. The default is 2.50, which corresponds to SiO_2 .
<code>OxideEpsilonInt = float</code>	Intermediate dielectric constant (in ϵ_0) of the interfacial oxide. The default is 3.05, which corresponds to SiO_2 .
<code>OxideEpsilonZero = float</code>	Static dielectric constant (in ϵ_0) of the interfacial oxide. The default is 3.90, which corresponds to SiO_2 .
<code>OxideHOmegaT01 = float</code>	Lowest-energy transverse-optical phonon energy (in meV) of the interfacial oxide. The default is 55.60, which corresponds to SiO_2 .

Chapter 2: Input Specification

MonteCarlo Section

Table 2 Parameter keywords in MonteCarlo section (Continued)

Keyword	Explanation
PhosphorousCalibration -PhosphorousCalibration	This activates a doping-dependent calibration to the Masetti mobility measurements for phosphorous instead of arsenic. Default: -PhosphorousCalibration
Plot {range intervals}	This specifies the times at which plot files are written and Poisson updates are triggered (in self-consistent mode).
PlotValleyOccupations -PlotValleyOccupations	This specifies that the valley or valence-band occupations for the valleys or valence bands given in the Plot section must be plotted. Default: -PlotValleyOccupations
Range = (start, end)	This specifies Range, where start and end are float values that denote the start time and the maximum simulation time. In the tool, start must be zero.
RCSHighKEpsilon = float	Dielectric constant (in ϵ_0) of the high-k dielectric used in the expression for the inverse microscopic relaxation time of RCS. The default is 22.0, which corresponds to HfO ₂ .
RoughnessMeanSquare = float	Root mean square (RMS) amplitude for normal field-dependent surface roughness scattering (in nm). Default: 0.3
SelfConsistent (frozenParams)	This defines the Monte Carlo simulation as self-consistent. If SelfConsistent is not defined, the frozen field of the initial Sentaurus Device simulation is used throughout the simulation. <i>frozenParams</i> defines which parameters are frozen during the Poisson solve. For stability reasons, this must be the quasi-Fermi potentials (<i>frozenParams</i> is equal to FrozenQuasiFermi) in the case of Sentaurus Device Monte Carlo. That is, the nonlinear Poisson equation is solved.
StatRatioa = float	The ratio of the time that the particle spends inside Windowa with respect to the total simulated time. In the screen output, it is reported in terms of how many copies a particle is split into when entering Windowa to achieve the specified ratio. Default: 0.5

Chapter 2: Input Specification

MonteCarlo Section

Table 2 Parameter keywords in MonteCarlo section (Continued)

Keyword	Explanation
<code>SubstrateContact = string</code>	The Monte Carlo simulator can estimate the substrate current by integrating the avalanche generation rate over the volume of the Monte Carlo window. If no <code>SubstrateContact</code> is specified, this current is not included in the standard Sentaurus Device current plot file. Specifying a contact name (for example, <code>SubstrateContact="bulk"</code>) causes this current to be associated with the selected contact in the current plot file.
<code>SurfaceRoughnessModel = integer</code>	This activates a normal field-dependent surface roughness scattering rate: <ul style="list-style-type: none">• Use <code>SurfaceRoughnessModel=1</code> for a Gaussian power spectrum.• Use <code>SurfaceRoughnessModel=2</code> for an exponential power spectrum.• Use <code>SurfaceRoughnessModel=3</code> for the Pirovano power spectrum. The default is <code>SurfaceRoughnessModel=0</code> , that is normal field-dependent surface roughness scattering is switched off.
<code>SurfScattRatio = float</code>	This defines the ratio between specular and diffusive scattering at SiO ₂ interfaces. A value of 1 corresponds to pure specular scattering. Default: 0.85
<code>SurfScattRatioBallistic = float</code>	This defines the ratio between specular and diffusive scattering at SiO ₂ interfaces inside <code>WindowBallistic</code> . A value of 1 corresponds to pure specular scattering. Default: 1
<code>useTestFunction</code> <code>-useTestFunction</code>	This controls whether terminal currents are evaluated by direct particle counting (<code>-useTestFunction</code> ; this is the default) or by the test function method (<code>useTestFunction</code>). Direct particle counting is the more reliable approach in the subthreshold regime. Monte Carlo simulations with contact resistance must use the direct particle counting method (<code>-useTestFunction</code>).

Chapter 2: Input Specification

MonteCarlo Section

Table 2 Parameter keywords in MonteCarlo section (Continued)

Keyword	Explanation
Window <i>shape</i> [vector vector]	In Sentaurus Device Monte Carlo simulations, <i>shape</i> can be one of the following: <ul style="list-style-type: none">• Line in one dimension• Rectangle in two dimensions• Cuboid in three dimensions The two vectors define the corners of the window. The Monte Carlo simulation is performed in all elements that lie completely inside the defined window.
Windowa <i>shape</i> [vector vector]	In Sentaurus Device Monte Carlo simulations, <i>shape</i> is Rectangle. The two vectors define the corners of the rectangle. Gathering of statistics is enhanced inside Windowa.
WindowBallistic <i>shape</i> [vector vector]	In Sentaurus Device Monte Carlo simulations, <i>shape</i> is Rectangle. The two vectors define the corners of the rectangle. All scattering mechanisms are switched off, and the ratio of specular scattering upon surface scattering set to 1 (that is, the ratio of diffusive scattering is 0) in all elements that lie completely inside the defined rectangle.
WindowImpFactor1 <i>shape</i> [vector vector]	In Sentaurus Device Monte Carlo simulations, <i>shape</i> can be one of the following: <ul style="list-style-type: none">• Line in one dimension• Rectangle in two dimensions• Cuboid in three dimensions Inside the shape, the maximum carrier density is set to the doping density for the computation of the Fermi energy, if the keyword ImpWindowtoSelectSDforFermi is true.
WindowImpFactor2 <i>shape</i> [vector vector]	In Sentaurus Device Monte Carlo simulations, <i>shape</i> can be one of the following: <ul style="list-style-type: none">• Line in one dimension• Rectangle in two dimensions• Cuboid in three dimensions Inside the shape, the maximum carrier density is set to the doping density for the computation of the Fermi energy, if the keyword ImpWindowtoSelectSDforFermi is true.

Chapter 2: Input Specification

MonteCarlo Section

Table 2 Parameter keywords in MonteCarlo section (Continued)

Keyword	Explanation
WithMCBrooksHerring -WithMCBrooksHerring	This specifies that the Brooks–Herring model (with calibration to the formula of Masetti) is used for ionized impurity scattering (this is the default). For deactivation, which means that the Ridley model (with calibration to the Caughey–Thomas formula) is used, specify -WithMCBrooksHerring.
WithFermiDiracScreening -WithFermiDiracScreening	This specifies that <i>not</i> a Boltzmann distribution but a heated Fermi distribution is used for the screening of impurity scattering (this is the default). For deactivation, that is, using a Boltzmann distribution for screening, specify -WithFermiDiracScreening.
WithLatticeTemperatureScreening -WithLatticeTemperatureScreening	This specifies that impurity scattering is screened by the lattice temperature (this is the default) instead of the carrier temperature. For deactivation, that is, to use the carrier temperature for screening, specify -WithLatticeTemperatureScreening.
WithMCConwellWeisskopf -WithMCConwellWeisskopf	This specifies that the Conwell–Weisskopf model (with calibration to the formula of Masetti) is used for ionized impurity scattering instead of the Brooks–Herring model. The default is -WithMCConwellWeisskopf.
WithStrainedMaterial	This specifies that the material Strained Silicon is simulated with the tool. It is necessary to specify the strain level through the germanium content, y_{Ge} , in the virtual SiGe substrate, or to give the value (in MPa) for a uniaxial stress, or to specify the germanium content, x_{Ge} , in the active p-type SiGe layer, which is grown on a silicon substrate.
WithStressAlpha	This specifies that the influence of stress on the nonparabolicity parameter in the analytic electron band model is considered (this is the default). For deactivation, specify -WithStressAlpha.
WithSubstrateError	This specifies that it is <i>not</i> the drain current, which is the default, but the substrate current that is used to stop the simulation with an error criterion. That is, the simulation will stop if the relative error of the substrate current is below the value specified by CurrentErrorBar. In the case of substrate currents, a typical value is CurrentErrorBar=20.0.

Chapter 2: Input Specification

Coordinate System

Table 2 Parameter keywords in MonteCarlo section (Continued)

Keyword	Explanation
<code>xGe = float</code>	Germanium content (in percent) in the active SiGe layer that defines the contribution of Ge phonon scattering and alloy scattering. The pseudopotential band-structure tables for (100)-Si _{0.7} Ge _{0.3} under biaxial compressive strain (specify <code>xGe=30.0</code> and <code>withStrainedMaterial</code>) and relaxed Ge (specify <code>xGe=100.0</code>) are included. Alternatively, arbitrary analytic structures can be used with <code>MCStrain</code> . If <code>IsInGaAs</code> is specified, <code>xGe</code> corresponds to the gallium content. The pseudopotential band-structure tables for relaxed InGaAs with gallium contents of 0%, 47%, and 100% are included.
<code>yGe = float</code>	Germanium content (in percent) in the virtual SiGe substrate that defines the strain level in the silicon layer grown on top of the SiGe substrate. For current technologies, the typical value is <code>yGe=20.0</code> . Other supported strain levels are <code>yGe=10.0</code> , <code>30.0</code> , and <code>40.0</code> . The band structure tables of these three additionally supported strain levels can be obtained from Synopsys on request (send an email to <code>support-tcad-eu@synopsys.com</code>) and must be placed in corresponding directories parallel to the existing directory <code>strain20</code> for <code>yGe=20.0</code> .

Coordinate System

In the absence of stress and when using the `ReadInStress` option, Sentaurus Device Monte Carlo can use both the unified coordinate system and the DF–ISE coordinate system in the same way as Sentaurus Device.

If the `MCStrain` option or a pseudopotential table is used, then the DF–ISE coordinate system with the x-axis pointing into the channel direction must be used.

3

Physical and Numeric Models

This chapter describes the underlying physical models and various parts of the Monte Carlo algorithm, emphasizing crucial aspects for computational performance. Other points are addressed briefly.

Models for Band Structure and Scattering Mechanisms

The full band structure for Si is obtained by nonlocal pseudopotential calculations [1] where, in addition, the spin-orbit interaction is taken into account [2]. Four conduction bands and three valence bands are stored on a mesh in momentum space, with an equidistant grid spacing of $1/96 \frac{2\pi}{a_0}$, where a_0 denotes the lattice constant. Within each cube, the band energy is expanded to linear order around the middle of the cube. Therefore, the group velocity is constant in each momentum-space element.

The scattering mechanisms comprise phonon scattering, impact ionization, impurity scattering, and surface roughness scattering. The phonon scattering model for electrons includes three g-type and three f-type intervalley processes [3], as well as inelastic intravalley scattering [4]. In the case of holes, optical phonon scattering and inelastic acoustic phonon scattering are considered [5]. At present, only impact ionization for electrons with the scattering rate taken from the literature [6] is considered. The comparison of the resulting velocity field characteristics at different lattice temperatures [4][5] with time-of-flight measurements [7][8][9][10] is shown in Figure 3 and Figure 4, respectively.

Impurity scattering is important in MOSFETs because of the highly doped source and drain contacts. Unfortunately, it is also computationally intensive due to high scattering rates at low energies, with almost no change in the momentum. This effect is particularly strong in the Brooks–Herring (BH) model, which describes the screened two-body interaction with one ionized impurity [11]. It is reduced in the Ridley (RI) statistical screening model, taking into account the probability that there is no closer scattering center [12].

The most significant reduction of the computational burden, however, is achieved by approximating the scattering rate by the inverse microscopic relaxation time, and selecting at random the state-after-scattering on the equi-energy surface. This is shown in Figure 5 where, for purposes of illustration, density and doping concentrations have been chosen so that this effect is particularly pronounced. Comprehensive investigations [13][14] have

Chapter 3: Physical and Numeric Models

Models for Band Structure and Scattering Mechanisms

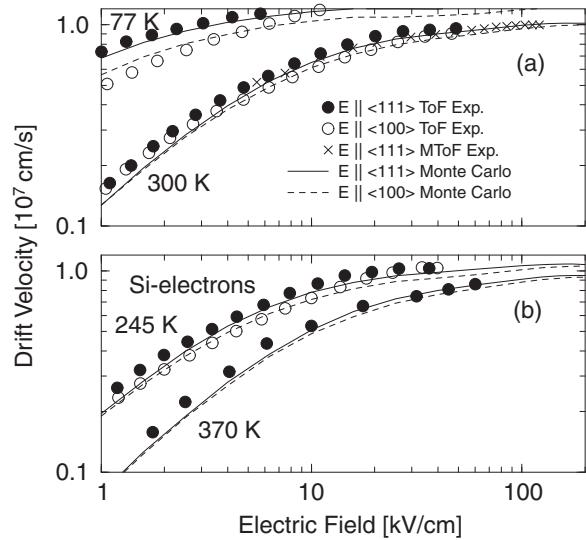
shown that at high fields there is also almost no difference between this and the exact treatment. Since impurity scattering is only important at low energies, an analytic, isotropic, and nonparabolic band structure is used for the calculation of the inverse microscopic relaxation time up to 1.0 eV, and it neglects impurity scattering for higher electron energies.

The inverse relaxation time is given by:

$$\frac{1}{\tau_{RI}(E)} = \frac{V}{(2\pi)^3} \int d^3 k' S_{RI}(\mathbf{k}'|\mathbf{k})(1 - \hat{\mathbf{k}}' \cdot \hat{\mathbf{k}}) \quad (1)$$

where V denotes the crystal volume, $S_{RI}(\mathbf{k}'|\mathbf{k})$ is the transition probability per unit time, and $\hat{\mathbf{k}} = \mathbf{k}/|\mathbf{k}|$.

Figure 3 Comparison of full band Monte Carlo results for velocity field characteristics of Si-electrons at different lattice temperatures with corresponding time-of-flight measurements (measurements from literature [7][8])



Chapter 3: Physical and Numeric Models

Models for Band Structure and Scattering Mechanisms

Figure 4 Comparison of full band Monte Carlo results for velocity field characteristics of Si-holes at different lattice temperatures with corresponding time-of-flight measurements (measurements from [7][9][10])

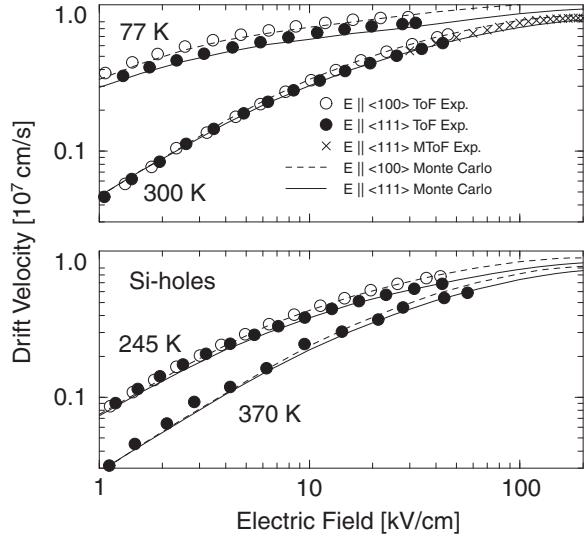
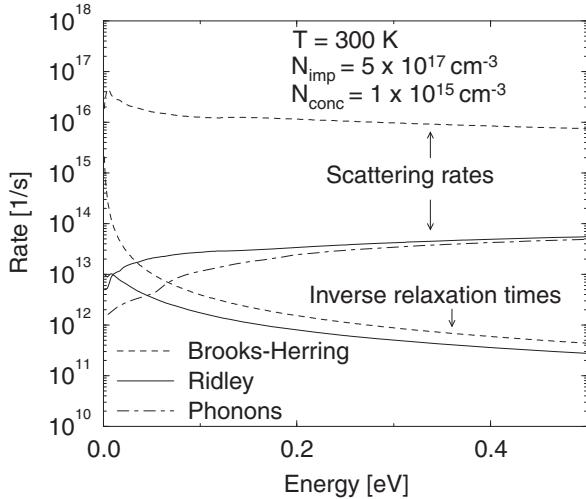


Figure 5 Scattering rates and inverse microscopic relaxation times of impurity scattering in formulation of Brooks–Herring and Ridley (phonon scattering rate is shown for comparison)



In the case of the Brooks–Herring (BH) model, the result is:

$$\frac{1}{\tau_{BH,n}(\varepsilon)} = \frac{\pi e^4 N_{imp}}{(4\pi\varepsilon\varepsilon_0)^2 \sqrt{2m_{d,n}}} \frac{1 + 2\alpha_n E_n}{(E_n(1 + \alpha_n E_n))^{3/2}} \Phi(\eta)\Theta(E_n) \quad (2)$$

Chapter 3: Physical and Numeric Models

Models for Band Structure and Scattering Mechanisms

with these abbreviations:

$$\begin{aligned}\Phi(\eta) &= \ln(1 + \eta) - \frac{\eta}{1 + \eta} \\ \eta &= \frac{8m_{d,n}E_n(1 + \alpha_n E_n)}{\hbar^2\beta^2} \\ E_n &= \varepsilon - \varepsilon_{0,n}\end{aligned}\tag{3}$$

where:

- e is the elementary charge.
- n is the valley index.
- N_{imp} is the impurity concentration.
- ε is the static dielectric constant of silicon.
- m_d is the density-of-states mass at the band edge.
- α is the nonparabolicity factor.
- $\beta = \sqrt{e^2 \left(\frac{dn}{d\mu} - \frac{dp}{d\mu} \right) / \varepsilon}$ is the inverse screening length.

Here, n is the electron density, p is the hole density, and μ is the Fermi energy. See [Fermi Statistics on page 54](#).

Since the Ohmic drift mobility with the above impurity scattering model significantly deviates from experimental results, especially for high doping concentrations, a doping-dependent prefactor is introduced in [Equation 2](#) to reproduce the mobility measurements of [15]. This approach to impurity scattering is heuristic, but it correctly and efficiently accounts for the mobility reduction in highly doped contact regions. It is activated by the `MasettiCalibration` keyword, which is true by default.

By default, the calibration is for arsenic doping in n-type silicon (boron doping in p-type silicon). For phosphorus doping in n-type silicon, `PhosphorousCalibration` must be specified, which is false by default.

If this doping-dependent prefactor calibrated to silicon mobilities must not be used, for example, for InGaAs, it can be switched off by specifying `-MasettiCalibration`. The effect of degeneracy is noticeable and most important for screening, even for applications where Fermi–Dirac statistics is not important otherwise [16]. Therefore, the inverse screening length is computed, by default, under nondegenerate conditions (that is, the corresponding keyword `WithFermiDiracScreening` is true by default), and the Masetti calibration of the Brooks–Herring model is performed only for this case.

Chapter 3: Physical and Numeric Models

Models for Band Structure and Scattering Mechanisms

If the keyword `withMCConwellWeisskopf` is specified, impurity scattering can also be simulated without screening using the Conwell–Weisskopf model. In this case, only the function Φ in [Equation 3](#) must be changed according to [\[17\]](#):

$$\Phi_{\text{CW}, n} = \ln \left(1 + \frac{16\pi^2 \varepsilon^2 E_n^2}{e^4 N_{\text{imp}}^{2/3}} \right) \quad (4)$$

In this case, the function Φ does not depend on the inverse screening length β .

In addition, the Conwell–Weisskopf model is calibrated by default to the Masetti mobilities in silicon, resulting in a different doping-dependent prefactor.

Note:

While the results shown were obtained with inelastic acoustic phonons, the actually employed phonon models use the elastic equipartition approximation for acoustic intravalley phonon scattering (with a coupling constant of 8.52 eV in the case of electrons; in the case of holes, all coupling constants are those of [\[18\]](#)). The differences in results are very small.

Surface Roughness Scattering

The following models for surface roughness scattering are available:

- A combination of specular and diffusive scattering to be applied to a classical density profile where the quantum correction is considered in terms of effective oxide thickness and workfunctions
- A scattering rate proportional to the square of the normal electric field to be applied to a quantum density profile as present, for example, in a density-gradient Monte Carlo simulation

The combination of specular and diffusive scattering is a semi-empirical treatment where at random either a specular or a diffusive scattering process is selected when an electron hits the interface to the oxide. The probability of diffusive scattering can be adjusted to measured long-channel effective mobilities. The default value of 15% diffusive scattering reproduces, with high accuracy [\[19\]](#), the measured FinFET mobility curves [\[20\]](#), for both electrons and holes, and for both (100) and (110) sidewall orientations. The orientation dependency of the effective mobility results from the energy- and parallel-momentum conservation of specular surface scattering (see the detailed discussion in [\[19\]](#)).

In the second approach, the perturbation potential is the component of the electric field normal to the gate interface [\[21\]](#). The corresponding transition rate per unit time from Fermi's Golden Rule reads:

$$S(\mathbf{k}'|\mathbf{k}) = \frac{2\pi}{\hbar} \left| \frac{dV}{dz} \right|^2 \frac{1}{A} S(|\mathbf{K}' - \mathbf{K}|) \delta_{k_x, k_z} \delta \left(\frac{\hbar^2 K'^2}{2m^*} - \frac{\hbar^2 K^2}{2m^*} \right) \quad (5)$$

Chapter 3: Physical and Numeric Models

Fermi Statistics

where, in analogy to the treatment of soft-optical phonon scattering [22], the scattering potential is treated parametrically for the 3D wavevector $\mathbf{k} = (\mathbf{K}, k_z)$ conserving the normal wavevector component k_z upon scattering. Here, V is the fluctuating confining potential, A is the unit area, and m^* is the effective mass.

The power spectrum of surface roughness $S(|\mathbf{K}' - \mathbf{K}|) = S(Q)$ is given by:

- $S(Q) = \pi L^2 \Delta^2 \exp\left(-\frac{L^2 Q^2}{4}\right)$ for the Gaussian model (SurfaceRoughnessModel=1)
- $S(Q) = \frac{\pi L^2 \Delta^2}{\left(1 + \frac{L^2 Q^2}{2}\right)^\beta}$ for the exponential model (SurfaceRoughnessModel=2)
- $S(Q) = \pi L^2 \Delta^2 \exp\left(-\frac{L^4 Q^4}{4}\right)$ for the Pirovano model (SurfaceRoughnessModel=3)

where:

- Δ (RoughnessMeanSquare in units of nm) is the roughness mean square amplitude.
- L (CorrelationLength in units of nm) is the correlation length.
- For the exponential model, the exponent β (BetaExponentialSurfaceRoughness) can be specified as well.

As for elastic ionized impurity scattering, the scattering rate is approximated by the inverse microscopic relaxation rate:

$$\frac{1}{\tau(\mathbf{k})} = \sum_{\mathbf{k}'} S(\mathbf{k}'|\mathbf{k})(1 - \cos(\phi)) \quad (6)$$

where ϕ is the angle between the in-plane wavevector components \mathbf{K}' and \mathbf{K} , and the after-scattering state is selected at random on the equienergy surface under conservation of the wavevector component normal to the gate interface.

In addition, this rate is multiplied by $\exp(-z/MCl_crit)$, which is analogous to the Lombardi model in drift-diffusion simulations.

This allows you to concentrate the effect of surface roughness in the vicinity of the surface (see *Sentaurus™ Device User Guide*, Enhanced Lombardi Model).

Fermi Statistics

For high doping levels, degeneracy of the carrier gas can have a significant impact on the drain current. The Monte Carlo models affected by Fermi statistics can be split into two groups.

Chapter 3: Physical and Numeric Models

Fermi Statistics

First, the expression for the inverse screening length in the Brooks–Herring model for ionized impurity scattering is:

$$\beta = \sqrt{e^2 \left(\frac{dn}{d\mu} - \frac{dp}{d\mu} \right) / \epsilon} \quad (7)$$

where:

- n is the electron density.
- p is the hole density.
- μ is the Fermi energy.

The expression is evaluated under degenerate conditions if `WithFermiDiracScreening` is specified.

This keyword is true by default and is applied if Fermi statistics is not used otherwise, since screening is the only aspect that is always significantly affected by Fermi statistics (compare with [Models for Band Structure and Scattering Mechanisms on page 49](#)).

Considering nonparabolicity and valley/band splitting (for example, under stress), the density expression for Fermi statistics is given by:

$$\begin{aligned} n &= \sum_i \tilde{N}_{c,i} \left(F_{1/2}(\eta_i) + \frac{15}{4} \alpha_i k_B T F_{3/2}(\eta_i) \right) \\ \eta_i &= \frac{\mu - E_i - E_c}{k_B T} \end{aligned} \quad (8)$$

where:

- E_c is the conduction band edge.
- E_i is the minimum of valley i with respect to E_c .
- α_i is the nonparabolicity factor of valley i .
- $N_{c,i}$ is the effective density-of-states of valley i with respect to E_i .
- $F_{1/2}$ and $F_{3/2}$ denote the Fermi–Dirac integrals of the order 1/2 and 3/2, respectively.

The Fermi energy μ at which the inverse screening length is evaluated in [Equation 7](#) is obtained for given density n by numerically solving:

$$n = \int dE D(E) \frac{1}{e^{\frac{E - \mu}{k_B T} - \eta} + 1} \quad (9)$$

$$\eta = \frac{\mu - E_c}{k_B T} \quad (10)$$

Chapter 3: Physical and Numeric Models

Fermi Statistics

where $D(E)$ is the total density-of-states of the band structure that is used in the Monte Carlo simulation. The Fermi energy for screening is always computed according to [Equation 9](#) even if Fermi statistics is not used otherwise, unless screening of impurity scattering is not used, that is, if the Conwell–Weisskopf model is used.

On the other hand, if Fermi statistics will be used completely during a Monte Carlo simulation, the following aspects are affected additionally:

- Carriers are injected from Ohmic contacts according to a semi-Fermi function.
- The Dirichlet boundary condition for the Poisson equation is changed for Ohmic contacts (see *Sentaurus™ Device User Guide*, Electrical Boundary Conditions).
- The Pauli blocking term $(1 - f)$, with f denoting the distribution function, must be considered in the scattering term of the Boltzmann transport equation. It is implemented according to the scheme proposed by Ungersboeck and Kosina [23].

Fermi statistics for these aspects is activated by specifying the keyword `Fermi` in the global `Physics` section of the command files of both the initial drift-diffusion and the actual Monte Carlo device simulations.

The Fermi energy is also needed for these aspects. The options for determining the Fermi energy are:

- The density expression $n = N_c F_{1/2}(\eta)$, where N_c is the total effective density-of-states, is used for the Fermi energy computation.

This expression is only strictly valid for a parabolic band structure. In the general case, it is only an approximation where full-band effects come into play only by using a different value for N_c . This option is the default.

- The density expression in [Equation 9](#) is used for the Fermi energy computation.

To activate this option, a short Monte Carlo run, where `CreateDOSFile=1` is specified in the `MonteCarlo` section of the Monte Carlo command file, must be performed before the initial drift-diffusion simulation and the actual Monte Carlo device simulation.

This run creates a file named `<MonteCarloOut>.dostot.dat`, containing the density-of-states of the Monte Carlo band structure where `MonteCarloOut` is an output field in the `File` section.

To read this file in the initial drift-diffusion and the actual Monte Carlo device simulation, for example, `eMultivalley(mcDOS(" <MonteCarloOut>.dostot.dat"))` must be specified in the `Physics` section of the command files of both the initial drift-diffusion and the actual Monte Carlo device simulations. For details about the options for reading this file, see *Sentaurus™ Device User Guide*, Using Multivalley Band Structure.

Chapter 3: Physical and Numeric Models

Arbitrary Lattice Temperature

Note:

The nonparabolicity option of the multivalley model in Sentaurus Device is not supported for Fermi statistics in Monte Carlo simulations.

For high source/drain (S/D) doping and low mobilities, Monte Carlo device simulations using Fermi statistics could involve numeric instabilities. In this case, you can restrict the maximum carrier density, in S/D regions, to the maximum doping for the computation of the Fermi energy, thereby preventing a too strong mobility reduction due to Monte Carlo carrier density fluctuations. This is *not* an approximation for S/D regions since the carrier density equals the doping. The options to specify S/D regions in the `MonteCarlo` section of the Sentaurus Device Monte Carlo command file are:

- `DopingtoSelectSDforFermi` selects all elements in the device where the doping is equal to or higher than the specified value.
- `WindowImpFactor1` and `WindowImpFactor2` select two windows that cover the S/D regions, and `ImpWindowtoSelectSDforFermi` activates the maximum carrier density restriction.

Arbitrary Lattice Temperature

Performing a Monte Carlo simulation for any lattice temperature between 50 K and 500 K is possible using, for example, the specification `temperature=245.0` in the `Physics` section of both the initial Sentaurus Device command file and the Sentaurus Device Monte Carlo command file.

Trajectory Calculation

Along lines that have been developed [24], the time during which the electron is propagated according to Newton's law is determined as the minimum of four times:

- The flight time to reach the border of the 3D momentum-space element
- The flight time to reach the border of the 2D real-space element
- The remaining time to the end of a time interval into which the whole simulation time is divided where, for example, simulation results are stored
- The stochastically selected time for a scattering event

Since momentum-space changes occur often, the equidistant tensor grid in momentum space is very useful for the calculation of the intersection with the border of a momentum-space element. There is an explicit proof of this time-step propagation scheme within the framework of basic probability theory [25].

Chapter 3: Physical and Numeric Models

Trajectory Calculation

This kind of trajectory calculation has several advantages. Within the scheme of self-scattering [3], it allows for the use of different and small upper estimates of the real scattering rates in each phase-space element. For the energy-dependent scattering rates of phonon scattering and impact ionization, an upper estimation is computed and stored for each momentum-space element. The corresponding rate for impurity scattering in [Equation 2](#) depends, in addition, on the impurity concentration N_{imp} and the electron density n . Therefore, an upper estimation is determined and stored for each real-space element by using the density obtained in the previous iteration (initially from the drift-diffusion simulation).

In addition, the computation of the logarithm for the free flight time can, for the most part, be avoided by first considering the probability, P_3 , that (real or fictitious) scattering occurs before the other three events:

$$P_3 = 1 - e^{-\Gamma t_3} \quad (11)$$

where Γ is the upper estimate of the real scattering rate and t_3 is the minimum of the times for the electron to leave the momentum-space element, leave the real-space element, and reach the end of the given time interval.

Therefore, the collisionless time-of-flight than t_f only needs to be computed if an equally (between 0 and 1) selected random number r is smaller than P_3 , and then is given by:

$$t_f = -\frac{1}{\Gamma} \ln(1 - r) \quad (12)$$

Another advantage is the simple integration of the Newton equations of motion, as the group velocity is constant in a momentum-space element and a constant electric field (taken from the drift-diffusion simulation) is assigned to a real-space element. However, an additional action is required for the Newton equations because the channel in MOSFETs, that is, the corresponding line from source to drain, is oriented along the crystallographic $\langle 110 \rangle$ direction, but the crystal momentum in the band structure calculation refers to a coordinate system with the coordinate axes parallel to the principal axes of Si.

Note:

This discussion does *not* refer to the growth direction of the wafer, which is in the z-direction, but to the direction within the xy plane parallel to the Si–SiO₂ interface.

Under the orthogonal transformation $\mathbf{k}' = U\mathbf{k}$, where \mathbf{k}' refers to the Cartesian frame that is aligned with the principal axes, the equations of motion become:

$$\frac{d}{dt}\mathbf{k}' = \left(-\frac{e}{\hbar}UE(\mathbf{r})\right) \quad (13)$$

$$\frac{d}{dt}\mathbf{r} = (U^T\mathbf{v}'(\mathbf{k}')) \quad (14)$$

Chapter 3: Physical and Numeric Models

Self-Consistent Single-Particle Approach

with the transformation matrix:

$$U = \begin{bmatrix} a & b & 0 \\ -b & a & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (15)$$

where $a = b = 1/\sqrt{2}$. This transformation must also be invoked for the surface roughness scattering process.

A further advantage is the possibility to restrict computational actions to the necessary cases only, for example, updating the group velocity of a particle only when the momentum-space element is left, or accessing the table with the real scattering rates only when a scattering process is to be performed.

Finally, the selection of the state-after-scattering is modeled with linked lists in the spirit of [26]. All cubes of the irreducible wedge of the Brillouin zone are stored in a list of energy intervals when they have a common energy range. The energy after scattering determines a corresponding energy interval in the list, and a cube is selected according to its partial density-of-states by the acceptance–rejection technique [3], with a constant upper estimation of the partial densities-of-states of all cubes in this energy interval. The momentum-after-scattering is then stochastically chosen on the equi-energy plane in this cube.

Self-Consistent Single-Particle Approach

In the tool, the total simulation time as given by the second argument of the parameter Range (see [Table 2 on page 38](#)) is split into a number of intervals, the number of which is given by the parameter `Intervals`.

During the simulation within each interval, first, a single particle is injected from a contact. It carries the whole electron charge as obtained by integrating the electron density of the drift-diffusion simulation over the entire Monte Carlo window. The probability for injection from an edge of a contact is proportional to the length of the edge multiplied by the density in the adjacent element. When the position is selected randomly on this edge, the momentum of the particle is chosen from a velocity-weighted Maxwellian [27]. Then, the particle is propagated in a frozen electric field (initially taken from a drift-diffusion simulation) until it is absorbed at a contact. These single-particle simulations continue until the time for the current simulation interval is over.

Then, the nonlinear Poisson equation [28] is solved to achieve self-consistency and it is solved with the density computed from the simulation interval that just ended. This density does not contain statistics gathered during previous simulation intervals. After solving the Poisson equation, the tool uses the computed electric field in the next simulation interval.

Chapter 3: Physical and Numeric Models

Gathering Statistics

The whole procedure is iterated until the end of the maximum simulation time or until a stopping criterion for the drain current is fulfilled. The tool ignores the statistical information gathered during a certain time (as specified by the parameter `FinalTime` (see [Table 2 on page 38](#)) at the beginning of the simulation when it computes the cumulative averages available to the user. When this ignored time span is long enough so that steady state is reached, it can be verified that the simulation results do not depend on the initial density distribution (obtained from a drift-diffusion or hydrodynamic simulation). Furthermore, the results do not depend on the duration of a simulation interval, provided it is long enough to gather sufficient statistics for the density distribution needed to solve the Poisson equation [\[29\]](#).

Gathering Statistics

During the simulation within a frozen-field iteration, cumulative expectation values of microscopic quantities, such as the group velocity, energy, and impact ionization scattering rate, are collected in each real-space element.

Usually, this is performed at equidistant time steps of the simulation, but this is very time-consuming CPU-wise if the time step is small. Therefore, statistics are gathered at times just before scattering [\[3\]](#).

Within the scheme of phase-space element-dependent upper estimations of the real scattering rate, the expectation value of a microscopic quantity A is given by:

$$\langle A \rangle_r = \frac{\sum_{i, r(t_i) \in r} \Gamma_{r, k_i}^{-1} A(\mathbf{k}(t_i))}{\sum_{i, r(t_i) \in r} \Gamma_{r, k_i}^{-1}} \quad (16)$$

where the sum runs over the times t_i of scattering events in the real-space element r , k_i denotes the momentum-space element occupied before t_i only, and $\Gamma_{r, k}^{-1}$ is the inverse upper estimation in the phase-space element (r, k) . This scheme for gathering statistics is applied to all quantities except for the density and drift velocity, which are estimated by time averaging. In addition, since a single-particle simulation is performed, the gathering of statistics can begin at the start of the simulation without the need to reach a stationary state beforehand, as is necessary in an ensemble simulation.

In the scheme described, phase-space elements are visited by the particle according to the real particle density. It is also possible to enhance the gathering of statistics in a specified window, which as a whole has a lower probability of being visited than the region outside this window. This is achieved by the following repetition scheme. The simulation domain is divided into two parts: one with a higher probability and one with a lower probability of being visited by the electron.

The history of the electron begins in the part with the higher probability and its state is stored when it enters the region of lower probability. When the electron leaves this region, the state

Chapter 3: Physical and Numeric Models

Estimating Currents

is changed back and the electron travels again through the low-probability region. This procedure is repeated a prescribed number of times before the electron continues along its path in the high-probability region and its statistical weight is adjusted accordingly. Therefore, the statistics in the low-probability region are improved. Whereas this scheme was used previously for hot-electron bulk transport [30], it was applied here to device simulation by identifying the low-probability and high-probability regions with, for example, the channel region and the highly doped source–drain regions, respectively.

Estimating Currents

This section briefly explains how estimations for drain and substrate currents are obtained. These estimations are performed after each frozen-field iteration, before the field is updated for the next iteration by using the nonlinear Poisson equation. After a certain number of iterations, the stationary state is reached and the solution fluctuates around its average value.

Although there is still a certain correlation between the iterations, the relative error provides a reasonable criterion to stop the simulation. Therefore, it is possible to discriminate adequately between the different simulation times that are necessary to obtain a similar accuracy, at different bias points, in analogy to a non-self-consistent simulation where there is strict statistical independence [31].

Drain Current

The drain current I_D is estimated by either direct particle counting (the default) or the test function method. For details about the latter, refer to the literature and its references [32].

Substrate Current

The substrate current I_S is calculated by using the expectation value of the impact ionization scattering rate S_{II} according to:

$$I_s = e \int d^2r n(\mathbf{r}) \langle S_{II} \rangle(\mathbf{r}) \quad (17)$$

where the integration is over the entire Monte Carlo window.

Note:

This formula is valid only when all generated holes can be assumed to leave the device through the substrate contact.

Contact Resistance

A contact resistance occurs, for example, at a silicide–silicon junction, which typically forms the contacts for the source and drain of a MOSFET [33]. The contact produces an additional voltage drop, and this effectively changes the applied biases and terminal currents in the device. To account for the contact resistance in Monte Carlo simulations, the nonlinear Poisson equation coupled with the contact equation is solved (similarly as it is performed in the drift-diffusion approach) after each carrier propagation step. The contact equation accounts for Monte Carlo–computed contact currents, computes the voltage drop at each contact, and sets it in the boundary conditions of the nonlinear Poisson equation. Note that the stochastical nature of the Monte Carlo method might affect the convergence of such iterations and, therefore, by default, Monte Carlo contact currents are averaged starting from the first step. This averaging could be justified by the assumption that finally the steady-state solution should be reached.

Similarly to any drift-diffusion simulation, both lumped and distributed resistances must be specified in the `Electrode` section:

$$\text{Resist} = \text{value } [\Omega \cdot \mu\text{m}^{3-d}]$$

or:

$$\text{DistResist} = \text{value } [\Omega \cdot \text{cm}^2]$$

where d is the dimension. For example, for 2D simulations, `Resist` must be specified in $\Omega \cdot \mu\text{m}$ (see *Sentaurus™ Device User Guide*, Resistive Contacts).

In addition, in the `MonteCarlo` section, the statement `SelfConsistent(FrozenQF)` must be replaced by `SelfConsistent(Coupled{Poisson Contact})`, which activates the coupled solution of the Poisson and contact equations.

Averages and Statistical Error of the Currents

The tool computes the drain current and substrate current for each simulation interval as described in [Drain Current on page 61](#) and [Substrate Current on page 61](#). After the tool has reached steady state, it takes cumulative averages of the currents obtained from single simulation intervals, I_i , to obtain their average value \bar{I}_n and their relative statistical error ΔI_n :

$$\bar{I}_n = \frac{1}{n} \sum_{i=1}^n I_i \quad (18)$$

$$\Delta I_n = \frac{2\bar{\sigma}_n}{\bar{I}_n} \quad (19)$$

Chapter 3: Physical and Numeric Models

Considering Traps

where the variance of the average current is:

$$\bar{\sigma}_n^2 = \frac{1}{nn-1} \sum_{i=1}^n (I_i - \bar{I}_n)^2 \quad (20)$$

and n is the number of simulation intervals since the tool reached steady state (see [Self-Consistent Single-Particle Approach on page 59](#)). The indexing of all quantities starts with 1 for the first simulation interval after steady state is reached; the tool discards the statistical information from simulation intervals before steady state is reached.

Unlike non-self-consistent simulations, the current estimations for different simulation intervals are not strictly stochastically independent, because two successive simulation intervals are coupled by the Poisson equation. Consequently, the usual interpretation of the relative error for the confidence interval of the expectation value does not hold.

Nevertheless, the relative error still shows the usual $1/\sqrt{n}$ behavior. Therefore, practically, the tool can use it as a stopping criterion. This is vital for the tool to adjust automatically the simulation time needed to reach the required accuracy for a particular device and bias point, which is simulated.

The probability that the average value \bar{I}_n deviates from the (unknown) expectation value \bar{I} by less than $\bar{\sigma}_n t_{\alpha, n-1}$, that is:

$$|\bar{I}_n - \bar{I}| < \bar{\sigma}_n t_{\alpha, n-1} \quad (21)$$

is $1 - \alpha$. Here, $t_{\alpha, n-1}$ is the student-t distribution function.

Use [Equation 19](#), [Equation 21](#), and the `CurrentErrorBar` and `MinCurrentComput` parameters (see [Table 2 on page 38](#)) to select the stopping criterion for a simulation.

For $n \geq 19$, the student-t distribution function obeys the inequality $1.9 \leq t_{0.05, n-1} \leq 2.1$. Then, $[\bar{I}_n(1 - \Delta I_n), \bar{I}_n(1 + \Delta I_n)]$ is approximately a 95% confidence interval for the true average of the current (in the case of statistically independent values). Values for the student-t distribution function can be found in the literature [34].

Considering Traps

The electrostatic effect of acceptor-like border traps, which give rise, in particular, to Fermi-level pinning, or donor-like interface traps can be taken into account in Monte Carlo device simulations by specifying the following in the `Solve` section of the Sentaurus Device Monte Carlo command file:

```
Coupled{ poisson electron }
Set(Traps(Frozen))
montecarlo
Set(Traps(-Frozen))
```

Chapter 3: Physical and Numeric Models

Quantum Correction

The parameters of the trap profiles must be specified in the `Physics` section for both the initial drift-diffusion simulation and the Monte Carlo device simulation. See *Sentaurus™ Device User Guide*, Chapter 17, for more information.

Quantum Correction

This section discusses quantum correction.

Effective Quantum Correction

Transport in short-channel devices is influenced both by quasiballistic and quantum-mechanical effects. However, the Boltzmann equation describes only semiclassical transport and does not include quantum confinement explicitly. The main quantum effects consist of a modification of the effective oxide thickness and a threshold shift. Therefore, the question arises as to how a quantum correction can be incorporated into a Monte Carlo simulation to consider these two effects. Recently, an approach was proposed that achieves this in a simple manner [35]: The oxide thickness is increased by using the difference of the charge centroids of quantum-mechanical (density gradient) and classical drift-diffusion simulations, and the threshold shift is compensated by a modification of the workfunction.

Using the modified oxide thickness (in terms of a modified dielectric constant of the gate oxide) and workfunction in the tool, the authors achieved a good agreement with measured on-currents in sub-0.1 µm NMOSFETs [35]. While analytic formulas were used for the modifications [35], this can also be achieved fully and automatically using Sentaurus Workbench. The result is shown in [Figure 6](#).

The three continuous curves refer to drift-diffusion simulations: The dashed curve is a classical drift-diffusion simulation, the dot-dashed curve is a density-gradient drift-diffusion simulation, and the solid curve is a drift-diffusion simulation, where the quantum correction was incorporated by the modified oxide thickness and workfunction. It can be seen that the quantum-corrected drift-diffusion simulation accurately reproduces the density-gradient simulation above the subthreshold regime.

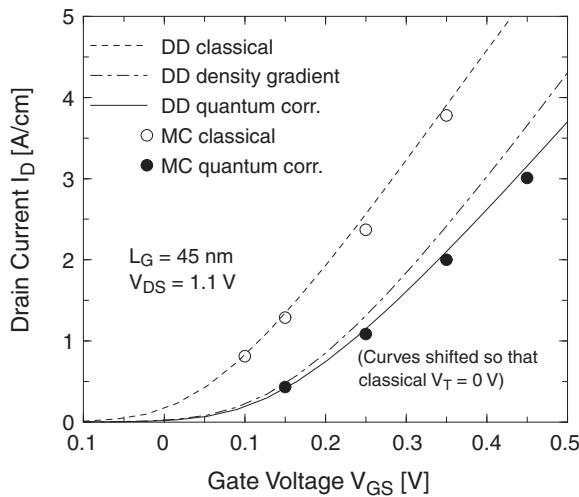
At higher gate voltages, the current is influenced by the mobility models and the emerging differences arise from the density-gradient mobility model not being calibrated to the universal mobility curve. Recalibrating this mobility would allow a perfect agreement [35], but here the aim is to use the modifications for oxide thickness and workfunction in Monte Carlo simulations.

In this respect, it can be seen that the quantum-corrected Monte Carlo curve shifts in the same way as the quantum-corrected drift-diffusion simulation.

Chapter 3: Physical and Numeric Models

Quantum Correction

Figure 6 Transfer characteristics of a 45 nm bulk NMOSFET according to classical, quantum-mechanical, and quantum-corrected simulations



From a technical perspective, this procedure can be implemented in the following way in a Sentaurus Workbench project [36]:

1. Process simulation and mesh generation using, for example, Sentaurus Mesh.
2. Density-gradient simulation storing the density profile perpendicular to the interface in the source-side of the channel (in [35], it is demonstrated that the result is insensitive to the choice of the position). This is achieved by the `NonLocalPlot` option of Sentaurus Device while also having specified a nonlocal mesh (which is, for example, also used for Schrödinger simulations).
3. Extraction of the quantum-mechanical charge centroid and the quantum-mechanical threshold voltage using Inspect.
4. Analogous classical drift-diffusion simulation.
5. Extraction of the classical charge centroid and calculation of the increased oxide thickness being converted into a reduced dielectric constant in the oxide using Inspect.
6. Classical drift-diffusion simulation with the modified oxide thickness.
7. Extraction of the threshold voltage for this modified classical drift-diffusion simulation using Inspect.
8. Classical drift-diffusion simulation with the changed oxide thickness where, in addition, the workfunction is changed by the threshold voltage difference to the density-gradient simulation.
9. Monte Carlo simulation with Sentaurus Device Monte Carlo using the modified oxide thickness and workfunction.

Chapter 3: Physical and Numeric Models

Visualizing Energy Distributions

While this procedure might appear lengthy, this approach is fully automatic when a Sentaurus Workbench project is set up and the CPU-limiting factor is still due to the Monte Carlo simulation. From the physical perspective, this methodology has the advantage that the surface roughness scattering model can still be consistently used in the Monte Carlo simulation when considering the quantum correction. Therefore, changes in the surface mobility arising, for example, under strain are still physically taken into account and it is possible to predict the associated tendencies.

For nonplanar multigate devices, this scheme must be extended to a second value of the effective workfunction in the on-state [37].

Density-Gradient Monte Carlo

As an alternative to the effective quantum correction based on effective oxide thickness and workfunctions, a quantum density profile can be explicitly taken into account in Monte Carlo simulations.

This can be achieved by considering a quantum potential from the modified local-density approximation (MLDA) model or the density gradient model, where the corresponding gradient contributes to the driving force and leads to a quantum-density profile. This approach is activated by introducing the same corresponding keywords as in the initial drift-diffusion simulation, for example, `eQuantumPotential` in the `Physics` section and in the `Coupled` statement of the `Solve` section.

See *Sentaurus™ Device User Guide*, Chapter 14, for more information.

Visualizing Energy Distributions

The energy distribution, corresponding to the product of the density-of-states multiplied by the distribution function, can be visualized for up to nine positions in a device by specifying, for example, for a 2D device simulation in units of μm :

```
EnergyDistributionPosition1 = (8.0e-3, -2.0e-3)
```

According to dimensionality, one, two, or three coordinates must be specified in the position vector. The integral of the energy distributions over energy is normalized to one. The energy distributions are stored in a file with the suffix `_endist.plt`, which shows the energy distributions as a function of energy.

Visualizing Valley Occupations

You can visualize the valley or valence-band occupations in a device by specifying the keyword `PlotValleyOccupations` in the `MonteCarlo` section of the Sentaurus Device Monte Carlo command file.

In addition, the variable names corresponding to the valleys of interest must be given in the `Plot` section of the Sentaurus Device Monte Carlo command file, that is:

- `eMCValleyDeltaX`, `eMCValleyDeltaY`, `eMCValleyDeltaZ`
- `eMCValleyGamma`, `eMCValleyLMinusMinus`, `eMCValleyLMinusPlus`,
`eMCValleyLPlusMinus`, `eMCValleyLPlusPlus`
- `hMCBandHeavyHole`, `hMCBandLightHole`, `hMCBandSplitOff`

The valley occupation is a number between 0 and 1.

Averages Over Forward- and Backward-Flying Carriers

To study carrier transport in more detail, averages of quantities such as group velocity can be computed over a subset of carriers. Specifically, carriers can be divided into forward-flying and backward-flying depending on whether the scalar product between their group velocity and a given direction vector is positive or negative, respectively (compare [38]). The corresponding forward and backward quantities are available for the density, current density, and drift velocity of the carrier.

You can activate the gathering of these additional statistics by specifying the keyword `InjectionDirection` in the `MonteCarlo` section. At the same time, `InjectionDirection` is the vector (in the device coordinate system) that determines whether a particle is forward flying or backward flying using the sign of the scalar product between its group velocity and `InjectionDirection`. As an illustration, based on the MCpFinFET example in the Applications Library, which can be found in the directory:

```
$STROOT/tcad/$STRELEASE/Applications_Library/GettingStarted/  
sdevice/MCpFinFET
```

the currents of forward- and backward-flying carriers are shown in [Figure 7](#), and the drift velocities of forward-flying carriers are shown in [Figure 8](#).

Note:

The difference between forward and backward currents equals the total current.

Of course, in addition to specifying `InjectionDirection` in the `MonteCarlo` section, you must also include the corresponding plot variables such as `hMCVelocityForward` in the `Plot` section (see [Table 1 on page 36](#)) to be able to visualize these quantities.

Chapter 3: Physical and Numeric Models

Ballistic Transport

Figure 7 Standard total current and the currents of forward- and backward-flying holes (absolute values, integrated over the fin width) along the channel of a 50 nm p-FinFET

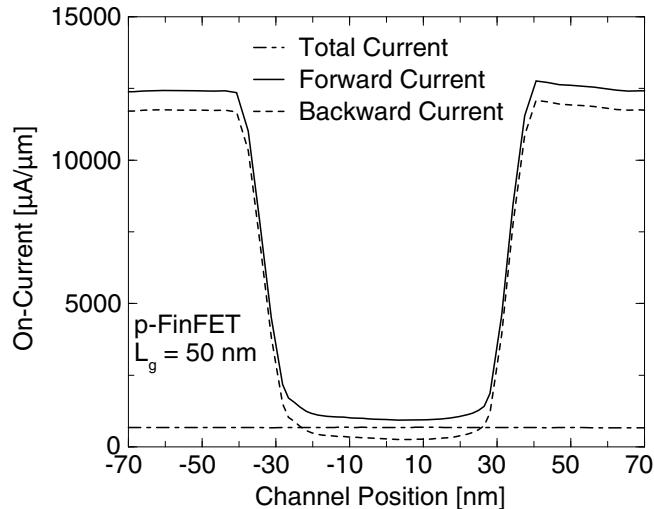
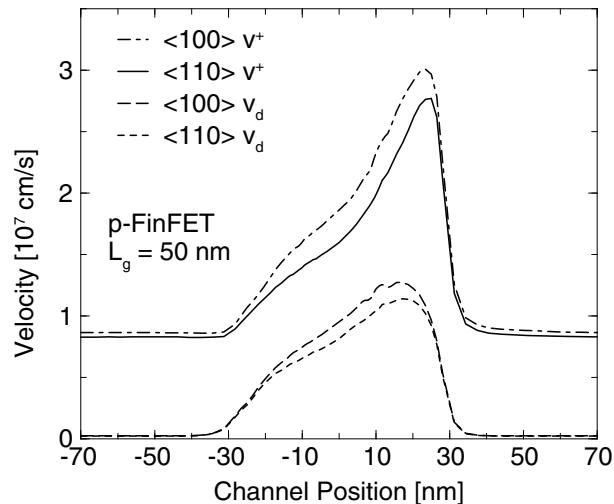


Figure 8 Standard drift velocity and the drift velocity of forward-flying holes (averaged with the corresponding hole densities over the fin width) along the channel of a 50 nm p-FinFET for two crystallographic channel orientations



Ballistic Transport

To investigate how far transport is from the ballistic limit, all scattering mechanisms can be switched off (and surface roughness scattering can be set to 100% specular) in a user-defined window in the device. Ballistic transport is activated by specifying the keyword

Chapter 3: Physical and Numeric Models

Ballistic Transport

`WindowBallistic` in the `MonteCarlo` section in analogy to the usual Monte Carlo window. At the same time, `WindowBallistic` specifies the window where scattering is switched off (see [Table 2 on page 38](#) for the syntax).

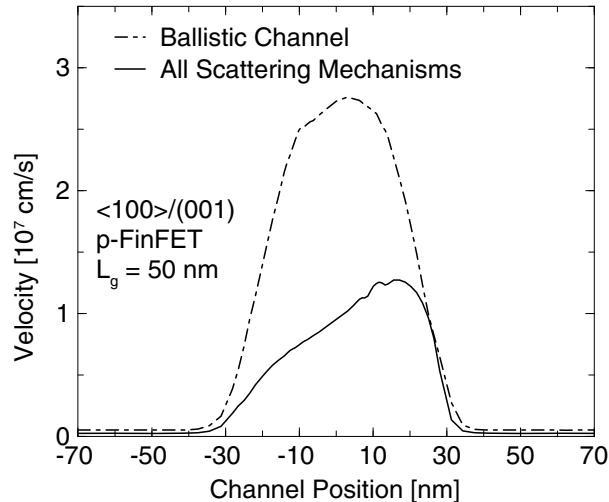
If surface roughness is still present in the ballistic window when switching off all other scattering mechanisms, `SurfScattRatioBallistic` can be set to a value smaller than 1 (which is the default corresponding to 100% specular scattering, that is, 0% diffusive scattering).

[Figure 9](#) shows as an example the drift velocity profiles along the channel of a 50 nm p-type FinFET when either all scattering processes are active or scattering is switched off in a window that comprises the channel between the two gate contacts. It can be seen that the ballistic velocity is more than a factor of two higher, that is, the device operates still far from the ballistic limit (the corresponding ballistic on-current is a factor of two higher).

Note:

Ballistic transport can lead to numeric instabilities. In this case, you can set `SurfScattRatioBallistic` to a value smaller than 1 or reduce the size of `WindowBallistic`.

Figure 9 Drift velocity (averaged with the corresponding hole densities over the fin width) in the on-state (supply voltage = -1.3 V) along the channel of a 50 nm p-FinFET when all scattering mechanisms are considered or when scattering is switched off in a window below the gate contact



References

- [1] M. M. Rieger and P. Vogl, "Electronic-band parameters in strained $\text{Si}_{1-x}\text{Ge}_x$ alloys on $\text{Si}_{1-y}\text{Ge}_y$ substrates," *Physical Review B*, vol. 48, no. 19, pp. 14276–14287, 1993.
- [2] F. M. Bufler, *Full-Band Monte Carlo Simulation of Electrons and Holes in Strained Si and SiGe*, Munich: Herbert Utz, 1998.
- [3] C. Jacoboni and L. Reggiani, "The Monte Carlo method for the solution of charge transport in semiconductors with applications to covalent materials," *Reviews of Modern Physics*, vol. 55, no. 3, pp. 645–705, 1983.
- [4] F. M. Bufler *et al.*, "Comparison of Single-Particle Monte Carlo Simulation with Measured Output Characteristics of an 0.1 μm n-MOSFET," *VLSI Design*, vol. 15, no. 4, pp. 715–720, 2002.
- [5] F. M. Bufler, A. Schenk, and W. Fichtner, "Simplified model for inelastic acoustic phonon scattering of holes in Si and Ge," *Journal of Applied Physics*, vol. 90, no. 5, pp. 2626–2628, 2001.
- [6] E. Cartier *et al.*, "Impact ionization in silicon," *Applied Physics Letters*, vol. 62, no. 25, pp. 3339–3341, 1993.
- [7] C. Canali, G. Ottaviani, and A. Alberigi Quaranta, "Drift Velocity of Electrons and Holes and Associated Anisotropic Effects in Silicon," *Journal of Physics and Chemistry of Solids*, vol. 32, pp. 1707–1720, 1971.
- [8] P. M. Smith, M. Inoue, and J. Frey, "Electron velocity in Si and GaAs at very high electric fields," *Applied Physics Letters*, vol. 37, no. 9, pp. 797–798, 1980.
- [9] C. Canali *et al.*, "Electron and Hole Drift Velocity Measurements in Silicon and Their Empirical Relation to Electric Field and Temperature," *IEEE Transactions on Electron Devices*, vol. ED-22, no. 11, pp. 1045–1047, 1975.
- [10] P. M. Smith, J. Frey, and P. Chatterjee, "High-field transport of holes in silicon," *Applied Physics Letters*, vol. 39, no. 4, pp. 332–333, 1981.
- [11] H. Brooks, "Scattering by Ionized Impurities in Semiconductors," *Physical Review*, vol. 83, p. 879, 1951.
- [12] B. K. Ridley, "Reconciliation of the Conwell–Weisskopf and Brooks–Herring formulae for charged-impurity scattering in semiconductors: Third-body interference," *Journal of Physics C: Solid State Physics*, vol. 10, no. 10, pp. 1589–1593, 1977.
- [13] P. Graf, *Entwicklung eines Monte-Carlo-Bauelementsimulators für Si/SiGe-Heterobipolartransistoren*, Munich: Herbert Utz, 1999.
- [14] H. Kosina, "A Method to Reduce Small-Angle Scattering in Monte Carlo Device Analysis," *IEEE Transactions on Electron Devices*, vol. 46, no. 6, pp. 1196–1200, 1999.

Chapter 3: Physical and Numeric Models

References

- [15] G. Masetti, M. Severi, and S. Solmi, "Modeling of Carrier Mobility Against Carrier Concentration in Arsenic-, Phosphorus-, and Boron-Doped Silicon," *IEEE Transactions on Electron Devices*, vol. ED-30, no. 7, pp. 764–769, 1983.
- [16] S.-M. Hong and C. Jungemann, "Inclusion of the Pauli Principle in a Deterministic Boltzmann Equation Solver for Semiconductor Devices," in *International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, Bologna, Italy, pp. 135–138, September 2010.
- [17] D. Chattopadhyay and H. J. Queisser, "Electron scattering by ionized impurities in semiconductors," *Reviews of Modern Physics*, vol. 53, no. 4, pp. 745–768, 1981.
- [18] F. M. Bufler and B. Meinerzhagen, "Hole transport in strained $\text{Si}_{1-x}\text{Ge}_x$ alloys on $\text{Si}_{1-y}\text{Ge}_y$ substrates," *Journal of Applied Physics*, vol. 84, no. 10, pp. 5597–5602, 1998.
- [19] F. M. Bufler, F. O. Heinz, and L. Smith. "Efficient 3D Monte Carlo Simulation of Orientation and Stress Effects in FinFETs," in *Proceedings of the International Conference on Semiconductor Processes and Devices (SISPAD)*, Glasgow, Scotland, pp. 172–175 September 2013.
- [20] K. Akarvardar *et al.*, "Impact of Fin Doping and Gate Stack on FinFET (110) and (100) Electron and Hole Mobilities," *IEEE Electron Device Letters*, vol. 33, no. 3, pp. 351–353, 2012.
- [21] M. Lundstrom, *Fundamentals of carrier transport*, Cambridge: Cambridge University Press, 2nd ed., 2000.
- [22] M. V. Fischetti *et al.*, "Theoretical Study of Some Physical Aspects of Electronic Transport in nMOSFETs at the 10-nm Gate-Length," *IEEE Transactions on Electron Devices*, vol. 54, no. 9, pp. 2116–2136, 2007.
- [23] E. Ungersboeck and H. Kosina, "The Effect of Degeneracy on Electron Transport in Strained Silicon Inversion Layers," in *International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, Tokyo, Japan, pp. 311–314, September 2005.
- [24] J. Bude and R. Kent Smith, "Phase-space simplex Monte Carlo for semiconductor transport," *Semiconductor Science and Technology*, vol. 9, pp. 840–843, 1994.
- [25] F. M. Bufler, A. Schenk, and W. Fichtner, "Proof of a simple time-step propagation scheme for Monte Carlo simulation," *Mathematics and Computers in Simulation*, vol. 62, pp. 323–326, 2003.
- [26] C. Jungemann *et al.*, "Efficient Full-Band Monte Carlo Simulation of Silicon Devices," *IEICE Transactions on Electronics*, vol. E82-C, no. 6, pp. 870–879, 1999.
- [27] T. González and D. Pardo, "Physical Models of Ohmic Contact for Monte Carlo Device Simulation," *Solid-State Electronics*, vol. 39, no. 4, pp. 555–562, 1996.
- [28] F. Venturi *et al.*, "A General Purpose Device Simulator Coupling Poisson and Monte Carlo Transport with Applications to Deep Submicron MOSFET's," *IEEE Transactions on Computer-Aided Design*, vol. 8, no. 4, pp. 360–369, 1989.

Chapter 3: Physical and Numeric Models

References

- [29] F. M. Bufler *et al.*, “Single-Particle Approach to Self-Consistent Monte Carlo Device Simulation,” *IEICE Transactions on Electronics*, vol. E86-C, no. 3, pp. 308–313, 2003.
- [30] A. Phillips, Jr. and P. J. Price, “Monte Carlo calculations on hot electron energy tails,” *Applied Physics Letters*, vol. 30, no. 10, pp. 528–530, 1977.
- [31] F. M. Bufler, A. Schenk, and W. Fichtner, “Efficient Monte Carlo Device Simulation with Automatic Error Control,” in *International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, Seattle, WA, USA, pp. 27–30, September 2000.
- [32] P. D. Yoder, K. Gärtner, and W. Fichtner, “A generalized Ramo–Shockley theorem for classical to quantum transport at arbitrary frequencies,” *Journal of Applied Physics*, vol. 79, no. 4, pp. 1951–1954, 1996.
- [33] N. Stavitski *et al.*, “Systematic TLM Measurements of NiSi and PtSi Specific Contact Resistance to n- and p-Type Si in a Broad Doping Range,” *IEEE Electron Device Letters*, vol. 29, no. 4, pp. 378–381, 2008.
- [34] I. N. Bronshtein and K. A. Semendyayev, *Handbook of Mathematics*, Berlin: Springer, 3rd ed., 1997.
- [35] R. Hudé *et al.*, “A Simple Approach to Account for the Impact of Quantum Confinement on the Charge in Semi Classical Monte Carlo Simulations of Bulk nMOSFETs,” in *6th International Conference on ULtimate Integration of Silicon (ULIS)*, Bologna, Italy, pp. 159–162, April 2005.
- [36] F. M. Bufler, R. Hudé, and A. Erlebach, “On a Simple and Accurate Quantum Correction for Monte Carlo Simulation,” in *11th International Workshop on Computational Electronics (IWCE)*, Vienna, Austria, pp. 101–102, May 2006.
- [37] F. M. Bufler and L. Smith, “3D Monte Carlo simulation of FinFET and FDSOI devices with accurate quantum correction,” *Journal of Computational Electronics*, vol. 12, no. 4, pp. 651–657, 2013.
- [38] P. Palestri *et al.*, “Understanding Quasi-Ballistic Transport in Nano-MOSFETs: Part I—Scattering in the Channel and in the Drain,” *IEEE Transactions on Electron Devices*, vol. 52, no. 12, pp. 2727–2735, 2005.

4

Strained Silicon

This chapter describes how to simulate strained-silicon devices with single-particle device Monte Carlo and presents some typical results.

Bulk Properties

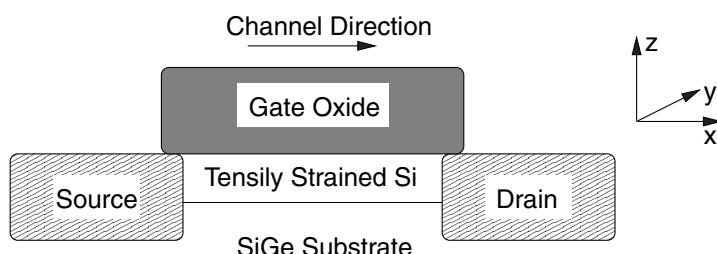
Bulk silicon and bulk germanium have different lattice constants, namely, $a_{0,\text{Si}} = 5.43 \text{ \AA}$ and $a_{0,\text{Ge}} = 5.65 \text{ \AA}$. The lattice constant in the alloy $\text{Si}_{1-x}\text{Ge}_x$ formed from both materials varies almost linearly between the values of the pure materials. A more precise fit [1] derived from measurements [2] is:

$$a_0(x) = a_{0,\text{Si}} + 0.200326x(1-x)\text{\AA} + (a_{0,\text{Ge}} - a_{0,\text{Si}})x^2 \quad (22)$$

A difference in the lattice constants between two bulk materials can be used to form a layer under mechanical strain with possibly different transport properties. If, for example, a thin silicon film is grown on top of a thick relaxed $\text{Si}_{1-x}\text{Ge}_x$ substrate, the so-called in-plane lattice constants parallel to the interface will adopt the value of the lattice constant in the substrate if the film thickness is less than a critical thickness in the order of 10 nm.

The out-of-plane lattice constant of the layer in the perpendicular direction acquires a value according to elasticity theory. In the case of a silicon layer on a relaxed SiGe substrate, the in-plane lattice constants are increased and, therefore, the silicon layer is under biaxial tensile strain. The corresponding strained-silicon channel in the conventional lateral MOSFET is schematically shown in [Figure 10](#). The strain changes the band structure.

Figure 10 Schematic structure of a planar MOSFET illustrating the formation and geometry of a Si channel under biaxial tensile strain

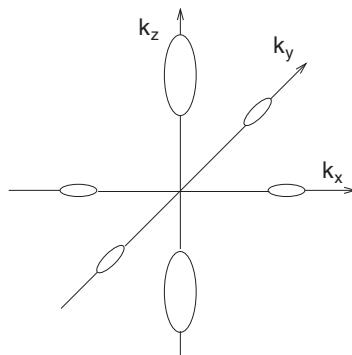


Chapter 4: Strained Silicon

Bulk Properties

In the case of electrons, four of the six valleys in the first conduction band shift upwards in energy as illustrated in [Figure 11](#). Here, the growth direction is along the z-axis. The x- and y-coordinates correspond to the in-plane directions. Since the two energetically lower-lying valleys are predominantly populated, electrons traveling in a direction parallel to the Si–SiGe interface experience only the small transverse effective mass. Obviously, this enhances in-plane transport. However, in contrast to what typical illustrations such as [Figure 11](#) suggest, the band energies within one lower-lying valley resulting from the pseudopotential calculations are, for higher energies, no longer identical along the in-plane <110> and <100> directions.

Figure 11 Equienergy surfaces of the first conduction band in strained silicon: Four of the six valleys shift upwards in energy under strain. For a Ge content of $x=0.2$ in the relaxed $\text{Si}_{1-x}\text{Ge}_x$ substrate, the valley splitting amounts to 126 meV.



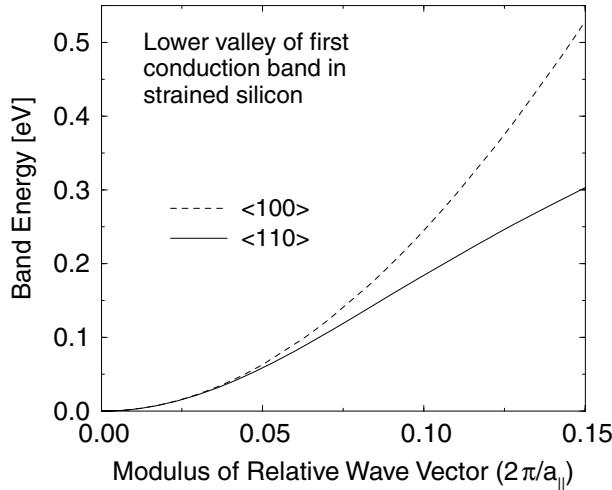
This is demonstrated in [Figure 12](#) where the band energies in one lower-lying valley are plotted as a function of the modulus of the in-plane \mathbf{k} -vector for the <110> and <100> directions. It can be seen that the energies differ significantly above 100 meV and the band curvature is then much stronger in the <100> direction.

This feature is not captured by the usual analytic formula for a nonparabolic, anisotropic band structure [3] where the energy dispersion in both directions is exactly the same as shown in the literature [4]. This difference is crucial for the on-current in nanoscale MOSFETs as explained in the literature [5] and discussed in [NMOSFETs on page 80](#).

Chapter 4: Strained Silicon

Bulk Properties

Figure 12 In-plane energy dispersion ($k_z = \text{const.} \approx 0.85 2\pi/a_{\perp}$, where a_{\perp} is the out-of-plane lattice constant) along the $\langle 100 \rangle$ and $\langle 110 \rangle$ directions in a lower valley of strained Si grown on a relaxed $\text{Si}_{0.8}\text{Ge}_{0.2}$ substrate



In the case of holes, the most important change concerns the splitting of the heavy-hole and light-hole bands at the Γ -point with the light-hole band being situated at the valence-band edge, but in contrast to the case of electrons, the shape within a band is also affected by strain. This is illustrated in [Figure 13](#) where the energy dispersions of the heavy-hole and light-hole bands in unstrained and (001)-strained Si are shown along two different crystallographic directions.

It can be clearly seen in [Figure 13 \(top\)](#) that the light-hole band is at the valence-band edge in strained Si. Growing strain increases the split between the bands and, therefore, decreases the occupation probability of the heavy-hole band. However, the curvature of the light-hole band structure along the $\langle 110 \rangle$ direction does not change further, below 130 meV, as strain increases. Only in the $\langle 011 \rangle$ direction does the band curvature continue to change under strain being thus the origin of the continuing decrease of the density-of-states for increasing strain as reported [\[6\]](#).

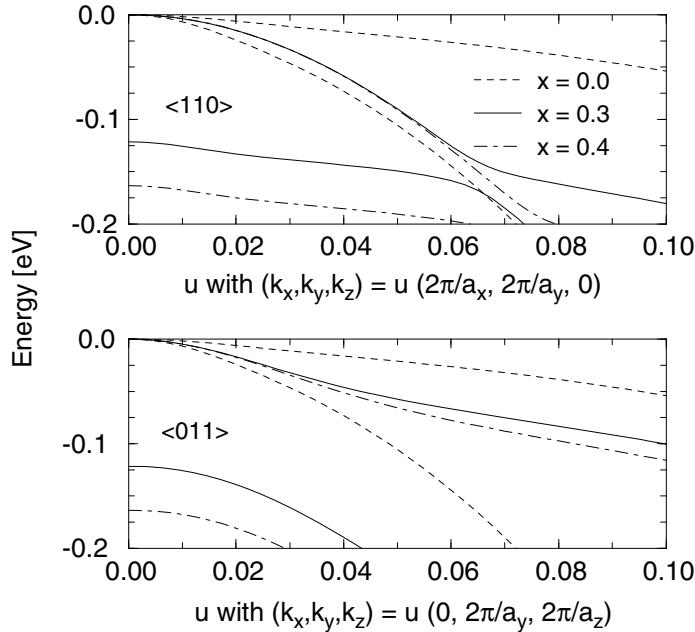
The scattering mechanisms are, as usual, assumed to be unaffected by strain. The modified transport properties of electrons and holes are, therefore, entirely related to the changes of the band structures described above. Impact ionization is not considered for strained silicon.

As confirmed by recent experiments [\[7\]](#), the strain-induced enhancement of transport saturates in the case of electrons above a germanium content of 20% in the SiGe substrate. The reason is that at this Ge content almost all electrons are already in the two lower valleys. Consequently, further increases of the valley splitting have no effect. Therefore, the investigation of electron transport in strained Si is restricted to the strain level corresponding to 20% Ge content in the substrate. In contrast, the anisotropy between the crystallographic $\langle 100 \rangle$ and $\langle 110 \rangle$ in-plane directions is studied, because the different band curvatures in [Figure 12](#) lead to an expectation of different transport properties.

Chapter 4: Strained Silicon

Bulk Properties

Figure 13 Energy dispersion along (top) $\langle 110 \rangle$ and (bottom) $\langle 011 \rangle$ direction of the heavy-hole and light-hole bands in unstrained Si and (001)-strained Si grown on a relaxed $Si_{0.7}Ge_{0.3}$ and $Si_{0.6}Ge_{0.4}$ substrate, respectively



For a small electric field, most electrons populate the two lower-lying valleys in strained Si, where they experience only the small transverse effective mass, which in conjunction with reduced intervalley scattering leads to an enhanced drift velocity. This can be seen in the velocity-field characteristics in [Figure 14](#).

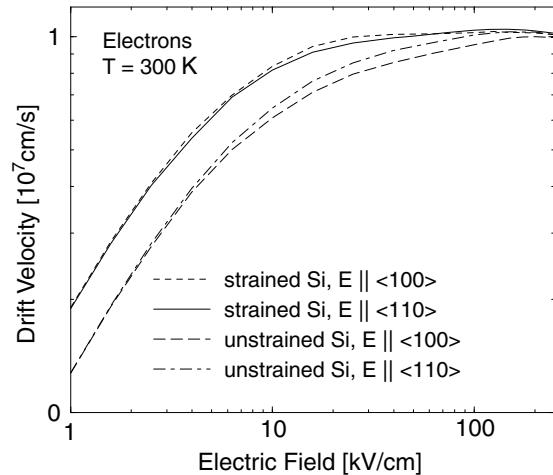
For the previously mentioned analytic band structure description together with the phonon model employed, it follows from symmetry that the Ohmic mobility is, both in unstrained and strained silicon, exactly the same in the $\langle 100 \rangle$ and $\langle 110 \rangle$ directions. [Figure 14](#) shows that this is still valid in the case of the full band structures that are used in the present Monte Carlo simulations.

In the nonlinear regime, the drift velocities begin to differ between the two crystallographic directions. In strained Si, the drift velocity is greater in the $\langle 100 \rangle$ direction because of the stronger band curvature shown in [Figure 12](#). However, the maximum improvement of the velocity in the $\langle 100 \rangle$ direction is only 3.8%, occurring at a field strength of approximately 25 kV/cm. In contrast, the drift velocity is greater in the $\langle 110 \rangle$ direction in the case of unstrained silicon. This is due to valley repopulations (compare [\[8\]](#)) and is explained later in this section when discussing quasiballistic transport.

Chapter 4: Strained Silicon

Bulk Properties

Figure 14 Velocity-field characteristics of electrons at 300 K in unstrained and strained Si grown on a relaxed $\text{Si}_{0.8}\text{Ge}_{0.2}$ substrate along crystallographic $\langle 100 \rangle$ and $\langle 110 \rangle$ directions



For the highest electric fields, the anisotropy vanishes for unstrained and strained silicon as energetically higher valleys and bands are being populated. The most important consequence of the increasing population of the four higher-lying valleys in strained Si is, however, the disappearance of the velocity improvement under strain. In fact, the saturation velocity is independent of strain and direction, attaining a value of approximately $1 \times 10^7 \text{ cm s}^{-1}$.

Regarding the issue of a comparison with measurements, they are very scarce for strained bulk Si. However, in the case of electrons, some published experimental data is available. In this respect, the low-field drift mobility of approximately $2200 \text{ cm}^2/(\text{Vs})$ for a substrate Ge content of 30% is, together with a Hall factor of 1.3 [9], in good agreement with the measured Hall mobility of $2800 \text{ cm}^2/(\text{Vs})$ [10]. In addition, the measured saturation of the drift velocity at a much lower field than in unstrained silicon [11] is reproduced by the model used in the tool (see [Figure 14](#) and [9]). In contrast to other theoretical models [12][13][14], which do not reproduce these experimental results, this model should be a reliable basis for the device simulation described in [NMOSFETs on page 80](#).

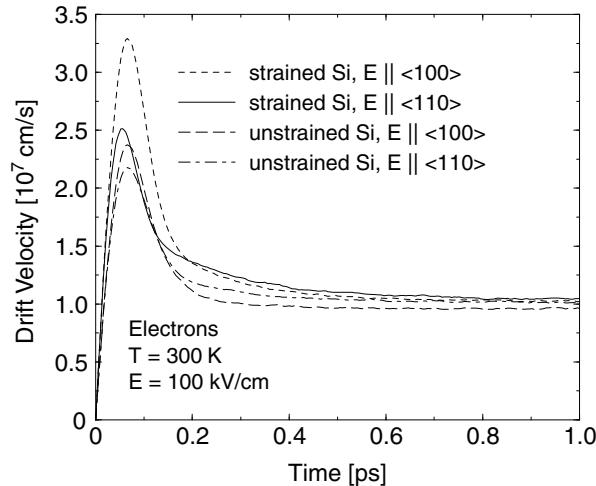
Finally, quasiballistic transport, which occurs when electrons in or near equilibrium suddenly experience a strong electric field, is addressed. This can occur in time (for example, in a spatially homogeneous bulk system) or in space (for example, under stationary-state conditions in a device). The effect is that the electrons then suffer only a few scattering events over a short distance (in time or space) and, therefore, attain a higher velocity than the velocity that would correspond to the electric field in the velocity-field characteristics shown in [Figure 14](#). Whether and to what extent quasiballistic transport occurs is fixed by (a) the strength of the driving field, (b) the scattering rate, and (c) the band curvature that determines the gain in velocity during a free flight. In this section, quasiballistic transport on the bulk level is studied by suddenly applying a strong field of 100 kV/cm to an ensemble of electrons in equilibrium. [Figure 15](#) displays the transient velocities for electrons in

Chapter 4: Strained Silicon

Bulk Properties

unstrained and strained silicon where the field is oriented along the $<100>$ or $<110>$ direction. An overshoot peak emerges before the electrons finally acquire the stationary drift velocity.

Figure 15 Transient velocity overshoots of electrons at 300 K after a sudden application of a 100 kV/cm field in unstrained Si and strained Si grown on a relaxed $\text{Si}_{0.8}\text{Ge}_{0.2}$ substrate along the crystallographic $<100>$ and $<110>$ directions



An overshoot only occurs if the applied field is strong enough. For weak fields, the drift velocity increases monotonically until the stationary value is reached according to the Drude theory of an electron gas. In unstrained silicon, the overshoot peak is larger in the $<100>$ direction while the velocity in the $<110>$ direction becomes higher when the stationary state is reached. The reason is that, in the $<100>$ direction, the electrons in *four* valleys experience only the small transverse effective mass and, therefore, at first, a larger drift velocity is attained. As these electrons also assume a higher energy, intervalley scattering is stronger, which then enhances the population of the two valleys with the large longitudinal mass in the transport direction. This finally leads to a smaller stationary drift velocity in the $<100>$ direction [8] (however, the anisotropy of the stationary velocities vanishes in the limit of very high fields).

In contrast, the overshoot phenomenon in strained Si is determined solely by the two lower-lying valleys. Since neither the scattering rate in the two valleys nor the field strength is affected by changing the direction of the electric field, the anisotropy of the overshoot peak stems only from the different band curvatures shown in [Figure 12 on page 75](#), which lead to a stronger peak in the $<100>$ direction. It must be emphasized that the most important difference between unstrained and strained silicon consists of the much stronger anisotropy of the overshoot peak in strained Si [4][15] (31% in strained silicon versus 9% in unstrained silicon).

In the case of holes, the performance of strained-silicon PMOSFETs reported [6] is only slightly affected by the crystallographic in-plane direction. In contrast, there is a strong

Chapter 4: Strained Silicon

Bulk Properties

dependence on strain, that is, on the Ge content in the substrate as confirmed by recent experiments [16]. Therefore, the strain dependence of hole transport is investigated explicitly.

The velocity-field characteristics of holes for different strain levels are shown in Figure 16 and the corresponding transient velocity overshoot peaks are shown in Figure 17.

Figure 16 In-plane velocity-field characteristics of holes in unstrained and strained silicon at 300 K with the field parallel to the crystallographic $\langle 110 \rangle$ direction

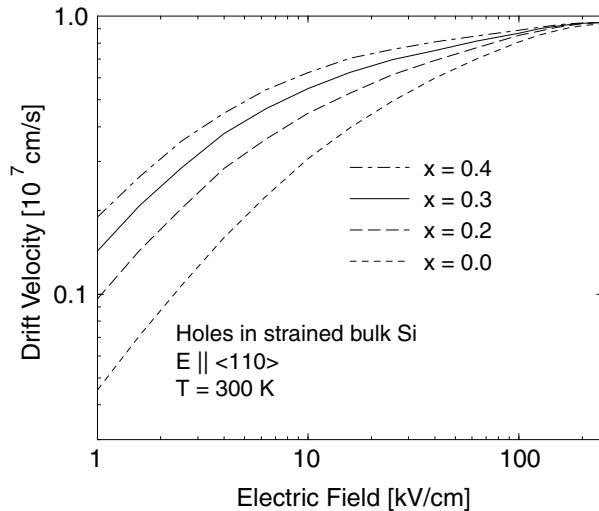
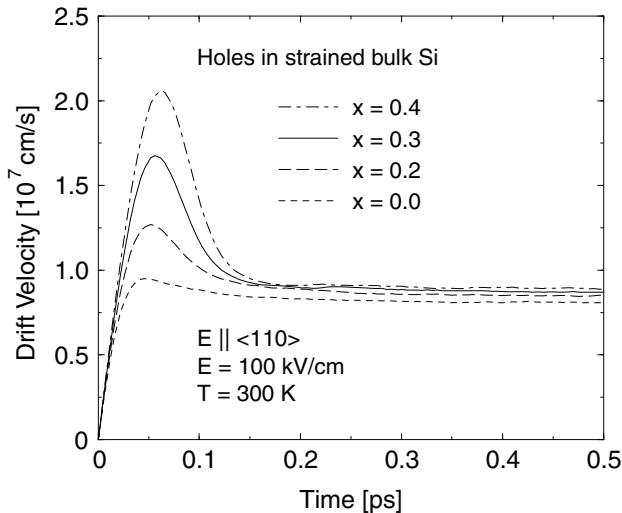


Figure 17 Transient velocity overshoots of holes at 300 K after a sudden application of a 100 kV/cm field along the crystallographic $\langle 110 \rangle$ direction in unstrained and strained silicon



It is clear that both the drift velocity at low fields and the transient velocity in [Figure 17](#) are enhanced as a function of strain. This is because the heavy-hole is pushed away from the valence-band edge under strain as shown in [Figure 13 on page 76](#), which reduces the effective mass experienced by the holes as well as interband scattering.

However, as a higher field eventually increases the energy of the holes, the population of the heavy-hole band is again enhanced and, consequently, the strain-induced improvement vanishes for the saturation velocity in [Figure 16](#). Note that the continuing enhancement for higher Ge substrate contents is due to the increase of the light-hole band curvature in the <011> direction in [Figure 13](#), which leads to a decrease of the density-of-states at lower energies and, therefore, to a smaller scattering rate. The overshoot peaks of holes in [Figure 17](#) are generally smaller than those for electrons in [Figure 15](#) because the effective masses are larger.

NMOSFETs

This section discusses the main findings regarding the reported performance of strained-silicon NMOSFETs [\[5\]](#). The focus is on the decisive issue of how the strain-induced performance enhancement scales as a function of the gate length.

MOSFETs with three different gate lengths are simulated. The scaling and simulation methodology for the comparison of unstrained and strained material is as follows. The original structure with a gate length of $L_{ch} = 100$ nm ($L_{eff} = 50$ nm) was taken from the literature [\[17\]](#) and then scaled to $L_{ch} = 50$ nm and $L_{ch} = 150$ nm, respectively, keeping only the oxide thickness constant with a value of $t_{ox} = 2$ nm similar to the literature [\[18\]](#) and readjusting a constant channel doping such that the off-current does not change.

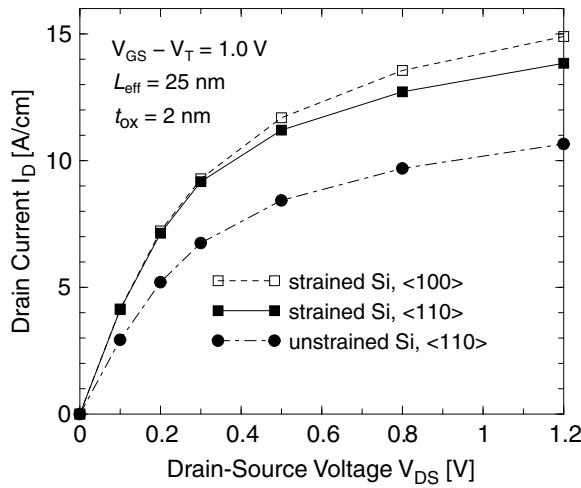
The geometry of the lightly doped drain (LDD) structure for the smallest simulated NMOSFET is shown in Figure 1 of the literature [\[5\]](#). The maximum values of the two *n*-type Gaussian profiles of the LDD structure are $N_D = 1.5 \times 10^{20}$ cm⁻³ and $N_D = 5 \times 10^{19}$ cm⁻³. The constant *p*-type doping levels in the channel are $N_A = 2.03 \times 10^{18}$ cm⁻³, 1.18×10^{18} cm⁻³, and 8.80×10^{17} cm⁻³ for the gate lengths of $L_{ch} = 50$ nm, 100 nm, and 150 nm, respectively, thereby increasing for smaller L_{ch} in accordance with the scaling rules.

Again in analogy to the literature [\[18\]](#), no attempt was made in the different configurations to achieve the same threshold voltages V_T , which are smaller for strained Si, in particular, due to the smaller band gap. The comparability of the drain currents between the different simulations is ensured by always applying the same gate overdrive of $V_{GS} - V_T = 1.0$ V. The value of the drain voltage V_{DS} , which defines the on-current, is taken to be 1.2 V [\[18\]](#). The SiGe substrate is not explicitly considered in the simulation. The germanium content of the substrate, which is 20% in this study as in the experimental work [\[18\]](#), only serves to define the amount of strain in the MOSFET. The strain-induced change of the band gap with respect to unstrained Si was computed with the help of the model solid theory of Van de Walle [\[19\]](#). The transport parameters such as the effective densities-of-states or velocity-field characteristics entering the drift–diffusion model, which was used for the

determination of the threshold voltage from the slope of the transfer characteristics, were consistently taken from Monte Carlo bulk simulations or the underlying band structure calculation.

[Figure 18](#) shows the output characteristics of the smallest simulated NMOSFET with an effective gate length of $L_{\text{eff}} = 25 \text{ nm}$ based on unstrained or strained Si.

Figure 18 Output characteristics of the smallest simulated NMOSFET with an effective channel length of 25 nm for strained Si with the channel oriented along the crystallographic $<100>$ direction, as well as for strained and unstrained Si with orientation of the channel along $<110>$ direction



Since the drain current in the unstrained-Si device was nearly the same for both crystallographic orientations, in this case, only results with a channel orientation along the $<110>$ direction are presented, as is usual in standard CMOS. (The substrate orientation, corresponding to the growth direction of the strained Si layer on the virtual SiGe buffer, is always in the (001) direction, that is, in the z-direction in the schematic MOSFET in [Figure 10 on page 73](#).)

Explicit results for the anisotropy between channel orientations along the $<110>$ and $<100>$ directions will be restricted to strained Si. In the linear regime, the drain currents in the strained-Si NMOSFET are the same for both directions, consistent with identical bulk low-field mobilities in [Figure 14 on page 77](#). This shows that surface roughness scattering does not affect this symmetry. Anisotropy appears only at higher drain voltages and results in approximately a 10% higher strain-induced improvement of the on-current for the $<100>$ direction.

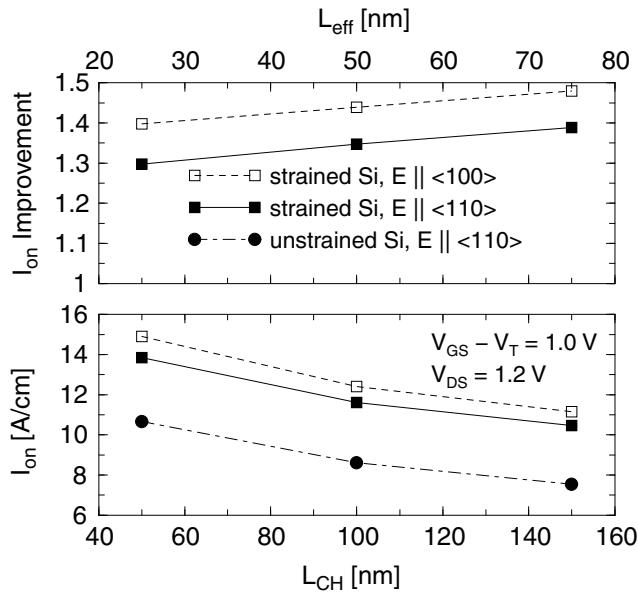
[Figure 19](#) demonstrates that the improvement by strain, despite a tendency to decrease, persists at a high level as the gate length is reduced. The enhancements in [Figure 19 \(top\)](#) are also in good agreement with the measured I_{on} improvement of approximately 35% reported in Figure 10 of the literature [18] for an effective gate length of $L_{\text{eff}} = 67 \text{ nm}$ (assuming the channel orientation was in the standard $<110>$ direction). These results show

Chapter 4: Strained Silicon

On-Current Interpretation

that strained-Si NMOSFETs offer also in the 0.1 μm regime strong performance enhancements over conventional CMOS. Together with the demonstration of the feasibility within standard technology [18], the present work suggests that strained silicon is a viable candidate for the continuation of increasing silicon nanoelectronics performance.

Figure 19 Dependence of on-current on gate length: (top) improvement by strain for <100> and <110> directions relative to the unstrained case and (bottom) absolute values for strained and unstrained Si



On-Current Interpretation

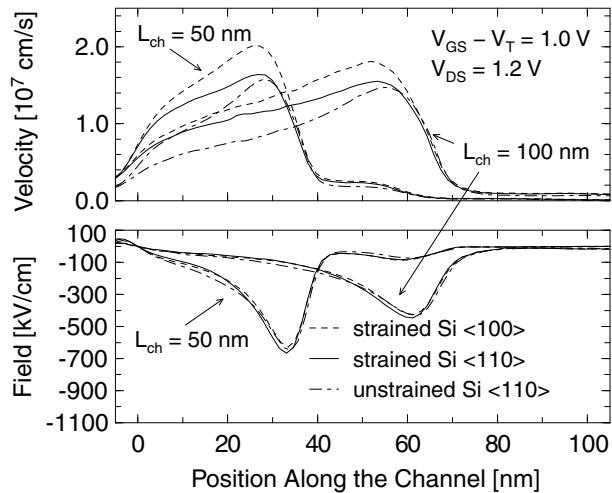
While the simulated performance enhancements by strain in nanoscale NMOSFETs are in good agreement with measurements [18], they are surprising from a physical point of view, because Figure 14 on page 77 shows that the strain-induced velocity enhancement vanishes for high fields. Therefore, this section explains the physics influencing the on-current by an investigation of the internal velocity profiles and especially the anisotropy in strained silicon.

Figure 20 shows the velocity and field profiles along the channel for the two smaller transistors. Scaling increases the electric field (*bottom plot*) and the drift velocity (*top plot*) in the source side of the channel and consequently enhances the on-current (see Figure 19 (*bottom*)). The position where the longitudinal field changes the sign at the source side of the channel is taken as origin for both MOSFETs. The velocity and field are averaged with the electron density perpendicularly to the Si–SiO₂ interface.

Chapter 4: Strained Silicon

On-Current Interpretation

Figure 20 (Top) Drift velocity and (bottom) longitudinal electric field profiles along the channel in the NMOSFETs with channel lengths of 50 nm and 100 nm, respectively, for strained Si in $<100>$ direction as well as for strained and unstrained Si in $<110>$ direction



Since the field always starts from zero before increasing into the channel, the velocity attained along a certain distance in the source side of the channel is caused by a combination of linear, nonlinear, and quasiballistic transport. In addition, quasiballistic transport will, even in the presence of a high field, only occur if at the same time the scattering rate is small and the band curvature is sufficiently strong.

The reduction of the strain-induced I_{on} improvement for decreasing gate length in Figure 19 (top) can be attributed to a contribution of nonlinear transport because the velocity improvement by strain decreases for an increasing field in the bulk velocity-field characteristics displayed in Figure 14 on page 77. On the other hand, it appears that nonlinear transport cannot explain an I_{on} improvement of 30% in the smallest NMOSFET because, after 5 nm, the field is already approximately 70 kV/cm where the strain-induced improvement in the velocity-field characteristics amounts to less than 5%. In contrast to a vanishing strain-induced improvement in the nonlinear regime, both linear and quasiballistic transport are enhanced under strain as discussed in Bulk Properties on page 73 and the correlation between an improved low-field mobility and an enhancement of the source-side velocity has been confirmed empirically [20].

The difference between linear and quasiballistic transport in strained silicon is, however, that symmetry implies identical low-field mobilities in the $<100>$ and $<110>$ directions in contrast to strong anisotropy for quasiballistic transport as seen in Figure 15 (the anisotropy of I_{on} in unstrained Si is nearly absent because the, on average, larger in-plane mass leads to a smaller contribution of quasiballistic transport and because the anisotropy of the overshoot peaks in Figure 15 is much weaker). Note that the maximum anisotropy in the velocity-field characteristics of strained Si is only 3.8% as shown in Figure 14. Therefore, the 10% stronger improvement of the on-current in the $<100>$ direction can be attributed to a

contribution of quasiballistic transport. The crucial role of quasiballistic transport is also supported by the fact that the velocity in the smallest strained-silicon MOSFET is significantly anisotropic after 7 nm and appreciably above the saturation drift velocity (see [Figure 20 \(top\)](#)).

It can be concluded that the correlation of the low-field mobility with I_{on} is, apart from contributing to the source–drain resistance, mainly due to a similar strain dependence as for quasiballistic transport (both the drift velocity in the linear regime in [Figure 14](#) and the overshoot peak in [Figure 15](#) are improved under strain).

Finally, it should be noted that models that relate the on-current to the thermal injection velocity and backscattering [21] cannot explain the anisotropy of I_{on} in strained silicon. The mean thermal injection velocity v_{inj} for the full band structures were computed and it was found that in strained silicon, v_{inj} is only 1% larger in the $\langle 100 \rangle$ direction than in the $\langle 110 \rangle$ direction (in comparison, it was 4% larger in the $\langle 100 \rangle$ direction than the $\langle 110 \rangle$ direction in unstrained Si). In addition, the backscattering coefficient does not depend on the crystallographic in-plane direction because the low-field mobilities are identical and the field is nearly the same.

Therefore, it is the velocity in the source side of the channel gained due to quasiballistic transport rather than the injection velocity that determines the on-current in nanoscale MOSFETs. In a completely analogous way, the failure of such models to explain the on-current can be deduced by comparing the results of the full-band and analytic-band Monte Carlo simulations for the output characteristics in unstrained-Si NMOSFETs [22]. The analytic thermal injection velocity is only 2% greater than in the full-band case, and the low-field mobilities are identical in both models. Whereas the model of Lundstrom [21] also predicts in that case almost the same on-currents, they are actually much higher for the analytic band structure.

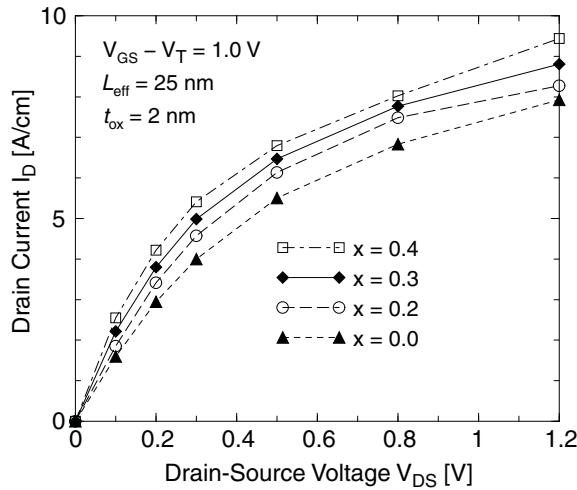
PMOSFETs

In this section, the analogous investigation as in [NMOSFETs on page 80](#) for NMOSFETs is performed for PMOSFETs, that is, the strain-induced performance dependence on scaling is addressed as reported in the literature [6]. In contrast to the case of NMOSFETs, in this study of strained-Si PMOSFETs, the channel direction is fixed along the usual $\langle 110 \rangle$ direction and the level of strain varies, that is, the germanium content in the SiGe substrate, because the significant performance variations concern the strain dependence.

The device structures and simulation methodology are the same as for the NMOSFETs. As already mentioned, all simulations are performed for a channel direction along the usual crystallographic $\langle 110 \rangle$ direction, because the anisotropy in strained-Si PMOSFETs is negligible. While the on-current in the 25 nm PMOSFET is, for unstrained Si, 6% greater in the $\langle 100 \rangle$ direction, the difference in strained Si is already reduced to 1% for $x = 0.2$. It can

be seen in [Figure 21](#) that the drain current for all voltages is continuously increased by strain.

Figure 21 Output characteristics of smallest simulated PMOSFET for unstrained and strained Si corresponding to Ge contents in the substrate of 20%, 30%, and 40%



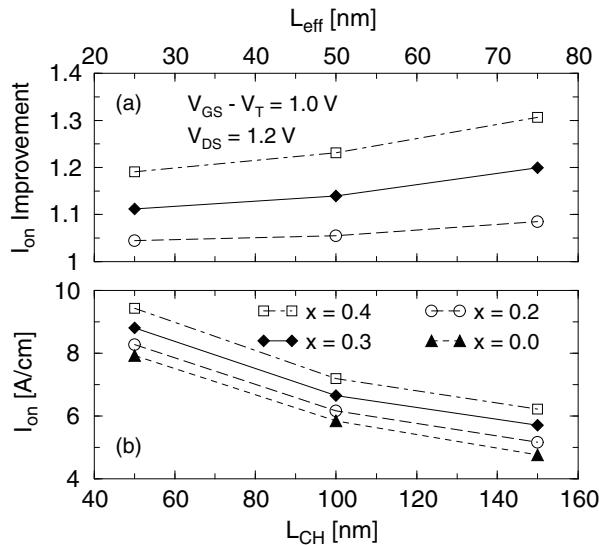
The variation of the on-current as a function of scaling and strain is depicted in [Figure 22](#). The relative strain-induced improvement of the on-current becomes smaller for a decreasing gate length (see [Figure 22 \(a\)](#)). Nevertheless, it can be observed that the reduction of the on-current is relatively modest and the I_{on} enhancement attained for the smallest gate length is still 20% for $x = 0.4$. In particular, the reduction of the I_{on} improvement has been observed [\[23\]](#) where the gate length was scaled from 0.5 μm to 0.1 μm . The dependence of the absolute currents on the gate length in [Figure 22 \(b\)](#) shows that scaling continues to increase the performance for both unstrained and strained silicon. The generally smaller currents in comparison to NMOSFETs are related to the smaller source-side velocities shown in Figure 7 of the literature [\[6\]](#).

In particular, it is interesting to observe that the on-current in the 25 nm strained-Si PMOSFET is smaller than in the corresponding unstrained-Si NMOSFET, even for $x = 0.4$, despite a higher bulk low-field mobility (compare [Figure 19 on page 82](#)). The reason is that the warped band structure of holes leads to a smaller effective channel mobility (compare Figure 4 in the literature [\[23\]](#) with Figure 3 in [\[24\]](#)). On the other hand, the bulk velocity of holes in strained Si with $x = 0.4$ becomes smaller above a field of 8 kV/cm than the electron velocity in unstrained bulk silicon (compare [Figure 16 on page 79](#) with [Figure 14 on page 77](#)). In addition, the bulk velocity overshoot is slightly stronger for electrons in unstrained Si. Therefore, under strain, nanoscale PMOSFET performance will also remain inferior to NMOSFET performance.

Chapter 4: Strained Silicon

Arbitrary Stress

Figure 22 Dependence of on-current on gate length: (a) improvement by strain relative to the unstrained case for different strain levels and (b) absolute values for strained and unstrained silicon



Arbitrary Stress

Besides MOSFETs under biaxial tensile strain as discussed in the previous section, Sentaurus Device Monte Carlo can simulate devices under arbitrary stress. The corresponding band-structure data can be generated with Sentaurus Band Structure (see [Creating Band Data for Sentaurus Device Monte Carlo on page 139](#)), or it can be computed using external programs such as $k \cdot p$ solvers (send an email to support-tcad-eu@synopsys.com).

The single-particle device Monte Carlo solver is instructed to read custom band-structure data by specifying MonteCarloPath in the File section. For example:

```
File {
    MonteCarloPath="/home/lsmith/STRESSTABLE/SPARTA/unil10comp85/"
}
```

Note that directory names for custom band-structure data must contain the substring SPARTA (all uppercase) to be identified correctly by the simulator.

Strain might reduce the symmetry of the band structure. Depending on the strain configuration, band-structure data must be supplied on either one octant of the Brillouin zone (orthorhombic strain) or half the Brillouin zone (for example, uniaxial compressive stress in the <110> direction). A valid set of band-structure data for orthorhombic or higher symmetry consists of an energy file `a.dat` and a group velocity file `b.dat`. In the presence

Chapter 4: Strained Silicon

References

of shear strain, additional files called a1.dat, a2.dat, a3.dat and b1.dat, b2.dat, b3.dat must be supplied.

Sentaurus Device Monte Carlo distinguishes between the two cases by checking for the presence of a1.dat. Sample output for the low symmetry case is shown here:

```
Path is /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/
Entering readina
First: reading /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/
      a1.dat.gz ...
Band structure data for non-orthorhombic symmetry to be imported
reading /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/cirate_taki.dat ...
      done
Entering setdosc finished with the map.
Entering readinb
reading /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/xi.dat ... done
reading /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/rotmat.dat ... done
reading /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/rot.dat ... done
reading /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/a.dat.gz ... done
reading /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/a1.dat.gz ... done
reading /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/a2.dat.gz ... done
reading /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/a3.dat.gz ... done
reading /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/b.dat.gz ... done
reading /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/b1.dat.gz ... done
reading /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/b2.dat.gz ... done
reading /home/lsmith/STRESSTABLE/SPARTA/unil10comp85/b3.dat.gz ... done
```

References

- [1] M. M. Rieger and P. Vogl, "Electronic-band parameters in strained $\text{Si}_{1-x}\text{Ge}_x$ alloys on $\text{Si}_{1-y}\text{Ge}_y$ substrates," *Physical Review B*, vol. 48, no. 19, pp. 14276–14287, 1993.
- [2] J. P. Dismukes, L. Ekstrom, and R. J. Paff, "Lattice Parameter and Density in Germanium–Silicon Alloys," *The Journal of Physical Chemistry*, vol. 68, no. 10, pp. 3021–3027, 1964.
- [3] C. Jacoboni and L. Reggiani, "The Monte Carlo method for the solution of charge transport in semiconductors with applications to covalent materials," *Reviews of Modern Physics*, vol. 55, no. 3, pp. 645–705, 1983.
- [4] F. M. Bufler, S. Keith, and B. Meinerzhagen, "Anisotropic Ballistic In-Plane Transport of Electrons in Strained Si," in *International Conference on Simulation of Semiconductor Processes and Devices (SiSPAD)*, Leuven, Belgium, pp. 239–242, September 1998.
- [5] F. M. Bufler and W. Fichtner, "Scaling of Strained-Si n-MOSFETs Into the Ballistic Regime and Associated Anisotropic Effects," *IEEE Transactions on Electron Devices*, vol. 50, no. 2, pp. 278–284, 2003.

Chapter 4: Strained Silicon

References

- [6] F. M. Bufler and W. Fichtner, "Scaling and Strain Dependence of Nanoscale Strained-Si p-MOSFET Performance," *IEEE Transactions on Electron Devices*, vol. 50, no. 12, pp. 2461–2466, 2003.
- [7] M. T. Currie *et al.*, "Carrier mobilities and process stability of strained Si n- and p-MOSFETs on SiGe virtual substrates," *Journal of Vacuum Science & Technology B*, vol. 19, no. 6, pp. 2268–2279, 2001.
- [8] C. Canali, G. Ottaviani, and A. Alberigi Quaranta, "Drift Velocity of Electrons and Holes and Associated Anisotropic Effects in Silicon," *Journal of Physics and Chemistry of Solids*, vol. 32, pp. 1707–1720, 1971.
- [9] F. M. Bufler *et al.*, "Full band Monte Carlo investigation of electron transport in strained Si grown on $\text{Si}_{1-x}\text{Ge}_x$ substrates," *Applied Physics Letters*, vol. 70, no. 16, pp. 2144–2146, 1997.
- [10] K. Ismail *et al.*, "Electron transport properties of Si/SiGe heterostructures: Measurements and device implications," *Applied Physics Letters*, vol. 63, no. 5, pp. 660–662, 1993.
- [11] A. Sadek and K. Ismail, "Si/SiGe CMOS Possibilities," *Solid-State Electronics*, vol. 38, no. 9, pp. 1731–1734, 1995.
- [12] Th. Vogelsang and K. R. Hofmann, "Electron transport in strained Si layers on $\text{Si}_{1-x}\text{Ge}_x$ substrates," *Applied Physics Letters*, vol. 63, no. 2, pp. 186–188, 1993.
- [13] B. Fischer and K. R. Hofmann, "Full-band Monte Carlo model of electron and hole transport in strained Si including inelastic acoustic phonon scattering," *Applied Physics Letters*, vol. 74, no. 15, pp. 2185–2187, 1999.
- [14] H. Miyata, T. Yamada, and D. K. Ferry, "Electron transport properties of a strained Si layer on a relaxed $\text{Si}_{1-x}\text{Ge}_x$ substrate by Monte Carlo simulation," *Applied Physics Letters*, vol. 62, no. 21, pp. 2661–2663, 1993.
- [15] B. Fischer, *A Full-Band Monte Carlo Charge Transport Model for Nanoscale Silicon Devices Including Strain*, Aachen: Shaker, 2000.
- [16] C. W. Leitz *et al.*, "Hole mobility enhancements and alloy scattering-limited mobility in tensile strained Si/SiGe surface channel metal–oxide–semiconductor field-effect transistors," *Journal of Applied Physics*, vol. 92, no. 7, pp. 3745–3751, 2002.
- [17] P. Houlet, Y. Awano, and N. Yokoyama, "Influence of non-linear effects on very short pMOSFET device performances," *Physica B*, vol. 272, pp. 572–574, 1999.
- [18] K. Rim *et al.*, "Strained Si NMOSFETs for High Performance CMOS Technology," in *Symposium on VLSI Technology*, Kyoto, Japan, pp. 59–60, June 2001.
- [19] C. G. Van de Walle, "Band lineups and deformation potentials in the model-solid theory," *Physical Review B*, vol. 39, no. 3, pp. 1871–1883, 1989.
- [20] A. Lohctefeld and D. A. Antoniadis, "Investigating the Relationship Between Electron Mobility and Velocity in Deeply Scaled NMOS via Mechanical Stress," *IEEE Electron Device Letters*, vol. 22, no. 12, pp. 591–593, 2001.

Chapter 4: Strained Silicon

References

- [21] M. Lundstrom, "Elementary Scattering Theory of the Si MOSFET," *IEEE Electron Device Letters*, vol. 18, no. 7, pp. 361–363, 1997.
- [22] F. M. Bufler *et al.*, "Monte Carlo Simulation and Measurement of Nanoscale n-MOSFETs," *IEEE Transactions on Electron Devices*, vol. 50, no. 2, pp. 418–424, 2003.
- [23] S. Keith, C. Jungemann, and B. Meinerzhagen, "Full Band Monte Carlo Device Simulation of 0.1 – 0.5 μm Strained-Si P-MOSFETs," in *Proceedings of the 28th European Solid-State Device Research Conference (ESSDERC)*, Bordeaux, France, vol. 28, pp. 312–315, 1998.
- [24] S. Keith, F. M. Bufler, and B. Meinerzhagen, "Full Band Monte-Carlo Device Simulation of an 0.1 μm N-Channel MOSFET in Strained Silicon Material," in *Proceedings of the 27th European Solid-State Device Research Conference (ESSDERC)*, Stuttgart, Germany, vol. 27, pp. 200–203, September 1997.

5

Surface and Channel Orientations

This chapter discusses surface and channel orientations with respect to increasing the performance of PMOSFETs.

Changing Surface and Channel Orientations

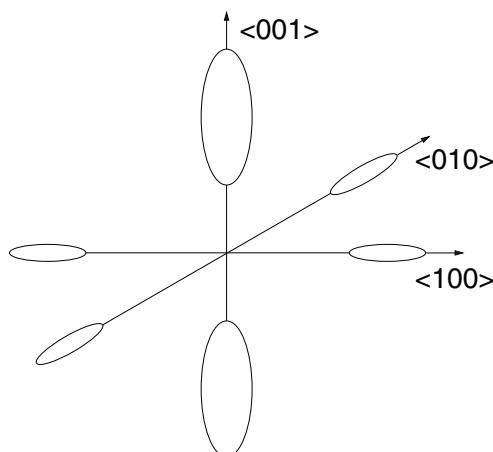
Apart from stress engineering, changing surface and channel orientations is another possibility to increase the performance of PMOSFETs [1][2]. This effect can be investigated in Sentaurus Device Monte Carlo by specifying the crystallographic orientations for the surface and channel directions. Its use in the command file is, for example:

```
ChannelDirection = ( 0 -1 1 )
Normal2OxideDirection = ( 0 1 1 )
```

See [MonteCarlo Section on page 38](#) for details about these keywords.

The band structure table that is used in the tool is always stored with respect to a Cartesian coordinate system with the three axes aligned to the principal axes of the six ellipsoids as illustrated in [Figure 23](#).

Figure 23 *Equienergy surface in the first conduction band of silicon under uniaxial compressive stress in <001> direction*



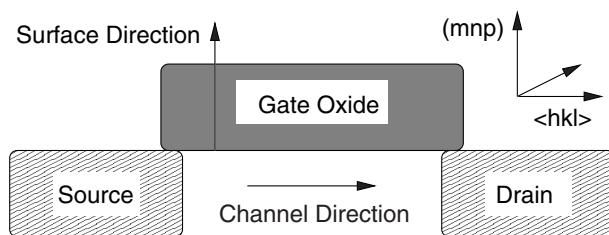
Chapter 5: Surface and Channel Orientations

Changing Surface and Channel Orientations

This example corresponds to uniaxial compressive stress in the $<001>$ direction and lifts four of the six conduction band valleys upwards in energy. Specifying the channel and surface orientations of a device entails orientating the device correspondingly, relative to the band structure in [Figure 23](#).

In [Figure 24](#), a schematic of a bulk MOSFET is shown where the channel direction is in the crystallographic $<\text{hkl}>$ orientation and the surface direction is in the crystallographic (mnp) orientation, with the directional vectors $<\text{hkl}>$ and (mnp) to be placed into [Figure 23](#).

Figure 24 Specification of crystallographic surface (mnp) and channel $<\text{hkl}>$ orientation in schematic MOSFET



Note that, in 2D device simulation, the channel direction corresponds to the x -axis and the surface direction to the $-y$ -axis. The corresponding transformation is performed internally. The specification of `ChannelDirection` and `Normal2OxideDirection` in the command file of Sentaurus Device Monte Carlo must always be specified according to the orientations as shown in [Figure 24](#).

[Figure 25](#) shows a comparison between measurements [1][2] and single-particle device Monte Carlo simulations of the enhancements to the effective hole mobility due to different surface and channel directions, relative to the standard (001) surface and $<110>$ channel orientations.

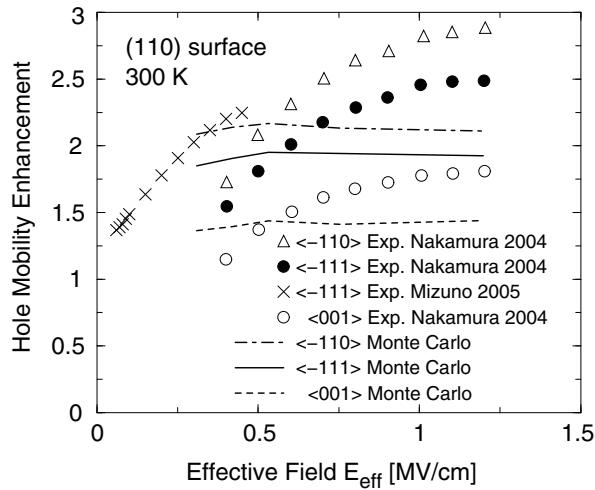
The simulated order of the enhancement and the dependence on the channel orientation are in satisfactory agreement with the measurements in view of the differences between different measurements. The physical reason for the enhancement of the hole surface mobility for a (110) surface orientation is specular scattering at the interface, which is governed by the conservation of energy and parallel wavevector. In the standard configuration, specular scattering can revert the direction of the parallel component of the group velocity and, therefore, degrades the mobility [3], whereas for a (110) surface orientation, this effect does not occur due to the orientation of the band structure relative to the interface [4].

In contrast, in the case of electrons, changing the standard configuration degrades the surface mobility. However, the effect appears to be underestimated in the simulation and might require calibration, that is, it might be necessary to increase the percentage of diffusive scattering upon surface roughness scattering.

Chapter 5: Surface and Channel Orientations

Changing Surface and Channel Orientations

Figure 25 Long-channel effective hole mobility enhancement for (110) surface and different $\langle hkl \rangle$ channel orientations



Another application of arbitrary surface and channel orientations is the possibility to use one band structure table of strained silicon *more than once*, the computation of which is time-consuming with the pseudopotential method and which requires considerable disk space. For example, using the band structure table for the uniaxial compressive stress in the <001> direction illustrated in Figure 23, together with the (001) surface and <100> channel orientations, corresponds to uniaxial stress that is perpendicular to the gate oxide interface, while using the same table with the (-100) surface and <001> channel orientations corresponds to uniaxial stress in the channel direction.

TCAD Sentaurus Tutorial: Simulation Projects

A corresponding Sentaurus Workbench project can be accessed from the TCAD Sentaurus Tutorial (HTML training material).

To access the TCAD Sentaurus Tutorial:

1. Open Sentaurus Workbench by entering the following on the command line: `swb`
2. From the menu bar of Sentaurus Workbench, choose **Help > Training** or click on the toolbar.
3. Click the **Sentaurus Device** module.
4. Click Section 2, **Carrier Transport Models**.
5. Click **Monte Carlo Transport**.

The project can be downloaded from here.

Chapter 5: Surface and Channel Orientations

References

Alternatively, to access the TCAD Sentaurus Tutorial:

1. Go to the directory `$STROOT/tcad/current/Sentaurus_Training`.

The `STROOT` environment variable indicates where the Synopsys TCAD distribution has been installed.

2. Open the `index.html` file in your browser.
3. Click the **Sentaurus Device** module.
4. Click Section 2, **Carrier Transport Models**.
5. Click **Monte Carlo Transport**.

The project can be downloaded from here.

References

- [1] H. Nakamura *et al.*, "Effects of Selecting Channel Direction in Improving Performance of Sub-100 nm MOSFETs Fabricated on (110) Surface Si Substrate," *Japanese Journal of Applied Physics*, vol. 43, no. 4B, pp. 1723–1728, 2004.
- [2] T. Mizuno *et al.*, "(110)-Surface Strained-SOI CMOS Devices," *IEEE Transactions on Electron Devices*, vol. 52, no. 3, pp. 367–374, 2005.
- [3] C. Jungemann, S. Keith, and B. Meinerzhagen, "Full-Band Monte Carlo Simulation of a $0.12\mu\text{m}$ -Si-PMOSFET with and without a Strained SiGe-Channel," in *IEDM Technical Digest*, San Francisco, CA, USA, pp. 897–900, December 1998.
- [4] F. M. Bufler and A. Erlebach, "Monte Carlo Simulation of the Performance Dependence on Surface and Channel Orientation in Scaled pFinFETs," in *Proceedings of the 36th European Solid-State Device Research Conference (ESSDERC)*, Montreux, Switzerland, pp. 174–177, September 2006.

6

Stress-Dependent Built-in Analytic Band Structures

This chapter describes the analytic band structures that are incorporated in single-particle device Monte Carlo for strained silicon.

Stress Engineering

Stress engineering requires the frequent computation of the corresponding band structures. Since the accurate pseudopotential calculations are very CPU-intensive, the tool offers as a third possibility – besides using the supplied pseudopotential band tables and the capability to import external band tables – to use analytic band structures. The corresponding formulas are incorporated in the tool and are evaluated for the symmetric strain tensor, which must be specified using the variable `MCStrain` in the Sentaurus Device Monte Carlo command file. Its use in the command file is:

$$\text{MCStrain} = (\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}, \varepsilon_{yz}, \varepsilon_{xz}, \varepsilon_{xy}) \quad (23)$$

Here:

$$\varepsilon_{ij} = \frac{1}{2} \left(\frac{du_i}{dx_j} + \frac{du_j}{dx_i} \right) \quad (24)$$

is the symmetric strain tensor in the Cartesian coordinate system, where the three axes are aligned to the principal axes of the three pairs of ellipsoids in the first conduction band, that is, in the same coordinate system as used for the band structure tables in [Chapter 5 on page 90](#). \mathbf{u} is the displacement vector. The relation of the symmetric strain tensor to the geometry of the device is specified by `ChannelDirection` and `Normal2OxideDirection` (see [Chapter 5 on page 90](#) for a more detailed description).

Note:

The symmetric strain tensor differs from the so-called engineering or conventional strain tensor appearing in Hooke's law, which relates the engineering strain tensor to the stress tensor. The off-diagonal components of the engineering strain tensor are larger than the off-diagonal components of the symmetric strain tensor by a factor of two.

Chapter 6: Stress-Dependent Built-in Analytic Band Structures

Visualizing Band Structures

Another alternative option to performing stress-dependent MC simulations is to use the stress tensor that is defined in the `Piezo` section of the Sentaurus Device command file. This option should be simpler than `MCStrain` for users because usage of the stress tensor in the `Piezo` section is a typical way to perform stress-dependent drift-diffusion device simulations.

In addition, unlike `MCStrain`, the stress tensor should be defined in the device simulation coordinate system, and it can be simply taken from the process simulation without any transformation. To activate this option, you must specify the keyword `ReadinStress`. In this case, Sentaurus Device Monte Carlo automatically transforms the stress tensor from the device simulation coordinate system to the crystal one, and computes the strain tensor above using the elasticity constants defined in the `LatticeParameters` section of the Sentaurus Device parameter file.

This option also defines the `ChannelDirection` and `Normal2OxideDirection` vectors automatically using the corresponding `x` and `y` vectors from the `LatticeParameters` section (these vectors set the device simulation coordinate system).

Note:

When you specify `ReadinStress`, the `MCStrain`, `ChannelDirection`, and `Normal2OxideDirection` specifications are ignored with warnings printed in the log file.

Since the influence of the position dependence of stress has been found to be negligible for gate lengths below 0.1 μm [1], the analytic band structures allow for using Monte Carlo simulation directly for stress engineering in a Sentaurus Workbench project, where the strain or stress tensors are picked from the source side of the channel.

Visualizing Band Structures

The new analytic band models for electrons and holes, which are activated by specifying `MCStrain` in the Sentaurus Device Monte Carlo command file, can also be visualized along a line in the Brillouin zone by giving the initial and final wavevectors of the line, for example:

$$\begin{aligned} \text{KVecStart} &= (0.0, 0.0, 0.85) \\ \text{KVecEnd} &= (0.2, 0.2, 0.85) \end{aligned} \tag{25}$$

which will generate a file with the suffix `_energy.plt` showing the band energies along the line between both wavevectors (the parameter on the x-coordinate varies between 0 and 1, and the unit of the wavevectors is $2\pi/a$ with a denoting the lattice constant).

Hole Band Structure

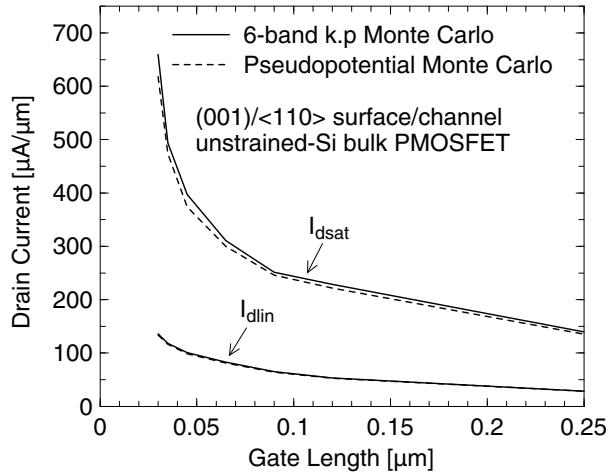
For holes, analytic solutions of the six-band $k \cdot p$ Hamiltonian are used [2]. The values used for the valence-band parameters and for the deformation potentials are shown in [Table 3](#) together with the elastic constants in silicon. The value of the spin-orbit splitting, $\Delta_{\text{so}} = 0.043409$ eV, results from the pseudopotential calculation when setting $\mu = 0.00018$ Ry and $\zeta = 4.6 \text{ \AA}^{-1}$ in the literature [3].

Table 3 Valence-band parameters, deformation potentials, and elastic constants used for silicon

Parameter	Unit	Value	Reference
L	$\hbar^2/(2m_0)$	-6.69	[3]
M	$\hbar^2/(2m_0)$	-4.62	[3]
N	$\hbar^2/(2m_0)$	-8.56	[3]
a_v	eV	2.46	[4]
b	eV	-2.35	[5]
d	eV	-5.32	[5]
C_{11}	Mbar	1.675	[3]
C_{12}	Mbar	0.65	[3]
C_{44}	Mbar	0.8	[3]

In [Figure 26](#), the drain currents in unstrained-silicon PMOSFETs based on the six-band $k \cdot p$ band structure are compared to the corresponding results using the pseudopotential band structure. The small differences show that six-band $k \cdot p$ band structures are well suited to drain current simulations where the high-energy regime is not important.

Figure 26 Monte Carlo simulation of saturation ($V_{DS} = -1.1$ V) and linear ($V_{DS} = -0.1$ V) drain currents in scaled PMOSFETs as a function of the gate length with a band structure obtained either from the nonlocal pseudopotential method [3] or by six-band $k \cdot p$ calculation



Electron Band Structure

For electrons, it had been found that the standard analytic band model, which is based on different values for the longitudinal and the transverse mass and uses an isotropic nonparabolicity factor [6], overestimates the drain current at short gate lengths and high drain voltages [7] and does not include the anisotropy of quasiballistic overshoot in biaxially strained silicon [8]. Therefore, the standard analytic band model has been extended to overcome these two limitations [9]. In addition, the influence of stress is considered in terms of valley splitting [10] and a change of effective mass in the case of a nondiagonal strain tensor [11].

For a given valley, the electron energy is decomposed according to:

$$E_c(\mathbf{k}) = E_{c,0} + E_l(\mathbf{k}) + E_t(\mathbf{k}) \quad (26)$$

where $E_{c,0}$ is the valley minimum. The longitudinal and transverse parts of the electron energy are given by:

$$E_l(\mathbf{k}) = \frac{\hbar^2}{2m_l} (k_l - k_{l,0})^2 \quad (27)$$

$$E_t(\mathbf{k}) = \frac{\sqrt{1 + 4\beta(\vec{k}_t)C(\vec{k}_t)} - 1}{2\beta(\vec{k}_t)} \quad (28)$$

Chapter 6: Stress-Dependent Built-in Analytic Band Structures

Electron Band Structure

respectively, with:

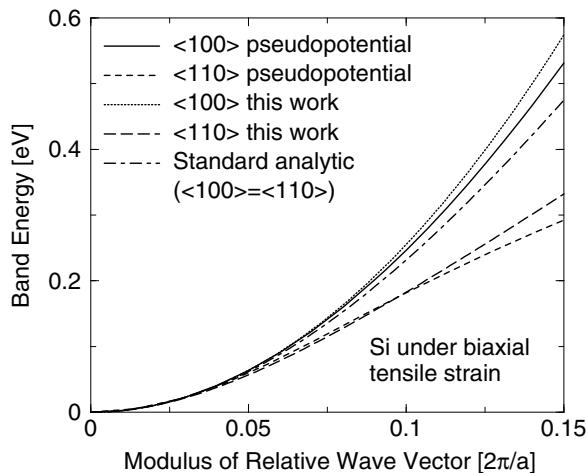
$$\beta(\vec{k}_t) = 4 \frac{k_y^2 k_z^2}{(k_y^2 + k_z^2)^2} \frac{\left(1 - \frac{m_t}{m_0}\right)^2}{\Delta E_0} \quad (29)$$

$$C(\vec{k}_t) = \frac{\hbar^2 k_t^2}{2m_t} \quad (30)$$

$$k_t^2 = k_y^2 + k_z^2 \quad (31)$$

when taking the longitudinal direction along the k_x -axis. m_l and m_t denote the longitudinal effective mass and the transverse effective mass, respectively, and m_0 is the free electron mass. The expansion of the square root in [Equation 28](#) leads, in second order, to the anisotropic nonparabolic transverse energy in Appendix B of [\[6\]](#) and represents the corresponding generalization to the standard nonparabolic expression. Here, however, no averaging is performed, so that the nonparabolicity factor is not an isotropic constant, but is given by [Equation 29](#). Since [Equation 28](#) is used beyond the validity of the expansion to second order, ΔE_0 loses its original meaning and is fitted to the pseudopotential energy dispersion with the result shown in [Figure 27](#).

Figure 27 Electron energy in a lower valley of biaxially strained silicon grown on $Si_{0.8}Ge_{0.2}$ versus modulus of the wavevector (measured with respect to the valley minimum) along two directions perpendicular to the longitudinal axis of the equienergy ellipsoid



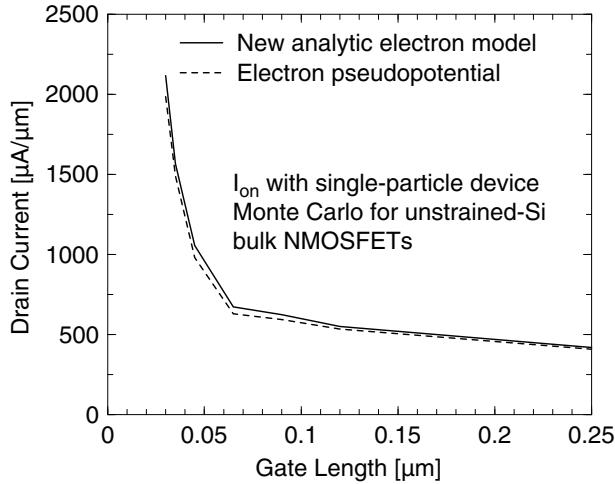
In addition, a second conduction band is introduced in the spirit of the extended zone scheme of the free electron model, where the conduction band minimum is correspondingly located at $k_{c,0} = 1.15 \times 2\pi/a$ (a is the lattice constant), resulting from a shift by a reciprocal lattice vector. The values for its effective masses are adjusted to compensate for the underestimation of the full-band density-of-states above 130 meV by the standard analytic band model. The comparison of the on-current scaling in [Figure 28](#) when using either the

Chapter 6: Stress-Dependent Built-in Analytic Band Structures

Electron Band Structure

pseudopotential or the new analytic electron band model now shows only a small overestimation of the drain current by the new analytic model.

Figure 28 Monte Carlo simulation of saturation ($V_{DS} = 1.1$ V) drain current in scaled NMOSFETs as a function of gate length with a band structure obtained from either the nonlocal pseudopotential method [3] or the new analytic electron band model



Under strain, the energy shift of, for example, the valley along the x-axis is given by [10]:

$$\Delta\epsilon_{c,0}^{(x)} = \Xi_d(\epsilon_{xx} + \epsilon_{yy} + \epsilon_{zz}) + \Xi_u\epsilon_{xx} + \Delta\epsilon_{\text{shear}}$$

$$\Delta\epsilon_{\text{shear}} = \begin{cases} -\epsilon_{yz}^2\Delta/(4\kappa^2) & , |\epsilon_{yz}| < \kappa \\ -(2|\epsilon_{yz}/\kappa| - 1)\Delta/4 & , \text{else} \end{cases} \quad (32)$$

with $\Xi_u = 9.29$ eV, $\kappa = \Delta/(4\Xi_u)$, $\Delta = 0.53$ eV, and $\Xi_u' = 7.0$ eV [12]. In addition, a nondiagonal symmetric strain tensor leads to a change of the effective mass according to [11]:

$$C(\vec{k}_t) \rightarrow C(\vec{k}_t) + A\epsilon_{yz}k_yk_z \quad (33)$$

and to a change of the nonparabolicity factor according to [13]:

$$\beta(\vec{k}_t) \rightarrow \beta(\vec{k}_t) \times \frac{1 + 2(\eta/1.2)^2}{1 - (\eta/1.2)^2} \quad (34)$$

with $\eta = \epsilon_{yz}/\kappa$. All values of the new analytic band model are given in Table 4.

Chapter 6: Stress-Dependent Built-in Analytic Band Structures

Electron Band Structure

Table 4 Electron band parameters

	$k_{l,0}$	m_l	m_t	A	ΔE_0
Unit	$(2\pi/a_l)$	(m_0)	(m_0)	$(1/m_0)$	(eV)
First conduction band	0.85	0.918	0.197	173.6	0.3
Second conduction band	1.15	1.2	0.4	173.6	–

Finally, simulation results of the new analytic electron band model are compared to pseudopotential results for the low-field mobility in [Figure 29](#) and for the anisotropic transient velocity overshoot in [Figure 30](#).

Good agreement between the results of the new analytic electron band model and the pseudopotential approach can be seen that suggests its usability for drain current simulations.

Figure 29 Low-field drift mobility of electrons under biaxial or uniaxial <110> tensile stress; in the case of biaxial stress, the symbols correspond to substrate Ge contents of 0%, 10%, 20%, and 30%; pseudopotential results are from [14]

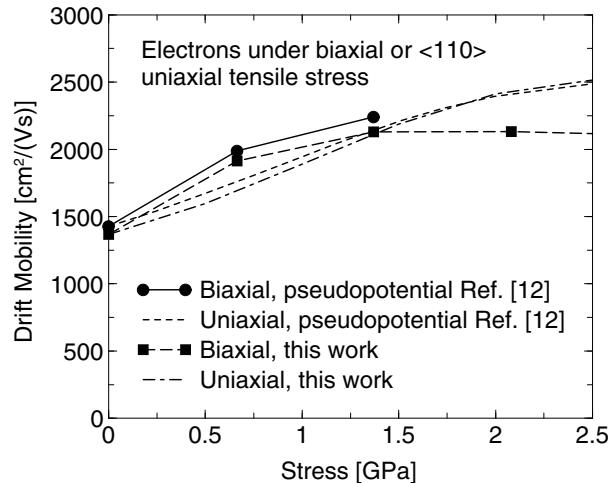
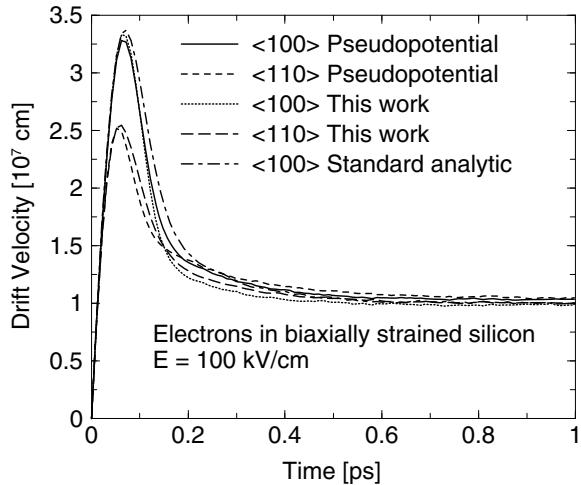


Figure 30 Transient in-plane velocity overshoot after a sudden application of a 100 kV/cm field to electrons in biaxially strained silicon grown on $\text{Si}_{0.8}\text{Ge}_{0.2}$



References

- [1] F. M. Bufler and R. Gautschi, "On the Influence of the Position-Dependence of Stress on Device Performance," in *ECS Transactions*, vol. 3, no. 7, Cancun, Mexico, pp. 439–442, October 2006.
- [2] F. M. Bufler, A. Tsibizov, and A. Erlebach, "Scaling of Bulk pMOSFETs: (110) Surface Orientation Versus Uniaxial Compressive Stress," *IEEE Electron Device Letters*, vol. 27, no. 12, pp. 992–994, 2006.
- [3] M. M. Rieger and P. Vogl, "Electronic-band parameters in strained $\text{Si}_{1-x}\text{Ge}_x$ alloys on $\text{Si}_{1-y}\text{Ge}_y$ substrates," *Physical Review B*, vol. 48, no. 19, pp. 14276–14287, 1993.
- [4] C. G. Van de Walle, "Band lineups and deformation potentials in the model-solid theory," *Physical Review B*, vol. 39, no. 3, pp. 1871–1883, 1989.
- [5] C. G. Van de Walle and R. M. Martin, "Theoretical calculations of heterojunction discontinuities in the Si/Ge system," *Physical Review B*, vol. 34, no. 8, pp. 5621–5634, 1986.
- [6] C. Jacoboni and L. Reggiani, "The Monte Carlo method for the solution of charge transport in semiconductors with applications to covalent materials," *Reviews of Modern Physics*, vol. 55, no. 3, pp. 645–705, 1983.
- [7] F. M. Bufler *et al.*, "Monte Carlo Simulation and Measurement of Nanoscale n-MOSFETs," *IEEE Transactions on Electron Devices*, vol. 50, no. 2, pp. 418–424, 2003.

Chapter 6: Stress-Dependent Built-in Analytic Band Structures

References

- [8] F. M. Bufler, S. Keith, and B. Meinerzhagen, "Anisotropic Ballistic In-Plane Transport of Electrons in Strained Si," in *International Conference on the Simulation of Semiconductor Processes and Devices (SISPAD)*, Leuven, Belgium, pp. 239–242, September 1998.
- [9] F. M. Bufler and A. Tsibizov, "Analytical Electron Band Model with Anisotropic Nonparabolicity for Strained Silicon," in *12th International Workshop on Computational Electronics (IWCE)*, University of Massachusetts, Amherst, USA, pp. 100–101, October 2007.
- [10] I. Balslev, "Influence of Uniaxial Stress on the Indirect Absorption Edge in Silicon and Germanium," *Physical Review*, vol. 143, no. 2, pp. 636–647, 1966.
- [11] J. C. Hensel, H. Hasegawa, and M. Nakayama, "Cyclotron Resonance in Uniaxially Stressed Silicon. II. Nature of the Covalent Bond," *Physical Review*, vol. 138, no. 1A, pp. A225–A238, 1965.
- [12] E. Ungersboeck *et al.*, "The Effect of General Strain on the Band Structure and Electron Mobility of Silicon," *IEEE Transactions on Electron Devices*, vol. 54, no. 9, pp. 2183–2190, 2007.
- [13] V. A. Sverdlov *et al.*, "Influence of Uniaxial [110] Stress on the Silicon Conduction Band Structure: Stress Dependence of the Nonparabolicity Parameter," in *International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, Vienna, Austria, pp. 329–332, September 2007.
- [14] E. Ungersboeck *et al.*, "Physical Modeling of Electron Mobility Enhancement for Arbitrarily Strained Silicon," in *11th International Workshop on Computational Electronics (IWCE)*, Vienna, Austria, pp. 141–142, May 2006.

7

Electron and Hole Transport

This chapter discusses electron and hole transport that can be simulated using Sentaurus Device Monte Carlo.

Electron and Hole Transport in Strained SiGe

This section describes the features of electron and hole transport in strained silicon germanium (SiGe) that can be simulated using Sentaurus Device Monte Carlo.

Hole Band Structure

The full band structure under consideration consists of three valence bands and is obtained from nonlocal empirical pseudopotential calculations in the virtual crystal approximation including spin-orbit interaction [1]. In addition to [1], the spin-orbit interaction in silicon is taken into account by setting the corresponding parameters in Table I of [1] to $\mu = 0.00018$ and $\zeta = 8.6927 \text{ \AA}^{-1}$.

Strain is due to the lattice mismatch between the $\text{Si}_{1-x}\text{Ge}_x$ alloy and the (100) $\text{Si}_{1-y}\text{Ge}_y$ substrate. The lattice constant of the unstrained SiGe alloy can be parameterized by [1][2]:

$$a_0(x) = a_{0,\text{Si}} + 0.200326x(1-x)\text{\AA} + (a_{0,\text{Ge}} - a_{0,\text{Si}})x^2 \quad (35)$$

with the bulk lattice constants in Si and Ge being $a_{0,\text{Si}} = 5.43 \text{ \AA}$ and $a_{0,\text{Ge}} = 5.65 \text{ \AA}$. Below a critical thickness, the $\text{Si}_{1-x}\text{Ge}_x$ layer is biaxially strained. Its lattice constants parallel (a_{\parallel}) and perpendicular (a_{\perp}) to the interface, between the layer and the substrate, then follow from elasticity theory with the result [1]:

$$\begin{aligned} a_{\parallel} &= a_0(y) \\ a_{\perp} &= a_0(x) \left[1 - 2 \frac{c_{12}(x)}{c_{11}(x)} \frac{a_{\parallel} - a_0(x)}{a_0(x)} \right] \end{aligned} \quad (36)$$

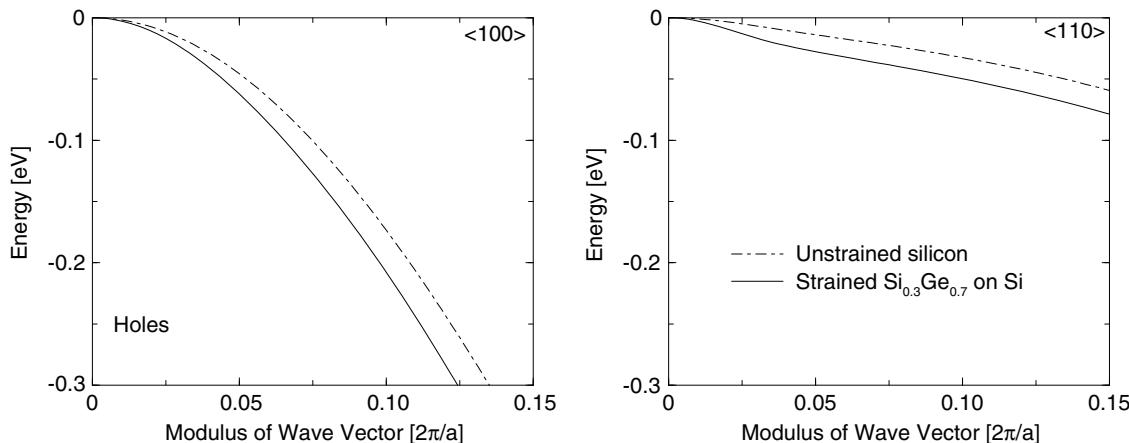
The elastic constants in Si (Ge) are $c_{11} = 1.675$ (1.315) Mbar and $c_{12} = 0.650$ (0.494) Mbar. Their values in the alloy are obtained by linear interpolation. Strain lifts the degeneracy of heavy-hole and light-hole bands at the Γ -point, but the heavy-hole band remains at the valence-band edge under biaxial compressive strain ($x > y$).

Chapter 7: Electron and Hole Transport

Electron and Hole Transport in Strained SiGe

In addition to this band-splitting, however, the effective mass of the hole band at the valence-band edge, which is decisive for transport, is reduced with increasing compressive strain. This is illustrated by the corresponding band curvatures in [Figure 31](#).

Figure 31 Energy dispersion of the topmost valence band along the (left) <100> and (right) <110> wavevector directions for unstrained Si and strained Si_{0.7}Ge_{0.3} grown on a Si substrate



In the case of the six-band $k \cdot p$ band structure, [Table 5](#) lists the values used for the valence-band parameters and for the deformation potentials in germanium, with the elastic constants, in analogy to the values for silicon in [Table 3 on page 96](#). The value for the spin-orbit splitting, $\Delta_{SO} = 0.29713214$ eV, results from a pseudopotential calculation with $\mu = 0.000965$ Ry and $\zeta = 10.0911 \text{ \AA}^{-1}$ in [\[1\]](#).

Table 5 Valence-band parameters, deformation potentials, and elastic constants used for germanium

Parameter	Unit	Value	Reference
L	$\hbar^2/(2m_0)$	-21.65	[1]
M	$\hbar^2/(2m_0)$	-5.02	[1]
N	$\hbar^2/(2m_0)$	-23.48	[1]
a _v	eV	1.24	[3]
b	eV	-2.55	[4]
d	eV	-5.50	[4]

Table 5 Valence-band parameters, deformation potentials, and elastic constants used for germanium (Continued)

Parameter	Unit	Value	Reference
C ₁₁	Mbar	1.315	[1]
C ₁₂	Mbar	0.494	[1]
C ₄₄	Mbar	0.668	[1]

The valence-band parameters L, M, and N in the Si_{1-x}Ge_x alloy are computed according to the expression given in [1], and the other parameters are obtained by linear interpolation between the values in Si and Ge.

Scattering Mechanisms

The scattering mechanisms included in the microscopic transport model are scattering by:

- Si-type and Ge-type phonons
- Alloy scattering in the formulation of Harrison and Hauser [5]
- Impurity scattering

Phonon scattering in SiGe is approximated for the transport applications by considering the scattering rates in Si and Ge, weighted by the relative fraction of the components in the alloy [1], and includes inelastic acoustic phonons as well as optical phonons [3]. The scattering rate for alloy scattering is:

$$S_{\text{alloy}}(\epsilon) = \frac{\pi}{\hbar}(1-x)x\Omega_{\text{cell}}U^2D(\epsilon) \quad (37)$$

where $\Omega_{\text{cell}} = (a_{\perp}a_{\parallel}^2)/4$ is the unit cell, and $D(\epsilon) = \sum_n D_n(\epsilon)$ is the density-of-states per spin of all three valence bands.

An important uncertainty of the transport model is the strength of alloy scattering. The underlying model of Harrison and Hauser [5] is supported by the old mobility measurements of Busch and Vogt [4] that, while being affected by a large scatter of the data and uncertainties regarding the extraction of the drift mobilities from the measurements, extend over the whole range of Ge contents and exhibit a Ge content dependency of the mobility in accordance with the model of Harrison and Hauser.

In this model, the alloy potential is the only free parameter of the microscopic SiGe transport model. Its value is crucial for transport in SiGe and, therefore, has been adjusted on the basis of comprehensive and accurate drift mobility measurements at 300 K in unstrained and low-doped SiGe alloys [6] with Ge concentrations varying between 0% and 13% with

the result $U = 0.7$ eV. The only change of the impurity scattering rate concerns the relative dielectric constant, which is interpolated linearly between the values of 11.7 for silicon and 16.0 for germanium.

Simulation Procedure and Results

Simulating $\text{Si}_{1-x}\text{Ge}_x$ involves specifying the Ge content using `xGe` in the Sentaurus Device Monte Carlo command file, which defines the contribution of Ge phonon scattering and alloy scattering. Either the corresponding pseudopotential band-structure table is loaded ($\text{Si}_{0.7}\text{Ge}_{0.3}$ under biaxial compressive strain when specifying `xGe=30.0` and `WithStrainedMaterial` or relaxed Ge when specifying `xGe=100.0`), or the analytic six-band $k \cdot p$ band structure can be used for which, in addition to `xGe`, the strain tensor `MCStrain` must be given. This means, for example, that specifying `xGe=30.0` with `MCStrain=(0,0,0,0,0,0)` corresponds to unstrained SiGe, while `MCStrain=(-0.0113,-0.0113,0.0087,0.0,0.0,0.0)` corresponds to SiGe under biaxial compressive strain.

Note:

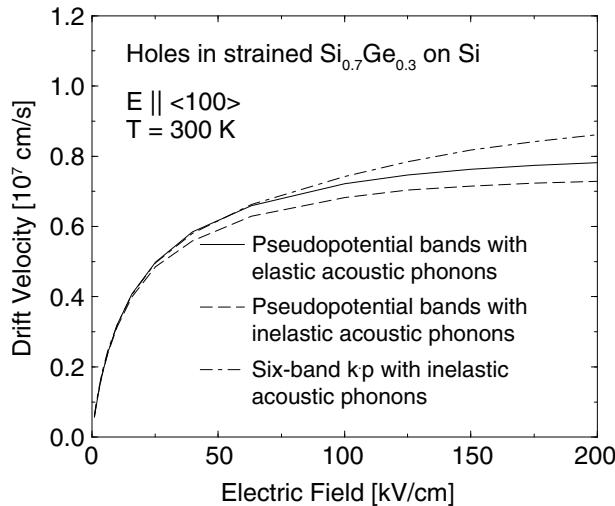
The calibration of the doping-dependent prefactor of the impurity scattering rate and the diffusive percentage for surface roughness scattering was performed for silicon and has not been changed for germanium. Simulating germanium or germanium-rich SiGe alloys requires the use of a screening length computed according to a heated Fermi–Dirac distribution (`WithFermiDiracScreening` activated, which is the default), because in this case the prefactor deviates only moderately from unity and, possibly, requires changing the default value for `SurfScattRatio` in the Sentaurus Device Monte Carlo command file.

In [Figure 32](#), the velocity field characteristics are shown for strained $\text{Si}_{0.7}\text{Ge}_{0.3}$ grown on a Si substrate resulting from:

- Pseudopotential bands and elastic acoustic phonons [\[7\]](#)
- Pseudopotential bands and inelastic acoustic phonons
- Six-band $k \cdot p$ band structure and inelastic acoustic phonons

As expected, the consideration of inelastic phonons results in a smaller saturation velocity [\[3\]](#), and a six-band $k \cdot p$ band structure involves an underestimation of the drift velocity at higher fields, which are, however, not important for the drain current [\[8\]](#).

Figure 32 Simulated in-plane velocity-field characteristics with the field parallel to the $<100>$ direction of holes at 300 K for strained $\text{Si}_{0.7}\text{Ge}_{0.3}$ grown on a Si substrate



Electron Transport

The scattering mechanisms and the phonon-coupling constants for electrons in silicon as well as for electrons in germanium are the same as given by Jacoboni and Reggiani [9]. The only changed values are those for intravalley scattering in order to reproduce the velocity-field characteristics with the pseudopotential band structure instead of the standard analytic band model used by Jacoboni and Reggiani. Figure 33 shows the corresponding comparison with the experimental time-of-flight results.

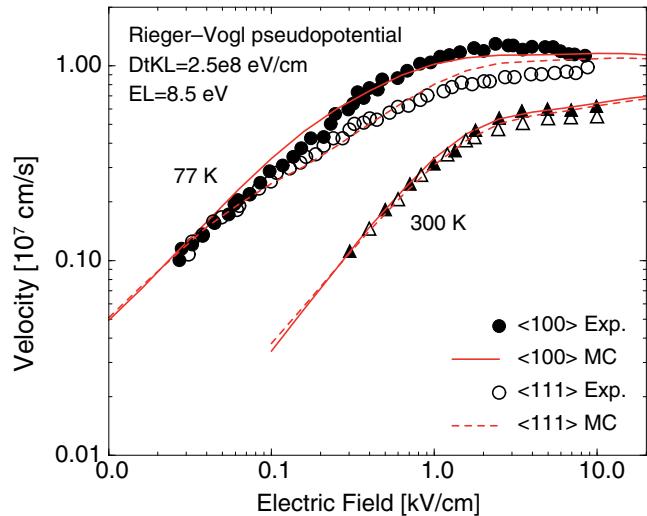
Electron transport in SiGe with arbitrary mole fraction, stress, and crystallographic orientation is possible. The band structures are Si-like up to $x_{\text{Ge}} = 85\%$ (only Δ -valleys are considered) and Ge-like above $x_{\text{Ge}} = 85\%$ (also the Γ -valley and L-valleys are considered).

The default value for the alloy potential of $U = 0.75 \text{ eV}$ was obtained from adjustment to drift mobility measurements as reported in [10].

Chapter 7: Electron and Hole Transport

Electron Transport in InGaAs

Figure 33 Simulated and measured velocity-field characteristics for electrons in germanium at 77 K and 300 K for transport in the crystallographic <100> and <111> directions



Electron Transport in InGaAs

This section describes the features of electron transport in indium gallium arsenide (InGaAs) that can be simulated using Sentaurus Device Monte Carlo.

Features of Electron Transport

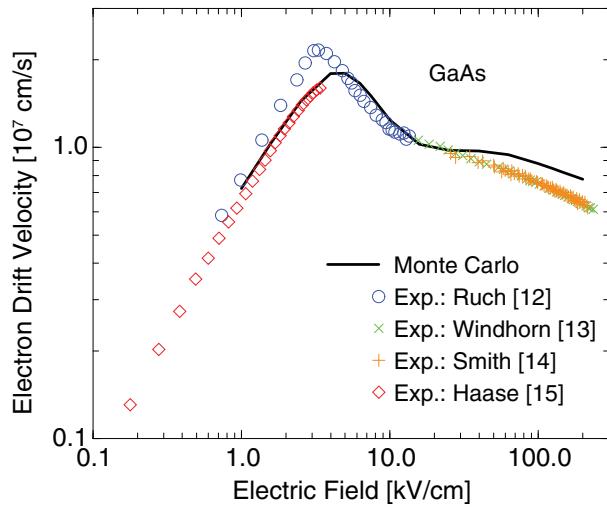
With regard to InGaAs, you can simulate electrons in relaxed InGaAs using the available pseudopotential tables for GaAs ($x_{\text{Ge}}=100$), InAs ($x_{\text{Ge}}=0$), and InGaAs ($x_{\text{Ge}}=47$).

The scattering mechanisms and the phonon-coupling constants for GaAs are the same as given by Lundstrom [11] (including, in particular, polar-optical phonon scattering) and reproduce the measured velocity-field characteristics as shown in [Figure 34](#).

Chapter 7: Electron and Hole Transport

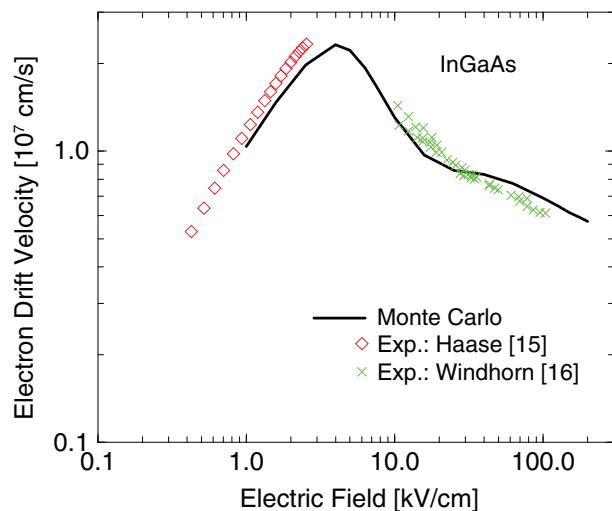
Electron Transport in InGaAs

Figure 34 Simulated and measured [12][13][14][15] velocity-field characteristics for electrons in gallium arsenide at 300 K for transport in crystallographic $<100>$ direction



For InAs, only the band structure and the dielectric constant change. In InGaAs, alloy scattering is added and adjusted to reproduce the measured velocity-field characteristics as shown in [Figure 35](#).

Figure 35 Simulated and measured [15][16] velocity-field characteristics for electrons in relaxed InGaAs (gallium content = 47%) at 300 K for transport in crystallographic $<100>$ direction



References

- [1] J. M. Hinckley and J. Singh, "Hole transport theory in pseudomorphic $\text{Si}_{1-x}\text{Ge}_x$ alloys grown on Si(001) substrates," *Physical Review B*, vol. 41, no. 5, pp. 2912–2926, 1990.
- [2] J. P. Dismukes, L. Ekstrom, and R. J. Paff, "Lattice Parameter and Density in Germanium–Silicon Alloys," *The Journal of Physical Chemistry*, vol. 68, no. 10, pp. 3021–3027, 1964.
- [3] F. M. Bufler, A. Schenk, and W. Fichtner, "Simplified model for inelastic acoustic phonon scattering of holes in Si and Ge," *Journal of Applied Physics*, vol. 90, no. 5, pp. 2626–2628, 2001.
- [4] G. Busch and O. Vogt, "Elektrische Leitfähigkeit und Halleffekt von Ge-Si-Legierungen," *Helvetica Physica Acta*, vol. 33, pp. 437–458, 1960.
- [5] J. W. Harrison and J. R. Hauser, "Alloy scattering in ternary III-V compounds," *Physical Review B*, vol. 13, no. 12, pp. 5347–5350, 1976.
- [6] P. Gaworzewski *et al.*, "Electrical properties of lightly doped *p*-type silicon-germanium single crystals," *Journal of Applied Physics*, vol. 83, no. 10, pp. 5258–5263, 1998.
- [7] F. M. Bufler and B. Meinerzhagen, "Hole transport in strained $\text{Si}_{1-x}\text{Ge}_x$ alloys on $\text{Si}_{1-y}\text{Ge}_y$ substrates," *Journal of Applied Physics*, vol. 84, no. 10, pp. 5597–5602, 1998.
- [8] F. M. Bufler, A. Tsibizov, and A. Erlebach, "Scaling of Bulk pMOSFETs: (110) Surface Orientation Versus Uniaxial Compressive Stress," *IEEE Electron Device Letters*, vol. 27, no. 12, pp. 992–994, 2006.
- [9] C. Jacoboni and L. Reggiani, "The Monte Carlo method for the solution of charge transport in semiconductors with applications to covalent materials," *Reviews of Modern Physics*, vol. 55, no. 3, pp. 645–705, 1983.
- [10] F. M. Bufler, *Full-Band Monte Carlo Simulation of Electrons and Holes in Strained Si and SiGe*, Munich: Herbert Utz, 1998.
- [11] M. Lundstrom, *Fundamentals of carrier transport*, Cambridge: Cambridge University Press, 2nd ed., 2000.
- [12] J. G. Ruch and G. S. Kino, "Transport Properties of GaAs," *Physical Review*, vol. 174, no. 3, pp. 921–931, 1968.
- [13] T. H. Windhorn *et al.*, "High field temperature dependent electron drift velocities in GaAs," *Applied Physics Letters*, vol. 40, no. 6, pp. 513–515, 1982.
- [14] P. M. Smith, M. Inoue, and J. Frey, "Electron velocity in Si and GaAs at very high electric fields," *Applied Physics Letters*, vol. 37, no. 9, pp. 797–798, 1980.

Chapter 7: Electron and Hole Transport

References

- [15] M. A. Haase *et al.*, "Subthreshold electron velocity-field characteristics of GaAs and $\text{In}_{0.53}\text{Ga}_{0.47}\text{As}$," *Journal of Applied Physics*, vol. 57, no. 6, pp. 2295–2298, 1985.
- [16] T. H. Windhorn, L. W. Cook, and G. E. Stillman, "The Electron Velocity–Field Characteristic for n– $\text{In}_{0.53}\text{Ga}_{0.47}\text{As}$ at 300 K," *IEEE Electron Device Letters*, vol. EDL-3, no. 1, pp.18–20, 1982.

8

Mobility Reduction in High-k Gate Stacks

This chapter describes how to account for the mobility reduction occurring in the presence of high- κ gate stacks in Sentaurus Device Monte Carlo.

Introduction

The scaling rules require that a further reduction of the gate length is accompanied by a reduction of the equivalent oxide thickness (EOT). However, further reduction of the physical (nitrided) silicon-dioxide (SiO_2) thickness below 1 nm is not possible, because the associated gate leakage current becomes too high.

An alternative is to replace SiO_2 with another gate-oxide material that has a higher dielectric constant so that a smaller EOT can be achieved with a physically thicker oxide, thereby suppressing the gate leakage. The current material of choice is hafnium dioxide (HfO_2), which features a relative dielectric constant of approximately 22 compared to the value of 3.9 in SiO_2 (new material `HfO2`).

However, a larger ionic polarization leading to the large static dielectric constant also involves a larger scattering potential arising from the coupling of the channel electrons with the soft-optical or surface-optical phonons arising at the semiconductor–insulator interface. This leads to a degradation of the surface mobility.

There are two models in Sentaurus Device Monte Carlo that take this mobility degradation into account. The first one (activated in the Sentaurus Device Monte Carlo command file with `HighKModus=2`) applies to the situation where the high- κ dielectric is separated from the semiconductor channel by a thin SiO_2 interfacial oxide (new material `InterfacialOxide`). Here, the scattering potential at a given electron position depends exponentially on the distance to the high- κ interface and explicitly on the thickness of the interfacial oxide.

The second model (activated in the Sentaurus Device Monte Carlo command file with `HighKModus=1`) applies to the situation where there is no interfacial oxide between the high- κ gate material and the semiconductor channel.

Soft-Optical Phonon Scattering

To illustrate the principal ingredients of soft-optical (SO) phonon scattering, the simplest case is described of a single interface between a semiconductor and an infinitely thick oxide with a single insulator transverse-optical (TO) phonon mode ω_{TO} . The solution of the secular equation for obtaining the SO phonon dispersion:

$$\epsilon_s(Q, \omega) + \epsilon_k(\omega) = 0 \quad (38)$$

with the following expressions for the dielectric function in the semiconductor:

$$\epsilon_s(Q, \omega) = \epsilon_s^\infty \quad (39)$$

and in the high- κ oxide:

$$\epsilon_k(\omega) = \epsilon_k^\infty + (\epsilon_k^0 - \epsilon_k^\infty) \frac{\omega_{\text{TO}}^2}{\omega_{\text{TO}}^2 - \omega^2} \quad (40)$$

yields:

$$\omega = \omega_{\text{SO}} = \omega_{\text{TO}} \sqrt{\frac{\epsilon_s^\infty + \epsilon_k^0}{\epsilon_s^\infty + \epsilon_k^\infty}} \quad (41)$$

with the material parameters being the high-frequency dielectric constant in the semiconductor ϵ_s^∞ as well as the high-frequency dielectric constant ϵ_k^∞ , the low-frequency dielectric constant ϵ_k^0 , and the TO phonon energy ω_{TO} in the high- κ oxide. The associated scattering potential reads [1]:

$$\Phi_Q(z) = \left\{ \frac{\hbar \omega_{\text{SO}}}{2Q} \left[\frac{1}{\epsilon_{\text{TOT,high}}^{\text{TO}}(Q)} - \frac{1}{\epsilon_{\text{TOT,low}}^{\text{TO}}(Q)} \right] \right\}^{1/2} e^{-Qz} \quad (42)$$

with:

$$\begin{aligned} \epsilon_{\text{TOT,high}}^{\text{TO}}(Q) &= \epsilon_s^\infty + \epsilon_k^\infty \\ \epsilon_{\text{TOT,low}}^{\text{TO}}(Q) &= \epsilon_s^\infty + \epsilon_k^0 \end{aligned} \quad (43)$$

in the simple case described above, where $z = 0$ is the position of the interface. Therefore, z is the distance to the semiconductor–high- κ interface, and \mathbf{Q} is the 2D momentum exchanged upon this scattering mechanism.

Chapter 8: Mobility Reduction in High- κ Gate Stacks

High- κ Mobility in the Presence of an Interfacial Oxide

In semiclassical approximation, the dependence on z is treated parametrically, and the resulting scattering rate is given by [2]:

$$S(\mathbf{k}) = \frac{e^2 \pi \omega_{SO}}{(2\pi)^2} \left(n_{\omega_{SO}} + \frac{1}{2} \mp \frac{1}{2} \right) \int d^2 Q \left[\frac{1}{\epsilon_{TOT,high}(Q)} - \frac{1}{\epsilon_{TOT,low}(Q)} \right] \times \frac{e^{-2Qz}}{Q} \delta(\epsilon(\mathbf{k} + \mathbf{Q}) - \epsilon(\mathbf{k}) \mp \hbar\omega_{SO}) \quad (44)$$

where the upper sign stands for phonon absorption and the lower sign stands for phonon emission. $n_{\omega_{SO}}$ is the Bose–Einstein distribution. Only the 2D \mathbf{Q} phonon vector is exchanged, while the k_z component of the 3D electron wavevector \mathbf{k} is conserved upon SO phonon scattering.

Note:

The actual denominator in [2] is $(2\pi)^3$, that is, the scattering rate in Equation 44 is larger by a factor of 2π . The denominator $(2\pi)^2$ emerges upon converting the sum over the 2D \mathbf{Q} phonon vector into an integral according to $(2\pi)^2 \Sigma_Q = A \int d^2 Q$, where A denotes the unit area.

High- κ Mobility in the Presence of an Interfacial Oxide

In the presence of an interfacial oxide (`HighKModus=2`), you also only consider the lowest-energy TO phonons of the two oxides, that is, SiO_2 and the high- κ oxide, because these modes are the most important ones for mobility degradation. These lead to three SO phonon modes.

Using the expressions for these three SO phonon modes and for the corresponding *total* effective dielectric functions reported in the Appendix of reference [1], the scattering rates are evaluated according to Equation 44 where one integration cancels due to the energy-conserving delta function, and the remaining integration is performed numerically using a parabolic approximation for the dispersion of the electron or hole energy.

The resulting scattering rates for a position directly under an interfacial oxide of 1-nm thickness are shown in Figure 36. In contrast to bulk phonon-scattering rates, the SO phonon-scattering rates do not depend on the total electron energy, but they do depend on the *parallel* energy. The parallel energy is obtained from the total energy by subtracting the energy component perpendicular to the gate oxide interface (for arbitrary geometry, the direction normal to the interface is replaced by the direction pointing from the electron position along the shortest distance to the high- κ interface). The *perpendicular* energy is approximated by multiplying the total energy by the ratio of the squares of the group-velocity component normal to the interface and of the total group velocity.

The material parameters, which can be specified in the Sentaurus Device Monte Carlo command file, are reported in Table 6 on page 115.

Chapter 8: Mobility Reduction in High- κ Gate Stacks

High- κ Mobility in the Presence of an Interfacial Oxide

For each real-space element of the semiconductor, the distance z to the high- κ oxide interface and the thickness t_{ox} of the interfacial oxide are extracted automatically from the geometry of the device.

Figure 36 Soft-optical phonon scattering rates directly under a 1-nm thick interfacial oxide for a HfO_2 dielectric compared to the total bulk phonon scattering rate. These SO phonon scattering rates are computed with the denominator of [2]; according to Equation 44, they would be larger by a factor of 2π .

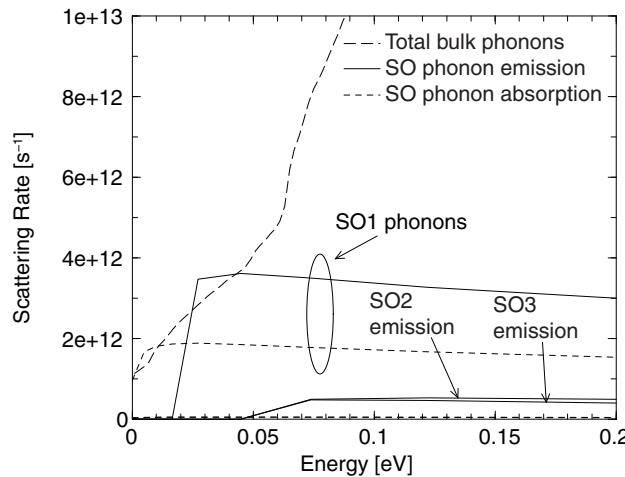


Table 6 shows the lowest-energy TO phonon energies as well as high-, intermediate-, and low-frequency dielectric constants of the high- κ oxide and the interfacial oxide with the default values taken from [1], which correspond to HfO_2 and SiO_2 , respectively.

Note:

In computing the distances to the high- κ oxide interface, every insulator is considered to be a high- κ material except `Oxynitride`, `Vacuum`, `Si3N4`, `SiO2`, and `InterfacialOxide`.

Table 6 Material parameters for SO phonon scattering in presence of an interfacial oxide

Parameter	Keyword	Value	Unit
$\hbar\omega_{\text{TO}}^{\kappa}$	HOMegaTO1	12.40	meV
$\hbar\omega_{\text{TO}}^{\text{ox}}$	OxideHOMegaTO1	55.60	meV
$\epsilon_{\kappa}^{\infty}$	HighKEpsilonInf	5.03	ϵ_0
ϵ_{κ}^i	HighKEpsilonInt	6.58	ϵ_0

Chapter 8: Mobility Reduction in High-k Gate Stacks

High-k Mobility in the Absence of an Interfacial Oxide

Table 6 Material parameters for SO phonon scattering in presence of an interfacial oxide

Parameter	Keyword	Value	Unit
ϵ_{k}^0	HighKEpsilonZero	22.0	ϵ_0
$\epsilon_{\text{ox}}^{\infty}$	OxideEpsilonInf	2.50	ϵ_0
ϵ_{ox}^i	OxideEpsilonInt	3.05	ϵ_0
ϵ_{ox}^0	OxideEpsilonZero	3.90	ϵ_0

Note:

The thickness of the interfacial oxide is computed for the material `InterfacialOxide`, that is, you have to deposit in the gate stack the material `InterfacialOxide` between the high-k insulator and the semiconductor channel. In addition, `InterfacialOxide` must be used exclusively for this interfacial layer and not, for example, as spacer material.

In view of the still-debated cause of high-k mobility degradation, the approximations involved in the described SO phonon scattering model, and experimental uncertainties, calibration to mobility measurements as reported, for example, in [3], is necessary. In addition to the model parameters in Table 6, it is therefore possible to specify the prefactor of the scattering rates for the three SO phonon modes in the Sentaurus Device Monte Carlo command file (`s01Factor`, `s02Factor`, and `s03Factor`) that are set to 1.0 by default. If the scattering rates coincide with those of [2], these prefactors must be equal to $1/(2\pi)$.

High-k Mobility in the Absence of an Interfacial Oxide

In the absence of an interfacial oxide, the previous model (`HighKModus=2`) can still be applied. In this case, however, an extended model, activated by `HighKModus=1`, can be used. Here, not only the lowest-energy TO phonon mode, but also the two lowest-energy transverse-optical phonon modes (TO1 and TO2) of the high-k oxide are considered.

Table 7 summarizes the material parameters associated with the two resulting SO phonon modes. It lists the two lowest-energy transverse-optical (TO1 and TO2) phonon energies as well as high-, intermediate-, and low-frequency dielectric constants of the high-k oxide with the default values taken from [1], which correspond to HfO2.

Again, the prefactor of the scattering rates of the two SO phonon modes (`s01Factor` and `s02Factor`) can be changed from the default value of 1.0.

Chapter 8: Mobility Reduction in High-k Gate Stacks

Remote Coulomb Scattering

Table 7 Material parameters for SO phonon scattering in absence of an interfacial oxide

Parameter	Keyword	Value	Unit
$\hbar\omega_{TO1}^k$	HOmegaTO1	12.40	meV
$\hbar\omega_{TO2}^k$	HOmegaTO2	48.35	meV
ϵ_k^∞	HighKEpsilonInf	5.03	ϵ_0
ϵ_k^i	HighKEpsilonInt	6.58	ϵ_0
ϵ_k^0	HighKEpsilonZero	22.0	ϵ_0

Note:

If this model is used in the presence of an interfacial oxide, the only effect of the interfacial oxide is to increase the distance z to the high- κ interface by the thickness t_{ox} of the interfacial oxide. This is not correct, however, because the interfacial oxide leads to a modification of the SO phonon spectrum and of the SO phonon scattering rates as incorporated in the model activated by `HighKModus=2`.

Remote Coulomb Scattering

Another possible source for mobility degradation is scattering from interface charges between the high- κ insulator and the interfacial oxide. In analogy to [Equation 44](#), the corresponding inverse microscopic relaxation time for unscreened remote Coulomb scattering (RCS) is:

$$\frac{1}{\tau_{RCS}(\mathbf{k})} = \frac{e^4 \pi n_S}{(2\pi)^2 2\hbar \epsilon_{eff}^2} \int d^2 Q \frac{e^{-2Qz}}{Q^2} (1 - \cos \varphi) \delta(\epsilon(\mathbf{k} + \mathbf{Q}) - \epsilon(\mathbf{k})) \quad (45)$$

where n_S denotes the density of interface charges per unit area, and φ is the angle between the 2D in-plane wavevectors of the carrier before and after scattering, and the effective dielectric constant is approximated by:

$$\epsilon_{eff} = (\epsilon_k^0 + \epsilon_s)/2 \quad (46)$$

in terms of the dielectric constants of the high- κ insulator and the semiconductor, respectively. For RCS, the value of the dielectric constant of the high- κ insulator can be

Chapter 8: Mobility Reduction in High-k Gate Stacks

References

adjusted. The wavevector after scattering is selected at random under conservation of energy and the wavevector component perpendicular to the interface. Note that the effect of RCS tends to be small for realistic densities of interface charges.

Table 8 summarizes the material parameters associated with RCS and lists the density of interface charges and the dielectric constant of the high- κ insulator. Again, the prefactor of the RCS rate (RCSHighKFactor) can be changed from the default value of 1.0.

Table 8 Material parameters for remote Coulomb scattering

Parameter	Keyword	Value	Unit
n_s	NintHighK	0.0	cm^{-2}
ϵ_{κ}^0	RCSHighKEpsilon	22.0	ϵ_0

References

- [1] M. V. Fischetti, D. A. Neumayer, and E. A. Cartier, "Effective electron mobility in Si inversion layers in metal–oxide–semiconductor systems with a high- κ insulator: The role of remote phonon scattering," *Journal of Applied Physics*, vol. 90, no. 9, pp. 4587–4608, 2001.
- [2] M. V. Fischetti *et al.*, "Theoretical Study of Some Physical Aspects of Electronic Transport in nMOSFETs at the 10-nm Gate-Length," *IEEE Transactions on Electron Devices*, vol. 54, no. 9, pp. 2116–2136, 2007.
- [3] M. Cassé *et al.*, "Carrier Transport in HfO₂/Metal Gate MOSFETs: Physical Insight Into Critical Parameters," *IEEE Transactions on Electron Devices*, vol. 53, no. 4, pp. 759–768, 2006.

9

Example: NMOS Transistor

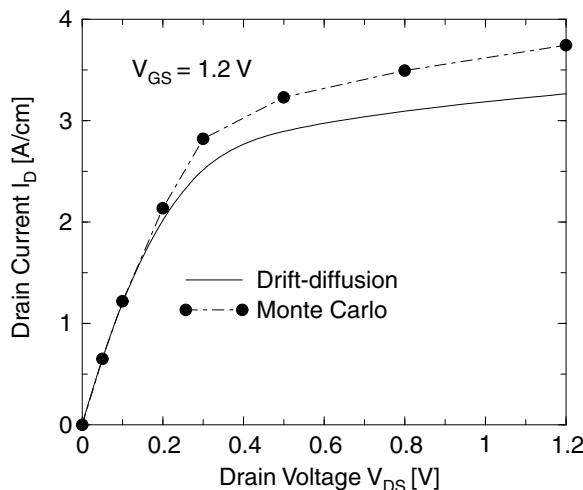
This chapter presents an example of an NMOS transistor as simulated by Sentaurus Device Monte Carlo.

Simulation of an NMOS Transistor

This example is a realistic 2D NMOS transistor with 100 nm gate length obtained from a process simulation. The geometry and the doping of the transistor are defined in a file named, for example, n5_msh.tdr.

After the drift-diffusion simulation invoked by `sdevice drift_new` is completed, the Monte Carlo simulation is performed by using `sdevice mc_new` as described in [Chapter 1 on page 19](#). Here, the emphasis is on the analysis of the simulation results. First, [Figure 37](#) shows the output characteristics of the NMOSFET. In comparison to Sentaurus Device Monte Carlo, the drift-diffusion simulation significantly underestimates the on-current.

Figure 37 Output characteristics of 0.1 μm NMOSFET



The simulation example in [Chapter 1 on page 19](#) refers only to the bias point with a drain voltage of 1.2 V, which is analyzed in detail here. In [Figure 38](#), the simulation results stored

Chapter 9: Example: NMOS Transistor

Simulation of an NMOS Transistor

in the file with the suffix `_time.plt` (to be viewed with Inspect) are displayed, that is, the currents as a function of the simulation time.

It can be seen that the currents begin to fluctuate around their average value after approximately ten iterations (with a simulation time Δt of $3.5 \mu\text{s}$ per iteration). Consequently, averaging over the current values shown above begins after ten iterations. The resulting, cumulative, current averages are shown in [Figure 39](#) as a function of iterations (after steady state is reached).

Figure 38 Drain and substrate currents calculated after consecutive time intervals Δt corresponding to single iterations of Sentaurs Device Monte Carlo as a function of simulation time

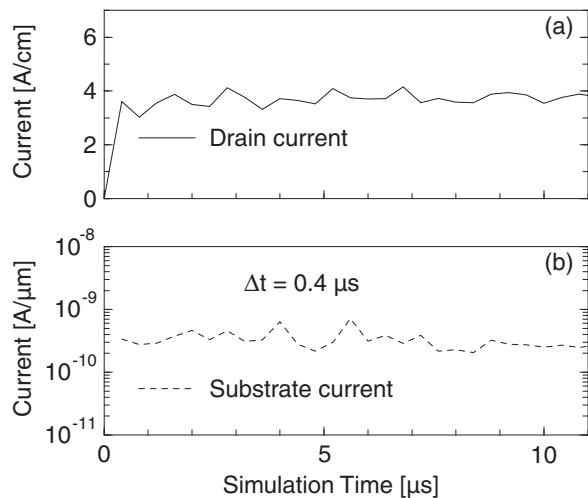
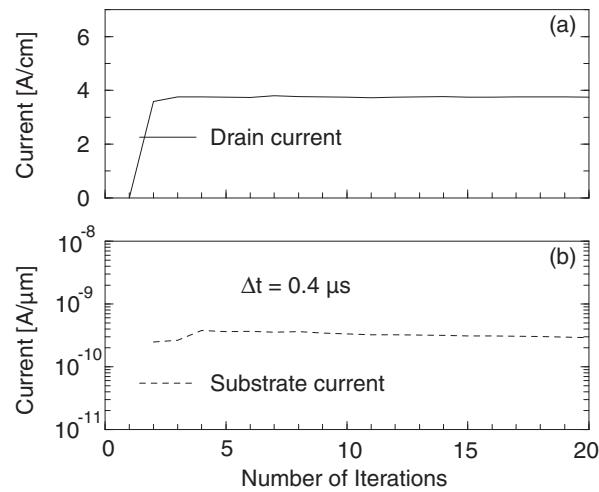


Figure 39 Cumulative averages for drain and substrate currents as a function of number of iterations



Chapter 9: Example: NMOS Transistor

Simulation of an NMOS Transistor

Figure 40 Electron density resulting from a drift-diffusion and Monte Carlo simulation, 0.5 nm below gate oxide along the channel

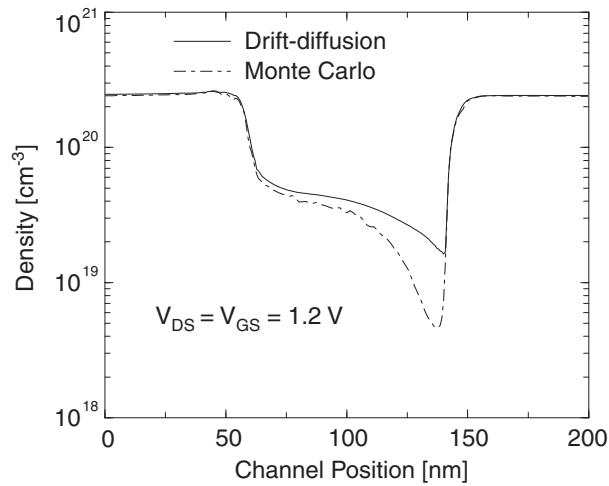
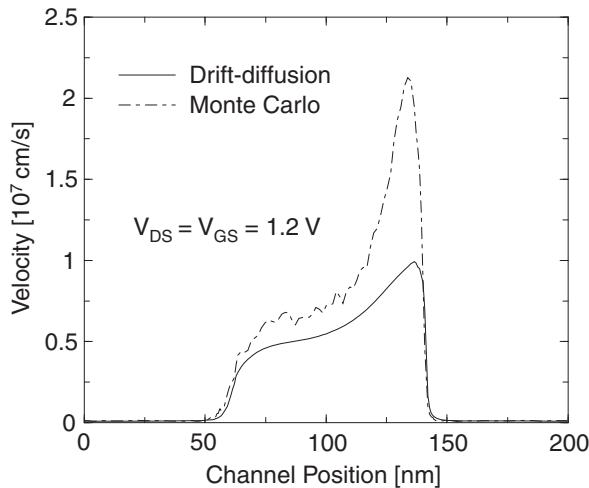


Figure 41 Electron drift velocity resulting from a drift-diffusion and Monte Carlo simulation, 0.5 nm below gate oxide along the channel



Finally, the internal variables in the `drift_new_des.tdr` and `mc_new_000024_des.tdr` data files can be viewed using Sentaurus Visual, and the values can be extracted along a line. In this example, a line along the channel 0.5 nm below the gate oxide is chosen. The corresponding profiles for the electron density and drift velocity are shown in [Figure 40](#) and [Figure 41](#), respectively. It can be seen that the velocity at the source side of the channel is greater for the Monte Carlo model than the drift-diffusion model, which is the reason for the higher on-current.

Part II: Band Structure and Mobility Calculation

This part of the *Sentaurus™ Device Monte Carlo User Guide* contains the following chapters:

- [Chapter 10, Using Sentaurus Band Structure](#)
- [Chapter 11, Empirical Pseudopotential Method](#)
- [Chapter 12, Analytic Bands for Bulk Crystals](#)
- [Chapter 13, Subband and Mobility Calculations](#)
- [Chapter 14, Sentaurus Band Structure/Tcl Commands](#)

10

Using Sentaurus Band Structure

This chapter describes how to use Sentaurus Band Structure.

Introduction to Sentaurus Band Structure

Sentaurus Band Structure is a tool for computing band-structure data for bulk crystalline solids, and the subband structure and inversion mobility of 1D and 2D devices.

Bulk Band Structure

The bulk band-structure capabilities of Sentaurus Band Structure include:

- The empirical pseudopotential method (EPM)
- A two-band $\mathbf{k} \cdot \mathbf{p}$ method for conduction bands
- A six-band $\mathbf{k} \cdot \mathbf{p}$ (Luttinger–Kohn or Bir–Pikus) solver for valence bands

Sentaurus Band Structure handles both relaxed and arbitrarily strained crystals. Band structures for disordered alloys (for example, SiGe) are computed by applying the virtual crystal approximation (VCA). Sentaurus Band Structure supports computation of band energies and their derivatives with respect to the \mathbf{k} -vector up to second order (that is, group velocities and reciprocal effective-mass tensors). Sentaurus Band Structure provides material parameters for Si, Ge, and $\text{Si}_{1-x}\text{Ge}_x$.

In addition to the usual band-structure computation at real \mathbf{k} -vectors, the EPM of Sentaurus Band Structure also supports the construction of real lines in complex \mathbf{k} -space. These consist of \mathbf{k} -points for which the analytic continuation of the band structure yields real eigen-energies. The complex band structure along real lines can be used to extract the E/\mathbf{k} dispersion relation needed to estimate tunneling probabilities in the framework of WKB tunneling theory.

Subband Structure and Inversion Mobility

The self-consistent subband dispersion in 1D and 2D device structures, for arbitrary surface orientation and strain, can be computed using several approaches:

- Parabolic Schrödinger equation with a perturbative nonparabolic correction
- Two-band $k \cdot p$ Schrödinger equation for electrons
- Six-band $k \cdot p$ Schrödinger equation for holes

From the subband dispersion in 1D devices, you can use a Kubo–Greenwood algorithm to compute the inversion-layer mobility using various scattering mechanisms.

TCAD Sentaurus Tutorial: Tcl Interface

To allow for maximum flexibility, Sentaurus Band Structure is driven by a Tcl interpreter into which band-structure and mobility computation capabilities have been added as application-specific Tcl commands. For an introduction to Tcl, see the Tool Command Language module in the TCAD Sentaurus Tutorial.

To access the TCAD Sentaurus Tutorial:

1. Open Sentaurus Workbench by entering the following on the command line: `swb`
2. From the menu bar of Sentaurus Workbench, choose **Help > Training** or click  on the toolbar.

Alternatively, to access the TCAD Sentaurus Tutorial:

1. Go to the directory `$STROOT/tcad/current/Sentaurus_Training`.
The `STROOT` environment variable indicates where the Synopsys TCAD distribution has been installed.
 2. Open the `index.html` file in your browser.
-

Starting Sentaurus Band Structure

The simplest way to start Sentaurus Band Structure is from the command line by entering:

`sband`

Chapter 10: Using Sentaurus Band Structure

Starting Sentaurus Band Structure

The startup message appears:

```
...
Initializing sBand.
sBand library path: (your-Sentaurus-Band-Structure-database-path)
sBand version (current-version)
```

This starts an interactive Sentaurus Band Structure session. The command prompt belongs to a Tcl interpreter, which accepts all standard Tcl commands as well as the application-specific Sentaurus Band Structure/Tcl commands described in [Chapter 14 on page 268](#).

Command-Line Options

The Sentaurus Band Structure executable `sband` supports both interactive and batch sessions. The complete startup syntax of the executable is:

```
sband [-|filename TclArg...] [--max_threads m] [-noLicense <string>]
[-nThreads n] [-showProgressBars] [--threads n] [-v]
```

If no arguments other than command-line options are supplied on the command line, then Sentaurus Band Structure operates in interactive mode. Otherwise, its mode is determined by the first argument that is not an option.

If this argument is a solitary dash (-), Sentaurus Band Structure enters interactive mode. Otherwise, the argument is interpreted as the `filename` of a Sentaurus Band Structure/Tcl script that is executed in batch mode. Any arguments following `filename` (or -) pass to the Tcl interpreter.

Table 9 Command-line options

Option	Description
<code>- filename TclArg...</code>	Specify a solitary dash (-) so that Sentaurus Band Structure enters interactive mode. Sets the file name of a Sentaurus Band Structure/Tcl script that is executed in batch mode, which might be followed by arguments. See Notational Conventions of Syntax Description on page 270 .
<code>--max_threads m</code>	Sets the maximum number of worker threads allowed during a simulation, that is, the effective number of worker threads is $\min(n,m)$. See the <code>--threads</code> option.

Chapter 10: Using Sentaurus Band Structure

Starting Sentaurus Band Structure

Table 9 *Command-line options (Continued)*

Option	Description
-noLicense <string>	Controls the behavior of Sentaurus Band Structure if, upon entering a parallel section of the tool, the number of available parallel licenses is insufficient for the requested number of threads. Supported values are: <ul style="list-style-type: none">• exit: Exits if insufficient parallel licenses are available.• reduce: (Default) Reduces the number of threads to the maximum allowed by the available number of parallel licenses.• throw: Throws an exception (which can be caught by the Tcl <code>catch</code> command).• wait: Waits until the required number of licenses becomes available.
-nThreads <i>n</i>	For backward compatibility, this option is still available to set the number of worker threads if nothing else is specified. Default: 1
-showProgressBars	Specify this option so that the confined k·p calculator displays progress bars.
--threads <i>n</i>	Sets the number of initial worker threads used for parallel computation. See Parallelization on page 132 .
-v	Specify this option so that Sentaurus Band Structure operates in verbose mode and outputs additional status messages.

Example

Start an interactive Sentaurus Band Structure session with three command-line arguments:

```
sband - arg1 arg2 arg3
```

Inside the Sentaurus Band Structure session, command-line arguments are stored in a Tcl list \$argv:

```
puts $argv
```

Result: arg1 arg2 arg3

The number of arguments is stored in \$argc:

```
puts $argc
```

Result: 3

\$argv0 is the name of the Tcl script being executed. In an interactive session, it is replaced by the executable name sband:

```
puts $argv0
```

Result: sband

EPM Band-Structure Tutorial

This section introduces the EPM band-structure calculation and describes how to compute the band energies of unstrained silicon at selected \mathbf{k} -vectors.

Creating a Silicon Crystal

First, you need to create an `EPM::Crystal` for unstrained silicon:

```
SiliconCrystal name=bulkSi
```

Result:

```
"Si" parameters from "(your-database-path)/Si_param.tcl".
```

The `SiliconCrystal` command creates an `EPM::Crystal` object representing an EPM calculator for unstrained bulk silicon using the default parameters (see [SiliconCrystal on page 281](#)). The name of the new crystal is `bulkSi`. You can use this name as a Tcl command to call methods of the `EPM::Crystal` class (see [Using EPM::Crystal Objects on page 283](#)).

For example, you can display status information about `bulkSi` by using the `<EPM::Crystal> status` method (see [<EPM::Crystal> status on page 290](#)).

Computing Band-Structure Data

Next, you need to specify the \mathbf{k} -vectors at which band-structure data is needed. For example, you can select the zone center, $\Gamma = (0, 0, 0)$, and the conduction-band minimum in the x -direction, $\Delta = \left(0.85 \cdot \frac{2\pi}{a_0}, 0, 0\right)$:

```
set Gamma { 0 0 0 }
set Delta { 0.85 0 0 }
```

In Sentaurus Band Structure, a three-dimensional vector (`vector3D`) is represented as a list of three numbers (see [Table 55 on page 268](#) for type-naming conventions in Sentaurus Band Structure).

In unstrained crystals, \mathbf{k} -vectors are specified in units of $2\pi/a_0$, where a_0 is the lattice constant. In strained crystals, an anisotropic metric based on the diagonal entries of the strain tensor is used; the \mathbf{k} -unit along direction $i \in x, y, z$ becomes:

$$\frac{2\pi}{a_i} = \frac{2\pi}{(1 + \varepsilon_{ii})a_0} \quad (47)$$

The band-structure calculation at the two specified **k**-vectors is performed by calling the `computeBandstructure` method of `bulkSi`:

```
bulkSi computeBandstructure kVectors=[list $Gamma $Delta]
```

Output:

```
writing "energy.dat"  
writing "velocity.dat"  
Band-structure calculation complete!
```

The call to `bulkSi` illustrates how Tcl commands in Sentaurus Band Structure use named arguments.

The line:

```
bulkSi computeBandstructure kVectors=[list $Gamma $Delta]
```

means: Use the Tcl `list` command to construct a list containing the two **k**-vectors `$Gamma` and `$Delta`, and pass this list to the `computeBandstructure` method of the `EPM::Crystal` object `bulkSi` using an argument name of `kVectors`.

For details, see [<EPM::Crystal> computeBandstructure on page 285](#). The notation `<EPM::Crystal>` denotes the name of an object of the class `EPM::Crystal`.

By default, the `<EPM::Crystal> computeBandstructure` method computes both band energies and their first derivatives with respect to the components of **k** (that is, the group velocities) and stores the results in both internal data objects and external files (default names are `energy.dat`, `velocity.dat`; the file format is described in [Generating Band-Structure Tables From k-Vector Files on page 134](#)).

These settings can be inspected and changed using the `sBandGet` and `sBandSet` commands (see [sBandGet and sBandSet on page 273](#)).

Inspecting the Results

Band-energy results of the call to `bulkSi computeBandstructure` are stored in a `bandstructure_t` container object named `bulkSi.bandstructure`. To store the band energies at the Γ -point in a Tcl variable `EGamma`, use:

```
set EGamma [bulkSi.bandstructure get kVector=$Gamma]
```

Result:

```
-12.4325779396 -0.0434084085113 0.0 0.0 3.34924141875 3.37860208567  
3.37860208567 4.28964364273
```

The result is a list of band energies at the specified **k**-value in eV, ordered from the lowest energy to the highest energy. By default, the band structure is shifted such that the highest

valence-band energy at the Γ -point is zero. This is controlled by the global argument `shiftValenceBands` (see [sBandGet and sBandSet on page 273](#)).

Individual band energies can be accessed by calling `lindex` on the result of the previous command. The following example returns the silicon spin-orbit splitting energy:

```
set LSsplitting [expr [lindex $EGamma 3] - [lindex $EGamma 1]]
```

Alternatively, you can use the optional `band` argument of the `<bandstructure_t> get` method (see [Using <bandstructure_t> Objects on page 320](#)). The following example returns the energy of the split-off band at Γ . Indexing in Tcl is zero based; the lowest band is `band=0`:

```
bulkSi.bandstructure get kVector=$Gamma band=1
```

Group velocities can be accessed in a similar way. They are stored in a `groupVelocity_t` container object named `bulkSi.groupVelocity`. The unit for velocities is $\frac{eV}{h}a_0$ for unstrained crystals with the anisotropic generalization:

$$\frac{eV}{h}a_i = \frac{eV}{h}(1 + \varepsilon_{ii})a_0 \quad (48)$$

along direction $i \in x, y, z$ for strained crystals; $h = 2\pi\hbar$ is Planck's constant.

In contrast to `bandstructure_t`, each entry of `groupVelocity_t` is a 3D vector $\nabla_{\mathbf{k}}\varepsilon_n(\mathbf{k})$ instead of a scalar $\varepsilon_n(\mathbf{k})$:

```
bulkSi.groupVelocity get kVector=$Gamma
```

Result:

```
{-4.86118497298e-16 5.96545799415e-16 -1.20600149345e-15}
{7.01328040343e-15 2.95588548854e-15 3.12817509024e-15}
{-1.00799393365e-14 -6.64426133894e-15 6.86148234339e-15}
{-8.3453929703e-15 -9.35143006744e-16 1.55941477847e-14}
{-1.1541324112e-15 -1.68419123141e-15 2.09038668377e-14}
{-2.6478279379e-14 2.62866341284e-14 -5.50735119429e-16}
{-4.00283363058e-15 7.59429672475e-15 -1.37298691945e-14}
{-3.36626334074e-15 -4.74914628742e-15 -2.34160613001e-14}
```

As expected for the Γ -point, all the velocities are essentially zero. Sentaurus Band Structure uses numeric derivatives; the deviation of the results from zero is indicative of the level of numeric noise.

More interesting are the results at $\Delta = \left(0.85 \cdot \frac{2\pi}{a_0}, 0, 0\right)$. The group velocity in the highest valence band is returned by the command:

```
bulkSi.groupVelocity get kVector=$Delta band=3
```

Result:

```
-1.29159365151 5.9538212891e-14 -4.65946642766e-14
```

The velocity vector is parallel to the x-axis; its magnitude of ≈ 1.29 is a typical value away from symmetry points.

The group velocity in the lowest conduction band is accessed by:

```
bulkSi.groupVelocity get kVector=$Delta band=4
```

Result:

```
0.00134385756792 1.07791226662e-13 3.70164103554e-13
```

There is still a small (but numerically significant) velocity component along the x-direction, which indicates that the wave-number vector Δ is close to, but not exactly at, the conduction band minimum.

Finding the Band Minimum

You can find the position of a local minimum of an energy band (default: the lowest conduction band, `band=4`) near a starting vector (see [findBandMinimum on page 316](#)).

The conduction band minimum near $\Delta = \left(0.85 \cdot \frac{2\pi}{a_0}, 0, 0\right)$ is found using:

```
set Delta_min [findBandMinimum crystal=bulkSi kStart=$Delta]
```

Output:

```
K = 0.849880 0.000000 -0.000000; |dK|=1.203e-04; |v| = 8.065e-05
K = 0.849880 0.000000 -0.000000; |v| = 4.410e-08; done.
0.849879655201 5.8368242931e-16 -5.07548889268e-14
```

The lines starting with `K =` show the progress of the minimization process where:

- $|d\mathbf{K}|$ is the magnitude of the last change in \mathbf{K} .
- $|v|$ is the magnitude of the group velocity at the current \mathbf{K} -point.

The extracted position of the conduction-band minimum is:

$$\text{Delta_min} = \begin{bmatrix} 0.849879655201 \\ 5.8368242931 \times 10^{-16} \\ -5.07548889268 \times 10^{-14} \end{bmatrix} \quad (49)$$

Computing Effective Masses

Now that you have found the position of the conduction-band minimum, you can compute the effective electron masses at the minimum.

Chapter 10: Using Sentaurus Band Structure

EPM Band-Structure Tutorial

To do this, you must switch on the computation of the second derivatives of the band energies – the reciprocal effective-mass tensor \underline{m}_n^{-1} of band n at \mathbf{k} is the Hessian matrix:

$$\frac{\partial^2 \epsilon_n(\mathbf{k})}{\partial k_i \partial k_j} \quad (50)$$

of the band energy divided by \hbar^2 .

To switch on the second derivatives, use the command:

```
sBandSet derivOrder=2
```

Now, calls to `<EPM::Crystal> computeBandstructure` will compute energies, group velocities, and reciprocal effective masses:

```
bulkSi computeBandstructure kVectors=[list $Delta $Delta_min]
```

Output:

```
writing "energy.dat"
writing "velocity.dat"
writing "eInvMass.dat"
Band-structure calculation complete!
```

The inverse mass tensors for each band are stored in an `inverseMass_t` container object of the name `bulkSi.inverseMass`. The reciprocal effective-mass tensor for the lowest conduction band at Δ_{\min} is obtained by the Tcl command:

```
set invM [bulkSi.inverseMass get kVector=$Delta_min band=4]
```

Result:

```
{1.09441250877 1.87446738737e-08 -1.08349502291e-08}
{1.87446738737e-08 5.10579737043 4.86819561794e-10}
{-1.08349502291e-08 4.86819561794e-10 5.10579746899}
```

The 3×3 reciprocal effective-mass tensor is represented as a list of three real row-vectors (type signature: `List#3/Double`, shorthand: `RealMatrix3D`). To obtain the effective masses proper, the matrix of inverse masses must be inverted:

```
set M [invertMatrix $invM]
```

Result:

```
{0.913732246283 -3.3545422434e-09 1.93902000089e-09}
{-3.3545422434e-09 0.195855794394 -1.86741579405e-11}
{1.93902000089e-09 -1.86741579405e-11 0.195855790613}
```

and you find the following values for the longitudinal mass:

```
set m_1 [lindex [lindex $M 0] 0]
```

Result: 0.913732246283

Chapter 10: Using Sentaurus Band Structure

Parallelization

and the transverse mass:

```
set m_t [lindex [lindex $M 1] 1]
```

Result: 0.195855794394

These values are in units of the free electron mass m_0 .

Comparing these results to those obtained at Δ instead of Δ_{\min} shows that $(0.85 \cdot \frac{2\pi}{a_0}, 0, 0)$ is a sufficiently good approximation to the position of the valley minimum in unstrained silicon.

Parallelization

To take full advantage of modern multicore architectures, Sentaurus Band Structure supports multithreading for the computationally expensive pseudopotential calculations. The default number of worker threads is 1.

This value can be overridden by setting the `--threads` option when starting Sentaurus Band Structure (see [Command-Line Options on page 125](#)).

At runtime, you can adjust the required number of worker threads by using the Tcl command:

```
sBandSet nThreads=Integer
```

Sentaurus Band Structure detects automatically the number of hardware threads supported by the architecture on which it runs. This is performed by specifying a thread number of -1 either on the command line or using `sBandSet` (see [sBandGet and sBandSet on page 273](#)).

Applying Strain

For crystals of cubic symmetry, Sentaurus Band Structure can compute the displacement of the atoms in the crystal lattice in response to mechanical strain (including the effect of *internal strain*; see [Bulk Strain and Internal Strain on page 157](#)). This information can be used to determine the effect of strain on the band structure.

If you already know the strain tensor ε , then you can apply it to an `EPM::Crystal` object using the `<EPM::Crystal> set` method (see [<EPM::Crystal> set on page 289](#)). For example:

```
SiliconCrystal name=strainedSi
set strain {{0.01 0 0} {0 0.01 0} {0 0 -0.00776}}
strainedSi set strain=$strain
```

Special commands support the calculation of the strain tensor for special situations.

Biaxial Strain

Biaxial strain resulting from growing a pseudomorphic material layer on top of a thick substrate can be computed by calling the `<EPM::Crystal> apply biaxialStrain` method on the pseudomorphic layer (see [<EPM::Crystal> apply on page 284](#)).

You need to provide the following information:

- The direction `dir` of the interface between the two materials.
- Information on the in-plane strain at the interface – you specify either `inPlaneStrain` numerically or the `substrate` material, and Sentaurus Band Structure computes `inPlaneStrain` from the lattice mismatch.

For example:

```
SiemensCrystal name=silicon_layer
SiGeCrystal name=SiGe30 xGe=0.3
silicon_layer apply biaxialStrain dir=[list 0 0 1] substrate=SiGe30
puts [silicon_layer get strain]
```

Output:

```
{0.0113938 0.0 0.0} {0.0 0.0113938 0.0} {0.0 0.0 -0.00884294925374}
```

Growing a pseudomorphic silicon layer on top of a (100) $\text{Si}_{0.7}\text{Ge}_{0.3}$ substrate results in $\approx 1.14\%$ of tensile in-plane strain and an out-of-plane contraction of $\approx 0.88\%$.

Uniaxial Strain

In situations where there is only a single direction of stress, the strain tensor can be computed from the direction of the stress `dir` (specified as a real 3D vector) and its magnitude (in Pa) where positive is tensile and negative is compressive.

For example:

```
GermaniumCrystal name=germanium
germanium apply uniaxialStrain dir=[list 0 1 1] stress=1.0e9
puts [germanium get strain]
```

Output:

```
{-0.00261270185635 0.0 0.0}
{0.0 0.0034774321266 0.00374251497005}
{0.0 0.00374251497005 0.0034774321266}
```

Chapter 10: Using Sentaurus Band Structure

Generating Band-Structure Tables From k-Vector Files

Strain Tensor From Stress Tensor in Principal-Axis System

If you know the stress tensor in the principal-axis coordinate system of a cubic crystal, then Sentaurus Band Structure can convert it to strain by using the anisotropic Hooke's law (Einstein convention implied):

$$\varepsilon_{ij} = S_{ijkl}\sigma_{kl} \quad (51)$$

where ε denotes the strain tensor, σ is the stress tensor, and S is the compliance tensor.

Following the conventions of Voigt notation, pairs of indices (range: x, y, z) are combined into a single index (range: 1...6) according to the mapping:

$$xx \rightarrow 1 \quad yy \rightarrow 2 \quad zz \rightarrow 3 \quad yz \rightarrow 4 \quad xz \rightarrow 5 \quad xy \rightarrow 6 \quad (52)$$

This allows certain operations involving second-rank and fourth-rank tensors to be mapped to vector matrix operations. For example, the tensor contraction on the right-hand side of [Equation 51](#) is mapped onto a six-dimensional matrix vector product (note that the 4,5,6 components of the *strain vector* must be divided by 2 during mapping to the components of the strain tensor – $\varepsilon_{yz} = \varepsilon_4/2$ and so on):

$$\varepsilon_\alpha = C_{\alpha\beta}\sigma_\beta \quad (53)$$

This form of Hooke's law is implemented in the Sentaurus Band Structure/Tcl command `computeStrainFromStress`. Unfortunately, the reduction in effective tensor rank is at the expense of the form invariance of the anisotropic Hooke's law. In Voigt form, it is only valid in the principal-axis coordinate system of the crystal. Therefore, stress tensors (for example, from process simulation) must be transformed to this coordinate system before supplying them to `computeStrainFromStress`.

For example:

```
SiCrystal name=silicon
silicon apply strainFromStress stress=[list {1e9 0 0} {0 0 0} {0 0 0}]
puts [silicon get strain]
```

Output:

```
{0.00762451321992 0.0 0.0}
{0.0 -0.00213158434106 0.0}
{0.0 0.0 -0.00213158434106}
```

Generating Band-Structure Tables From k-Vector Files

Full-band Monte Carlo simulation requires band energies and group velocities on a large number of **k**-vectors distributed throughout the Brillouin zone. For this situation, the `<EPM::Crystal>.computeBandstructure` method allows you to supply the name of a file containing the **k**-vectors rather than a list of **k**-vectors. Each line of that file must contain the three coordinates of a **k**-vector separated by space.

Chapter 10: Using Sentaurus Band Structure

Taking Advantage of Band-Structure Symmetries

For example:

```
sBandSet shiftConductionBands=1 shiftValenceBands=1
          ;# enable band shifting
sBandSet automaticBandstructureFiles=1 ;# output "energy.dat" and so on
set f [open "tmp.file" "w"]
puts $f "0 0 0"
puts $f "0.85 0 0"
puts $f "0 0.85 0"
puts $f "0 0 0.85"
close $f
SiliconCrystal name=bulk
bulk computeBandstructure kFile="tmp.file" ;# compute bands using ...
          ;# ... k-vectors from file
```

Output:

```
writing "energy.dat"
writing "velocity.dat"
Band-structure calculation complete!
```

Taking Advantage of Band-Structure Symmetries

For certain applications such as full-band Monte Carlo, it might be necessary or convenient to store band-structure data on a region that is larger than the irreducible wedge of the Brillouin zone. For crystals in the cubic system, Sentaurus Band Structure can infer automatically the symmetry group that corresponds to a particular strain configuration and can use this information to reduce the computational effort.

For each set of symmetry-related **k**-vectors, the band-structure calculation is performed only once. Results at the remaining **k**-vectors are constructed by applying symmetry operations to the results at the first **k**-vector.

For example:

```
SiliconCrystal name=bulk
bulk apply uniaxialStrain stress=1e9 dir=[list 1 1 1]
set sym [determineSymmetries strain=[bulk get strain] \
          crystalClass=cubic]
```

Output: {-1 -2 -3} {2 1 3} {3 2 1} {1 3 2}

The symmetry group for silicon under 1 GPa of tensile stress along the [111] direction is generated by Kramer's symmetry ($\{-1 -2 -3\}$, that is, $k_x \rightarrow -k_x$, $k_y \rightarrow -k_y$, $k_z \rightarrow -k_z$) and the permutations of the **k**-vector components (for example, $\{2 1 3\}$, that is, $k_x \rightarrow k_y$, $k_y \rightarrow k_x$, $k_z \rightarrow k_z$).

To utilize these symmetries during the band-structure calculation, the symmetry list `$sym` must be passed to the `<EPM::Crystal> computeBandstructure` call:

```
bulk computeBandstructure kFile="large.file" symmetries=$sym
```

File Formats for Storing Band Data

Sentaurus Band Structure supports different formats for storing band data:

- `$SHORT_FORMAT`: Includes only the three highest conduction bands and the four lowest valence bands. This is the default format used by Sentaurus Band Structure (see [File Format \\$SHORT_FORMAT on page 136](#)).
- `$LONG_FORMAT`: All bands are included (see [File Format \\$LONG_FORMAT on page 139](#)).
- `$TDR3DTENSOR_FORMAT`: Band data is stored as a 3D TDR tensor-product file. This format can only be selected if the **k**-vectors form a 3D tensor-product grid.
- `$MONTECARLO_FORMAT`: Creates band-structure data files in the format expected by the single-particle device Monte Carlo simulator (see [Creating Band Data for Sentaurus Device Monte Carlo on page 139](#)).

To select one of the formats, specify:

```
sBandSet fileFormat=String
```

You select multiple formats by using the binary OR operation. For example, to select both the short format and the long format, specify the command:

```
sBandSet fileFormat=[expr $SHORT_FORMAT|$LONG_FORMAT]
```

File Format \$SHORT_FORMAT

Table 10 Format of band-energy files (energy.dat)

n_k									
k_x	k_y	k_z	$-\epsilon_{hh}$	$-\epsilon_{lh}$	$-\epsilon_{so}$	ϵ_{c1}	ϵ_{c2}	ϵ_{c3}	ϵ_{c4}
:	:	:	:	:	:	:	:	:	:

The first line of this file contains the number of **k**-vectors n_k . The remaining lines list each **k**-vector with the band energies of the three highest valence bands and the four lowest conduction bands in eV. Valence-band energies are shown with a prefactor of -1 .

Chapter 10: Using Sentaurus Band Structure

Taking Advantage of Band-Structure Symmetries

Table 11 Format of group-velocity files (velocity.dat)

n_k									
k_x	k_y	k_z	$-\nabla_{\mathbf{k}}\epsilon_{hh}$	$-\nabla_{\mathbf{k}}\epsilon_{lh}$	$-\nabla_{\mathbf{k}}\epsilon_{so}$	$\nabla_{\mathbf{k}}\epsilon_{c1}$	$\nabla_{\mathbf{k}}\epsilon_{c2}$	$\nabla_{\mathbf{k}}\epsilon_{c3}$	$\nabla_{\mathbf{k}}\epsilon_{c4}$
:	:	:	:	:	:	:	:	:	:

The first line of this file contains the number of \mathbf{k} -vectors n_k . The remaining lines list each \mathbf{k} -vector with the group velocities of the three highest valence bands and the four lowest conduction bands. Velocity units are $\frac{\text{eV}}{\hbar} a_{x/y/z}$, where $a_{x/y/z}$ denotes an anisotropic metric that uses the effective lattice constant $a_i := (1 + \epsilon_{ii})a_0$ as the length unit along the coordinate direction $i \in \{x, y, z\}$. Valence-band velocities are shown with a prefactor of -1 .

Each velocity vector is output as its x-, y-, and z-components separated by space.

Reciprocal Effective Mass Files

If you specify `sBandSet derivOrder=2`, then Sentaurus Band Structure computes electron and hole reciprocal effective masses and outputs them to the files `eInvMass.dat` and `hInvMass.dat`, respectively.

The first line of these files contains the number of \mathbf{k} -vectors. The coordinates of the \mathbf{k} -vectors are not included in these files. Their order is the same as in the corresponding energy and group velocity files.

For each \mathbf{k} -vector, there is a block of four lines (for the highest four valence bands or the lowest four conduction bands) followed by an empty line. Each line lists the six independent components of the reciprocal effective-mass tensor of one band in units of the inverse of the electron mass m_0 .

Table 12 Format of hole reciprocal effective mass files (hInvMass.dat)

n_k					
$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{hh}}{\partial k_x \partial k_x}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{hh}}{\partial k_x \partial k_y}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{hh}}{\partial k_y \partial k_y}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{hh}}{\partial k_x \partial k_z}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{hh}}{\partial k_y \partial k_z}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{hh}}{\partial k_z \partial k_z}$
$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{lh}}{\partial k_x \partial k_x}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{lh}}{\partial k_x \partial k_y}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{lh}}{\partial k_y \partial k_y}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{lh}}{\partial k_x \partial k_z}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{lh}}{\partial k_y \partial k_z}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{lh}}{\partial k_z \partial k_z}$
$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{so}}{\partial k_x \partial k_x}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{so}}{\partial k_x \partial k_y}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{so}}{\partial k_y \partial k_y}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{so}}{\partial k_x \partial k_z}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{so}}{\partial k_y \partial k_z}$	$-\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{so}}{\partial k_z \partial k_z}$

Chapter 10: Using Sentaurus Band Structure
Taking Advantage of Band-Structure Symmetries

Table 12 Format of hole reciprocal effective mass files (hInvMass.dat) (Continued)

n_k	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{lo}}{\partial k_x \partial k_x}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{lo}}{\partial k_x \partial k_y}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{lo}}{\partial k_y \partial k_y}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{lo}}{\partial k_x \partial k_z}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{lo}}{\partial k_y \partial k_z}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{lo}}{\partial k_z \partial k_z}$
	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

Table 13 Format of electron reciprocal effective mass files (eInvMass.dat)

n_k	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{c1}}{\partial k_x \partial k_x}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{c1}}{\partial k_y \partial k_y}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{c1}}{\partial k_z \partial k_z}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{c1}}{\partial k_y \partial k_z}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{c1}}{\partial k_x \partial k_z}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{c1}}{\partial k_x \partial k_y}$
	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{c2}}{\partial k_x \partial k_x}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{c2}}{\partial k_y \partial k_y}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{c2}}{\partial k_z \partial k_z}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{c2}}{\partial k_y \partial k_z}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{c2}}{\partial k_x \partial k_z}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_{c2}}{\partial k_x \partial k_y}$
	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

Chapter 10: Using Sentaurus Band Structure

Creating Band Data for Sentaurus Device Monte Carlo

File Format \$LONG_FORMAT

In long format, data for all bands is written into files. The same sign conventions are used for electron and hole states.

Table 14 Format of band energy files (energy_dump.dat)

n_k						
k_x	k_y	k_z	ϵ_0	ϵ_1	...	ϵ_n
:	:	:	:	:	:	:

Table 15 Format of group velocity files (velocity_dump.dat)

n_k						
k_x	k_y	k_z	$\nabla_{\mathbf{k}}\epsilon_0$	$\nabla_{\mathbf{k}}\epsilon_1$...	$\nabla_{\mathbf{k}}\epsilon_n$
:	:	:	:	:	:	:

Each velocity vector is output as its x-, y-, and z-components separated by space.

Table 16 Format of reciprocal effective mass files (invMass_dump.dat)

n_k			Inverse mass tensor for band 0						... band 1		
k_x	k_y	k_z	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_0}{\partial k_x \partial k_x}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_0}{\partial k_y \partial k_y}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_0}{\partial k_z \partial k_z}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_0}{\partial k_y \partial k_z}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_0}{\partial k_x \partial k_z}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_0}{\partial k_x \partial k_y}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_1}{\partial k_x \partial k_x}$	$\frac{1}{\hbar^2} \frac{\partial^2 \epsilon_1}{\partial k_y \partial k_y}$...
:	:	:	:	:	:	:	:	:	:	:	:

Each line lists a \mathbf{k} -vector followed by the inverse mass components of the lowest band (in the order xx, yy, zz, yz, xz, xy) followed by the inverse mass components of the next band, and so on.

Creating Band Data for Sentaurus Device Monte Carlo

You can create band-structure tables for single-particle device Monte Carlo by using the `createMonteCarloFiles` command (see [createMonteCarloFiles on page 315](#)). The

Chapter 10: Using Sentaurus Band Structure

Creating Band Data for Garand MC

following example generates Monte Carlo input files for silicon under 3 GPa compressive stress in the SPARTA subdirectory of the current working directory:

```
SiliconCrystal name=bulk  
bulk apply uniaxialStrain stress=-3e9 dir=[list 1 1 0]  
createMonteCarloFiles crystal=bulk
```

[Arbitrary Stress on page 86](#) describes how to instruct the Monte Carlo simulator to import custom band-structure data.

Creating Band Data for Garand MC

You can create band-structure tables for Garand MC by using the `createGarandMCFfiles` command (see [createGarandMCFfiles on page 314](#)). The following example generates band-structure input files for silicon under 3 GPa compressive stress in the current working directory:

```
SiliconCrystal name=bulk  
bulk apply uniaxialStrain stress=-3e9 dir=[list 1 1 0]  
createGarandMCFfiles crystal=bulk
```

See the *Garand User Guide* for more information about importing band-structure files for Garand MC.

Analytic Band-Structure Models

In addition to EPM, Sentaurus Band Structure supports a six-band $\mathbf{k} \cdot \mathbf{p}$ method for computing valence-band energies and group velocities as well as analytic expressions for the lowest two conduction bands. Analytic band structures are computed by `AnalyticBandSolver` objects.

For example, you can create an `AnalyticBandSolver` object suitable for approximating silicon EPM results by using the command:

```
AnalyticBandSolverFromSpecies name=ana species=Si
```

An `AnalyticBandSolver` object can compute band energies at arbitrary \mathbf{k} -vectors. Due to the approximations inherent in the computation methods, analytic valence bands are inaccurate far away from the Γ -point; whereas, analytic conduction bands are most accurate in the vicinity of the band minima along the Δ -lines (that is, the global band minima of silicon-like band structures).

For silicon, Sentaurus Band Structure provides default parameters for both analytic conduction and valence bands. For germanium, only valence-band parameters are supplied because the analytic conduction-band expressions do not apply to materials whose L-valleys are below their Δ -valleys.

Chapter 10: Using Sentaurus Band Structure

Complex Band Structures

You can switch on or off the computation of the conduction-band or valence-band energies by setting the appropriate arguments of the `<AnalyticBandSolver> set` method (see [<AnalyticBandSolver> set on page 306](#)):

```
ana set conductionBands=Boolean  
ana set valenceBands=Boolean
```

The analytic band solver has an `<AnalyticBandSolver> computeBandstructure` method similar to that of `EPM::Crystal`. If `kList` is a list of **k**-vectors, then the corresponding analytic band-structure data is computed using:

```
ana computeBandstructure kVectors=$kList
```

By default, results are stored in the `ana.bandstructure` and `ana.groupVelocity` container objects.

Depending on the setting of `valenceBands`, the indices of the conduction bands can change. The index of the highest valence band stored in a `bandstructure_t` object can be queried by the following method:

```
<bandstructure_t> get topValenceBand
```

It returns `-1` if no valence bands are stored.

This means that the following Tcl command always returns the energy of the lowest conduction band at **k**-vector `K`:

```
ana.bandstructure get kVector=$K \  
band=[expr [ana.bandstructure get topValenceBand] +1]
```

Complex Band Structures

You can obtain the $E - \kappa$ dispersion relation for tunneling processes by solving the eigenvalue problem for the analytic continuation of the EPM Hamiltonian to complex **k**-vectors.

Real eigenvalues (complex eigenvalues do not correspond to physical states and are suppressed) of this non-Hermitian eigenvalue problem correspond to evanescent states in the sense of WKB tunneling theory. Sentaurus Band Structure supports only energy computation for complex **k**; group velocity and effective mass extraction are only available for real wavevectors.

The format for complex numbers in Sentaurus Band Structure is `(realPart,imaginaryPart)` (no spaces allowed). For example: `(0,1)` for $i = \sqrt{-1}$

Chapter 10: Using Sentaurus Band Structure

Calculating Electron Subbands and Mobility

All vector operations support complex vectors. For example, you can use the following commands to create a sequence of equally spaced \mathbf{k} -vectors between $\Gamma = (0, 0, 0)$ and

$$\mathbf{k}_{\text{end}} = 0.25 \frac{2\pi i}{a_0} (0, 1, 0) :$$

```
set kEnd {0 (0,0.25) 0}
set kList {}
for {set k 0} {$k <= 100} {incr k} {
    lappend kList [vectorMultiply v=$kEnd x=[expr $k/100.0]]
}
```

The complex band structure is computed using the commands:

```
sBandSet derivOrder=0 # derivatives not supported for complex k-vectors
SiliconCrystal name=bulk
bulk computeBandstructure kVectors=$kList nBands=16
```

In the real band structure, the conduction and valence bands are separated by the band gap; the complex bands link the conduction and valence bands.

Calculating Electron Subbands and Mobility

This section introduces the subband and mobility calculation features of Sentaurus Band Structure. It describes how to compute the self-consistent subbands of a 1D device as well as the electron inversion mobility.

Loading a 1D Device Structure

This example simulates a 1D bulk NMOS capacitor. The device structure is in a 1D TDR file called `nmoscap.tdr`. The device structure is loaded into Sentaurus Band Structure using the `LoadDevice` command:

```
LoadDevice tdrFile=nmoscap.tdr
```

This command loads device regions, the 1D grid, contacts, and the `DopingConcentration` field. After reading the TDR file, the `defaultPhysics.tcl` file is loaded automatically to set up several default models such as a set of valley models and the bulk carrier-density models for silicon. In addition, several default scattering models for silicon are created.

Specifying Top-Level Parameters of Physics Command

The orientation of the device axes and the temperature are specified using the top-level parameters of the `Physics` command (see [Physics for Top-Level Parameters on page 365](#)).

Chapter 10: Using Sentaurus Band Structure

Calculating Electron Subbands and Mobility

For example:

```
Physics surfaceOrientation=[list 0 0 1] xDirection=[list 1 0 0] \
temperature=297.0
```

This command sets the surface orientation, that is, the z-axis, to [001] and sets the x-axis to [100]. The ambient temperature is set to 297 K.

Performing an Initial Solve

Before solving the Schrödinger equation self-consistently, it is advisable to first solve with the classical density models. Since no Schrödinger solvers have yet been defined in this example, the following `Solve` command uses the default, bulk carrier-density models:

```
Solve V(gate)=0.5 initial
```

This command solves the Poisson equation at a gate bias of 0.5 V using a charge-neutral initial guess. After the solution converges, contact biases and other information are written to the active bias log file, that is, `eQuickStart.plt`.

Defining a Nonlocal Line

To compute the inversion layer, a nonlocal line is defined immediately under the oxide–silicon interface. This is performed using the `Math` command (see [Math on page 362](#)):

```
Math nonlocal name=NL1 minZ=0 maxZ=0.03
```

This command creates a nonlocal line starting at $z=0.0$, that is, the silicon–oxide interface in this example, and extending down 30 nm below the interface. The nonlocal line is named `NL1`, which allows you to refer to it in later commands.

Specifying a Schrödinger Solver

After the nonlocal line is defined, you use the `Physics` command to associate a Schrödinger solver to the nonlocal line for computing the confined carrier density (see [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#)). For example:

```
Physics nonlocal=NL1 eSchrodinger=Parabolic \
valleys=[list Delta1 Delta2 Delta3] Nk=32 Nphi=16 Nsubbands=32 \
Kmax=0.25
```

Here, the `Parabolic` Schrödinger solver is assigned to the previously created nonlocal line `NL1`. This solver will solve for the valleys named `Delta1`, `Delta2`, and `Delta3`. These valleys were created by default for all silicon regions in the `defaultPhysics.tcl` file. Parameters controlling the polar grid for computing the subband dispersion are given as well. In

Chapter 10: Using Sentaurus Band Structure

Calculating Electron Subbands and Mobility

particular, a radial polar grid out to `Kmax=0.25` with 32 points is specified along with an angular φ -grid with 16 points. The Schrödinger solver is also specified to solve for the 32 lowest-lying subbands in each valley.

Specifying a Mobility Calculator

After a nonlocal line and Schrödinger solver for that nonlocal line have been defined, you can specify a mobility calculator to use for the same nonlocal line by using another `Physics` command (see [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#)). For example:

```
Physics nonlocal=NL1 eMobilityCalculator=KGFromK Nk=64 Nphi=32
```

This command specifies to use the `KGFromK` mobility calculator to compute the electron mobility for this nonlocal line. The mobility calculation is performed over a polar grid, and the `Nk` and `Nphi` arguments specify details about this polar grid.

Performing a Self-Consistent Solve With the Schrödinger Equation

After a nonlocal line and Schrödinger solver for that nonlocal line have been defined, subsequent solutions of the Poisson equation will solve the Poisson equation self-consistently with your specified Schrödinger equation. To solve at the last converged bias point, use:

```
Solve
```

This command solves the Poisson equation at the bias point from the previous solve, that is, 0.5 V on the gate.

Computing the Mobility

After the Schrödinger equation has been solved, which necessarily computes the subband dispersion and wavefunctions, the inversion mobility can be computed with the previously defined mobility calculator using the `ComputeMobility` command (see [ComputeMobility on page 356](#)). For example:

```
ComputeMobility xx
```

This command computes the `xx` component of the mobility tensor and adds it to the active bias log file. Here, the `xx` is relative to the in-plane device coordinate system, which was previously defined using the `xDirection` argument of the `Physics` command. The result of this calculation is added to the active bias log file under the field name `mobilityxx`.

The scattering models that are used in the mobility calculation were created by default for all silicon regions in the `defaultPhysics.tcl` file.

Saving Models Over the Device Structure

Various physical models over the 1D device structure are saved to a TDR file by using the `Save` command (see [Save on page 383](#)). For example:

```
Save tdrFile=eQuickStartZ.tdr \
    models=[list ConductionBandEnergy eQuasiFermiEnergy \
    Delta3_0_SubbandEnergy Delta3_0_Wavefunction]
```

This command creates a file called `eQuickStartZ.tdr`. The models specified by `models` are saved as a function of the position `z`. Here, the conduction band energy and electron quasi-Fermi energy are saved. In addition, two subband-related quantities for the `Delta3` valley, that is, the lowest-lying ladder, are saved: the minimum energy and the wavefunction for the 0th subband of `Delta3`.

Saving Models Over 2D k-Space

Details of the subband dispersion and the momentum relaxation rates can be investigated by saving a 2D TDR file with **k**-space models for the subband dispersion and inverse momentum relaxation time (IMRT). These models are saved using the `SaveK` command (see [SaveK on page 385](#)). For example:

```
SaveK tdrFile=eQuickStartK.tdr \
    models=[list Delta3_0_Dispersion Delta3_0_IMRT]
```

This command saves a file called `eQuickStartK.tdr` with the **k**-space dispersion of the lowest-lying subband in this example, the `Delta3_0` subband. In addition, the IMRT for this subband is saved. The IMRT models can be used to look at the strength and anisotropy of the scattering.

Ramping the Bias

While no built-in bias ramping feature is available, bias ramps can be performed using the Tcl commands `for` or `foreach`. In the following example, the command block ramps the gate bias over three values. For each bias, a `Solve` is performed and an effective field is computed and added to the bias log file. Finally, at each bias the mobility is computed. The results of these calculations are saved in a new bias log file named `biasRamp.plt`:

```
foreach Vg [list 0.5 1.0 1.5] {
    # Solve at gate bias
    Solve V(gate)=$Vg logfile=biasRamp.plt

    # Extract Eeff by integrating eEeffIntegrand and add to bias log file
    set Eeff [Extract model=eEeffIntegrand region=sil integral]
    AddToLogFile name=Eeff value=$Eeff
```

Chapter 10: Using Sentaurus Band Structure

Calculating Electron Subbands and Mobility

```
# Compute xx component of mobility
ComputeMobility xx
}
```

The effective electric field that is used here is based on a predefined model that uses a sheet-charge formulation. The actual value of the effective field is computed by integrating the model with the name `eEffIntegrand` over the single silicon region in the device called `sil` using the `Extract` command. This formulation of the effective field value is used typically in measurements of mobility and provides a convenient way of comparing mobility versus effective-field data.

11

Empirical Pseudopotential Method

This chapter discusses the empirical pseudopotential method (EPM) that is used to compute the electronic structure of crystalline solids.

Introduction to Pseudopotentials

The EPM is used to compute the electronic structure of crystalline solids. To make this problem computationally tractable, the many-electron system of the crystal is approximated by a single-electron model, in which the interactions between the electrons are described by an effective potential (mean field approximation). Due to the highly oscillatory nature of the electron wavefunctions close to the atomic nuclei, the exact evaluation of the solutions of this effective single-electron system still requires a high computational effort. This effort can be reduced by the pseudopotential method.

The pseudopotential method further reduces computational complexity by separating the electrons of each atom in the crystal into inner (*core*) and outer (*valence*) electrons.

Chemical bond formation in a material is dominated by the valence electrons. Core electrons are closely localized around the nuclei and do not participate in bond formation; their states are assumed to be the same as in an isolated atom.

Now, it is necessary to formulate equations that only involve possible states of the valence electrons. Because of the Pauli exclusion principle, the wavefunction $|\psi\rangle$ of such a state must be orthogonal to the core states $|c\rangle$ of all atoms (c is a collective label for the position of the nucleus and all internal quantum numbers of the core states).

An arbitrary function $|\phi\rangle$ can be converted to a function that is orthogonal to the core states by subtracting its projection onto all core states:

$$|\psi\rangle := |\phi\rangle - \sum_c |c\rangle \langle c| \phi \quad (54)$$

Then, the Schrödinger equation for $|\psi\rangle$ can be rewritten in terms of $|\phi\rangle$:

$$H|\psi\rangle = E|\psi\rangle \quad (55)$$

Chapter 11: Empirical Pseudopotential Method

Introduction to Pseudopotentials

$$\left(H - \sum_{\mathbf{c}} H |\mathbf{c}\rangle\langle\mathbf{c}| \right) |\phi\rangle = E \left(|\phi\rangle - \sum_{\mathbf{c}} |\mathbf{c}\rangle\langle\mathbf{c}| \phi \right) \quad (56)$$

$$\underbrace{\left(H + \sum_{\mathbf{c}} (E - E_{\mathbf{c}}) |\mathbf{c}\rangle\langle\mathbf{c}| \right) |\phi\rangle}_{=: V^R} = E |\phi\rangle \quad (57)$$

In [Equation 57](#), the effect of the core electrons has been rewritten as an additional repulsive potential term V^R (the potential is repulsive because valence energies E are larger than core energies $E_{\mathbf{c}}$).

Instead of the original Hamiltonian $H = -\frac{\hbar^2}{2m} \nabla^2 + V$, an effective Hamiltonian $H = -\frac{\hbar^2}{2m} \nabla^2 + \tilde{V}$ is obtained with the pseudopotential $V := V + V^R$.

The additional potential term V^R is both nonlocal and energy dependent; however, the most important features of the band structure often can be captured by an energy-independent local approximation to V^R .

For some materials, there is significant variation in published pseudopotential parameters. For example, the silicon pseudopotential used by [\[1\]](#) is mostly local with only a small nonlocal correction; whereas, the nonlocal parameterization of silicon in [\[2\]](#) has such a strong nonlocal potential that its removal would result in a direct band gap.

The importance of [Equation 57](#) is that, even though it involves the *pseudo-wavefunction* $|\phi\rangle$ rather than the true wavefunction $|\psi\rangle$, the energy E is the true eigen-energy of the wavefunction that corresponds to $|\phi\rangle$ in the sense of [Equation 54](#).

In contrast to the true wavefunction $|\psi\rangle$, the pseudo-wavefunction $|\phi\rangle$ does not need to obey any orthogonality restriction and, therefore, it can be expanded in an arbitrary basis, for example, plane waves.

In a periodic crystal, the Bloch theorem can be invoked: The (pseudo-)wavefunction can be written as a product of a plane wave and a lattice periodic function $u(\mathbf{r}) = \sum_{\mathbf{G}} c_{\mathbf{G}} \exp(i\mathbf{G} \cdot \mathbf{r})$, where the sum is taken over all reciprocal lattice vectors \mathbf{G} :

$$\phi_{n\mathbf{k}}(\mathbf{r}) = u_{n\mathbf{k}}(\mathbf{r}) \exp(i\mathbf{k} \cdot \mathbf{r}) = \sum_{\mathbf{G}} c_{\mathbf{G}}^{(n\mathbf{k})} \exp(i(\mathbf{G} + \mathbf{k}) \cdot \mathbf{r}) \quad (58)$$

Substituting this into [Equation 57](#) results in the following equation for the Fourier coefficients of $u_{n\mathbf{k}}$:

$$\sum_{\mathbf{G}} \left(\frac{\hbar^2}{2m} (\mathbf{G}' + \mathbf{k})^2 + \langle \mathbf{G} + \mathbf{k} | \tilde{V} | \mathbf{G}' + \mathbf{k} \rangle \right) c_{\mathbf{G}'}^{(n\mathbf{k})} = E_{n\mathbf{k}} c_{\mathbf{G}}^{(n\mathbf{k})} \quad (59)$$

Empirical Pseudopotentials

In principle, the *orthogonalization potential* V^R can be obtained by solving the electronic structure of each atomic species in the crystal and by attaching projectors onto the resulting core electron states to each lattice site occupied by an atom of said species. In the EPM, however, the pseudopotential is expressed in terms of a small number of empirical parameters, which are obtained by fitting the band structure to experimental data or *ab initio* data.

The small size of the EPM parameter set arises from the observation that, in contrast to the bare ion potential V (which has a singularity at each atomic site), the local part of the pseudopotential V_{loc} is a smooth function of position without short-ranged oscillations. The periodicity of the crystal lattice implies that V_{loc} can be expanded in terms of plane waves whose wavevectors are taken from the reciprocal crystal lattice.

The locality of \tilde{V}_{loc} implies that its plane-wave matrix elements $\langle \mathbf{K} | V_{\text{loc}} | \mathbf{K}' \rangle$ depend only on the magnitude of the crystal momentum transfer: $\langle \mathbf{K} | V_{\text{loc}} | \mathbf{K}' \rangle = V_{\text{loc}}(\|\mathbf{K} - \mathbf{K}'\|)$. The absence of short-ranged oscillations in V_{loc} allows truncation of the local pseudopotential of large momentum transfer vectors $\mathbf{q} := \mathbf{K} - \mathbf{K}'$.

It is traditional to include only terms up to $\|\mathbf{q}\| = \sqrt{11} \cdot \frac{2\pi}{a}$ in the local pseudopotential of unstrained cubic crystals (a is the unstrained lattice constant of the crystal), with the claim that terms with larger $\|\mathbf{q}\|$ are ‘small’. Changing the $\|\mathbf{q}\|$ cut-off, however, might have a pronounced effect on the resulting band structure, especially in the presence of shear strain (which reduces the crystal symmetry; compare with [Using the Crystal Symmetry](#)). This shows that the matrix elements, which traditionally are neglected, are not small in a strict sense; however, since a $\|\mathbf{q}\|$ cut-off was used to calibrate the V_{loc} parameters, it needs to be retained for consistency.

Using the Crystal Symmetry

Depending on the crystal symmetry, a large number of matrix elements will vanish even within the cut-off sphere: The (pseudo)potential inside a crystal is the (pseudo)potential around an individual ion (valence electrons removed) convolved with the sublattice occupied by that atomic species:

$$V(\mathbf{r}) = \left(\sum_{\alpha}^{\text{species}} V_{\alpha}^{\text{ion}} * \sum_{\mathbf{R} \in R_{\alpha}}^{\text{unit cells}} \sum_{\tau}^{\alpha \text{ atoms in cell}} \delta_{\mathbf{R} + \tau} \right)_{\mathbf{r}} \quad (60)$$

where $\delta_{\mathbf{R} + \tau}$ denotes the three-dimensional Dirac delta-distribution shifted to position $\mathbf{R} + \tau$.

Chapter 11: Empirical Pseudopotential Method

Introduction to Pseudopotentials

The double sum over unit cells and the basis (=positions of the atoms in each unit cell relative to the origin of the cell) can be rewritten as a convolution of the primitive lattice with the basis:

$$\sum_{\text{unit cells}} \sum_{\mathbf{R} \in R_\alpha} \sum_{\tau} \delta_{\mathbf{R} + \tau} = \left(\sum_{\text{unit cells}} \sum_{\mathbf{R} \in R_\alpha} \delta_{\mathbf{R}} \right) * \left(\sum_{\tau} \delta_{\tau} \right) \quad (61)$$

so that its Fourier transform takes the form of a product:

$$\left[\sum_{\text{unit cells}} \sum_{\mathbf{R} \in R_\alpha} \sum_{\tau} \delta_{\mathbf{R} + \tau} \right] \hat{\wedge}(\mathbf{q}) = \sum_{\mathbf{G} \in G_\alpha} \underbrace{\delta_{\mathbf{G}}}_{\text{reciprocal lattice}} \underbrace{\sum_{\tau} \exp(-i\mathbf{q} \cdot \tau)}_{\alpha \text{ atoms in cell}} \quad (62)$$

=: S_\alpha(\mathbf{q}) \text{ (natural definition)}

where the structure factor S_α is introduced as a quantity, which encodes all relevant information on the positions of atoms of species α .

Note that this *natural* definition of the structure factor in [Equation 62](#) is not the definition usually used in the literature. The *conventional* definition of the structure factor is:

$$S_\alpha(\mathbf{q}) := \frac{1}{N_{\text{atoms}}} \sum_{\tau} \exp(-i\mathbf{q} \cdot \tau) \quad (63)$$

and a factor of N_{atoms} (the total number of atoms in the unit cell) must be included in all potential energy components for consistency (this is equivalent to using plane waves normalized on the atomic volume $\Omega_a := \Omega_c / N_{\text{atoms}}$ instead of the unit cell volume Ω_c). To be able to use published pseudopotential parameterizations, Sentaurus Band Structure uses the conventional definition of the structure factor.

In terms of the structure factor, the Fourier transform of the local potential takes the form:

$$\hat{V}(\mathbf{q}) = \sum_{\alpha} \text{species} \hat{V}_\alpha^{\text{ion}}(\mathbf{q}) S_\alpha(\mathbf{q}) \sum_{\mathbf{G} \in G_\alpha} \text{reciprocal lattice} \delta(\mathbf{q} - \mathbf{G}) \quad (64)$$

Since the potential must obey the periodicity of the crystal lattice, its Fourier transform can only contain reciprocal lattice vectors as expressed by the last term of [Equation 64](#). This turns the Fourier integral:

$$V(\mathbf{r}) = \frac{1}{(2\pi)^3} \int d^3 q \exp(i\mathbf{q} \cdot \mathbf{r}) \hat{V}(\mathbf{q}) \quad (65)$$

into a discrete sum:

$$V(\mathbf{r}) = \frac{1}{(2\pi)^3} \sum_{\mathbf{G}} \exp(i\mathbf{G} \cdot \mathbf{r}) \sum_{\alpha} \hat{V}_\alpha^{\text{ion}}(\mathbf{q}) S_\alpha(\mathbf{q}) \quad (66)$$

Chapter 11: Empirical Pseudopotential Method

Introduction to Pseudopotentials

Consequently, Fourier components of the ionic potential are needed only at reciprocal lattice vectors. Many of the Fourier components at reciprocal lattice vectors disappear because the structure factor vanishes. In (unstrained) crystals of diamond structure, for example, the local pseudopotential has only three different nonvanishing contributions inside the conventional $\|\mathbf{q}\| \leq \frac{2\pi}{a} \sqrt{11}$ cut-off sphere:

$$\|\mathbf{q}\| \in \left\{ \sqrt{3} \cdot \frac{2\pi}{a}, \sqrt{8} \cdot \frac{2\pi}{a}, \sqrt{11} \cdot \frac{2\pi}{a} \right\} \quad (67)$$

In the absence of strain, the next higher nonvanishing contribution is at $\|\mathbf{q}\| = \frac{2\pi}{a} \cdot 4$. The symmetry reduction caused by the application of shear strain gives rise to additional terms with $\|\mathbf{q}\| \approx \frac{2\pi}{a} \cdot \sqrt{12}$. Therefore, the choice of the radius of the cut-off sphere is more critical in the presence of shear strain than in relaxed or orthorhombic strained crystals.

Nonlocal Corrections to the Pseudopotential

To obtain a better fit to data, it is usual to introduce nonlocal corrections to the purely local empirical pseudopotential of the previous section. The nonlocal contribution to the atomic form-factor of species α is usually expressed as:

$$V_{NL}^{(\alpha)}[\psi](\mathbf{r}) = \sum_{l=0}^{\infty} \langle \mathbf{r} | \underbrace{A_l^{(\alpha)} f_l(r) P_l}_{=: V_l^{(\alpha)}} | \psi \rangle \quad (68)$$

where:

- f_l models the effect of core states of angular momentum l .
- P_l is the projector on the subspace of angular momentum l wavefunctions centered around $\mathbf{r} = 0$.

The model potential f_l is chosen to be either a step function $f_l(r) = \Theta(r - R_l)$ or a Gaussian $f_l(r) = \exp(-r^2/R_l^2)$.

For $l = 0$, Sentaurus Band Structure uses a step-function model potential. For $l = 2$, you can choose either a step-function potential well (`nonLocalWell=Square`) or a Gaussian potential well (`nonLocalWell=Gaussian`). All other nonlocal corrections are set to zero. In spherical coordinates, the angular momentum projector takes the form:

$$\langle r, \theta, \phi | P_l | \psi \rangle = \sum_{m=-l}^l Y_{lm}(\theta, \phi) \int_0^{2\pi} d\phi' \int_{-1}^1 d\cos(\vartheta') Y_{lm}^*(\theta', \phi') \psi(r, \vartheta', \phi') \quad (69)$$

where Y_{lm} denotes the spherical harmonics.

Chapter 11: Empirical Pseudopotential Method

Introduction to Pseudopotentials

With this, the matrix elements for the $l = 0$ nonlocal potential with a square-well model potential can be written as ($K := \|\mathbf{K}\|$ and so on):

$$\begin{aligned} \langle \mathbf{K}' | A_l \Theta(R_l - r) P_l | \mathbf{K} \rangle &= \frac{A_l}{\Omega} \int_0^{R_l} r^2 dr \int_{-\pi}^{\pi} d\phi \int_{-1}^1 d\cos(\theta) \exp(-iK'r \cos(\theta)) Y_{00}^*(\theta, \phi) \\ &\quad \times \int_{-\pi}^{\pi} d\phi' \int_{-1}^1 d\cos(\theta') Y_{00}(\theta', \phi') \exp(iK'r \cos(\theta')) \\ &= \frac{4\pi A_l}{\Omega K K'} \int_0^{R_l} dr \sin(Kr) \sin(K'r) \\ &= \frac{4\pi A_l}{\Omega} \begin{cases} \frac{1}{K^2} \left(R_l - \frac{1}{K} \sin(K R_l) \cos(K R_l) \right) & K = K' \\ \frac{1}{K^2 - K'^2} (K' \sin(K' R_l) \cos(K R_l) - K \sin(K R_l) \cos(K' R_l)) & K \neq K' \end{cases} \end{aligned} \quad (70)$$

Ω is the plane-wave normalization volume; the definition of the structure factor from [Equation 63 on page 150](#) implies $\Omega = \Omega_a$ (the alternative definition without dividing by N_{atoms} implies $\Omega = \Omega_c$).

For arbitrary l , this generalizes to [2]:

$$\begin{aligned} \langle \mathbf{K}' | A_l \Theta(R_l - r) P_l | \mathbf{K} \rangle &= \frac{4\pi(2l+1)}{\Omega} A_l P_l(\cos(\angle(\mathbf{K}, \mathbf{K}'))) \\ &\times \begin{cases} \frac{1}{2} R_0^3 ((j_l(KR_l))^2 - j_{l-1}(KR_l)j_{l+1}(KR_l)) & K = K' \\ \frac{R_0^2}{K^2 - K'^2} K j_{l+1}(KR_l) j_l(K'R_l) - K' j_{l+1}(K'R_l) j_l(KR_l) & K \neq K' \end{cases} \end{aligned} \quad (71)$$

Here, P_l are the Legendre polynomials. For $l \geq 0$, j_l are spherical Bessel functions of the first kind; negative indices indicate spherical Bessel functions of the second kind (sometimes called spherical Neumann functions) according to the notational convention:

$$j_{-n}x := (-1)^n y_{n-1}(x), n \geq 1 \quad (72)$$

The corresponding expression for the $l = 2$ Gaussian well commonly used in conjunction with silicon and gallium arsenide is:

$$\langle \mathbf{K}' | A_2 \exp(-(r/R_2)^2) P_2 | \mathbf{K} \rangle = 5\pi^{\frac{3}{2}} A_2 \frac{R_2^3}{\Omega} P_2(\cos(\angle(\mathbf{K}, \mathbf{K}'))) i_2\left(\frac{1}{2} R_2^2 K K'\right) e^{-\frac{1}{4} R_2^2 (K^2 + K'^2)} \quad (73)$$

Chapter 11: Empirical Pseudopotential Method

Introduction to Pseudopotentials

where i_2 denotes the modified spherical Bessel function of the first kind:

$$i_2(x) = \frac{(x^2 + 3)\sinh(x) - 3x\cosh(x)}{x^3} \quad (74)$$

The prefactors A_l are treated usually as energy-independent fitting constants; Chelikowsky and Cohen [2] make the prefactor for the $l = 0$ nonlocal potential energy-dependent by choosing the parameterization:

$$A_0(K, K) = \alpha_0 + \beta_0 \frac{\hbar^2}{2m_0} (KK - k_F^2) \quad (75)$$

where:

$$k_F = \sqrt[3]{\frac{3\pi N_{\text{val}}^{\text{tot}}}{\Omega_c}} \quad (76)$$

is the Fermi wavevector of a Fermi gas of electron density $n = \frac{N_{\text{val}}^{\text{tot}}}{\Omega_c}$ and $N_{\text{val}}^{\text{tot}}$ is the number of valence electrons in the unit cell of the crystal.

Spin-Orbit Coupling

Following the literature [1], spin-orbit coupling is included in the calculation by adding a nonlocal spin-orbit coupling potential to the atomic structure factor of each species α :

$$V_{LS}^{(\alpha)} = \sum_{n,l} V_{\alpha,n,l}^{(LS)} \mathbf{L} \cdot \mathbf{S} P_{n,l}^{(\alpha)} \quad (77)$$

Here, $P_{n,l}^{(\alpha)}$ is the projector onto the subspace spanned by the core states of species α with principal quantum number n and angular momentum quantum number l :

$$P_{n,l}^{(\alpha)} = \sum_{m=-l}^l |n, l, m\rangle^{(\alpha)(\alpha)} \langle n, l, m| \quad (78)$$

The core wavefunctions $|n, l, m\rangle^{(\alpha)}$ can be either taken from *ab initio* methods (Chelikowsky and Cohen [2] use the radial functions tabulated in [3]) or approximated by analytic expressions.

Sentaurus Band Structure follows [1] in pursuing the latter approach. This will later introduce the length scale $\zeta_{nl}^{(\alpha)}$ of the radial wavefunction as an additional fitting parameter.

Typically, the dominant spin-orbit coupling contribution is the $l = 1$ term of the outermost closed shell ($n = N - 1$, where N is the period of element α). Other terms are usually omitted.

Chapter 11: Empirical Pseudopotential Method

Introduction to Pseudopotentials

In evaluating the dominant $l = 1$ term, it is convenient to switch from the usual Y_{1m} basis to the chemical p_x, p_y, p_z basis:

$$p_x(\theta, \phi) = \sqrt{\frac{3}{4\pi}} \frac{x}{r} = \sqrt{\frac{3}{4\pi}} \sin(\theta) \cos(\phi) \quad (79)$$

$$p_y(\theta, \phi) = \sqrt{\frac{3}{4\pi}} \frac{y}{r} = \sqrt{\frac{3}{4\pi}} \sin(\theta) \sin(\phi) \quad (80)$$

$$p_z(\theta, \phi) = \sqrt{\frac{3}{4\pi}} \frac{z}{r} = \sqrt{\frac{3}{4\pi}} \cos(\theta) \quad (81)$$

because of their vector-like transformation behavior. Writing the p_λ -state of shell n of species α as:

$$\langle r, \theta, \phi | n, 1, \lambda \rangle^{(\alpha)} = f_{n1}^{(\alpha)}(r) p_\lambda(\theta, \phi) \quad (82)$$

the projection of a normalized plane wave $|\mathbf{K}\rangle$ onto this state takes the form:

$$\begin{aligned} & \langle r, \theta, \phi | n, 1, \lambda \rangle^{(\alpha)(\alpha)} \langle n, 1, \lambda | \mathbf{K} \rangle \\ &= \frac{1}{\sqrt{\Omega_c}} f_{n1}^{(\alpha)}(r) p_\lambda(\theta, \phi) \int_0^\infty r^2 dr \int_0^{2\pi} d\phi' \int_{-1}^1 d\cos(\theta') p_\lambda(\theta', \phi') f_{n1}^{(\alpha)}(r') \\ &= \frac{3i}{\sqrt{\Omega_c}} f_{n1}^{(\alpha)}(r) p_\lambda(\theta, \phi) \mathbf{K} \cdot \hat{\mathbf{e}}_\lambda \underbrace{\frac{1}{K} \int_0^\infty r^2 dr' j_1(Kr') f_{n1}^{(\alpha)}(r')}_{=: B_{n1}^{(\alpha)}(K)} \end{aligned} \quad (83)$$

with the radial overlap integral $B_{n1}^{(\alpha)}(r)$. Note the factor $\frac{1}{K}$ in front of the integral in the definition of the overlap integral. In place of $\frac{1}{K}$, Chelikowsky and Cohen [2] have ‘a normalization constant $[\beta]$ as in reference [4].

However, the normalization convention used in [4] is $\lim_{K \rightarrow 0} B_{nl}^{(\alpha)}(r) = 1$, which implies a division by K for functions that exhibit linear behavior in K close to $K = 0$. The evaluation of the overlap integral in terms of analytic expressions for the radial core wavefunctions is discussed in [Evaluating Radial Overlap Integrals on page 155](#).

Now, the plane-wave matrix element of the projected spin-orbit coupling operator $\mathbf{L} \cdot \mathbf{S} P_{n1} = \mathbf{r} \times \mathbf{p} \cdot \mathbf{S} P_{n1}$ is evaluated.

Without loss of generality, you can choose coordinates such that $\mathbf{K} \parallel \hat{\mathbf{e}}_z$.

Chapter 11: Empirical Pseudopotential Method

Introduction to Pseudopotentials

Then, the matrix element takes the form (atomic species index α omitted):

$$\begin{aligned}
 \langle \mathbf{K}' | \mathbf{r} \times \mathbf{p} P_{n1} | \mathbf{K} \rangle \cdot \langle \chi' | \mathbf{S} | \chi \rangle &= -\hbar (\mathbf{K}' \times \langle \mathbf{K}' | \mathbf{r} P_{n1} | \mathbf{K} \rangle) \cdot \langle \chi' | \mathbf{S} | \chi \rangle \\
 &= \frac{3\hbar\pi i}{\Omega_c} B_{n1}(K) \left[\mathbf{K}' \times \int_0^\infty r^2 dr f_{n1}(r) \int_0^{2\pi} d\phi \int_{-1}^1 d\cos(\theta) e^{-iK'r\cos(\theta)} \begin{pmatrix} K_x x^2/r \\ K_y y^2/r \\ K_z z^2/r \end{pmatrix} \right] \cdot \langle \chi' | \mathbf{S} | \chi \rangle \\
 &= \frac{3\hbar\pi^2 i}{\Omega_c} B_{n1}(K) \left[\mathbf{K}' \times \int_0^\infty r^3 dr f_{n1}(r) \int_{-1}^1 du \exp(-iK'r u) \begin{pmatrix} K_x(1-u^2) \\ K_y(1-u^2) \\ K_z u^2 \end{pmatrix} \right] \cdot \langle \chi' | \mathbf{S} | \chi \rangle \\
 &= \frac{12\hbar\pi^2 i}{\Omega_c} B_{n1}(K) \left[\mathbf{K}' \times \frac{1}{K} \int_0^\infty r^2 dr j_1(K'r) f_{n1}(r) \begin{pmatrix} K_x \\ K_y \\ (\text{irrelevant}) \end{pmatrix} \right] \cdot \langle \chi' | \mathbf{S} | \chi \rangle \\
 &= \frac{12\hbar\pi^2 i}{\Omega_c} B_{n1}(K) B_{n1}(K') (\mathbf{K}' \times \mathbf{K}) \cdot \langle \chi' | \mathbf{S} | \chi \rangle
 \end{aligned} \tag{84}$$

where $\mathbf{K} \parallel \mathbf{e}_z$ is invoked to exclude contributions from the z -component of the vector integral from the cross product, and you will recognize the definition of B from [Equation 83](#).

Evaluating Radial Overlap Integrals

In Sentaurus Band Structure, analytic hydrogen-like radial functions are used to evaluate the overlap integrals $B_{nl}^{(\alpha)}$. The wavefunctions of the hydrogen atom have the form:

$$\psi_{lm}(r, \theta, \phi) \propto e^{-\rho/2} L_{n-l-1}^{2l+1}(\rho) Y_{lm}(\theta, \phi) \tag{85}$$

$$\rho := \frac{2r}{na_B} \quad (\text{Hydrogen}) \tag{86}$$

where $L_n^m(\rho)$ are the generalized Laguerre polynomials, and a_B is the Bohr radius.

To accommodate the nuclear charge and screening in species α , you introduce an adjustable length scale parameter $\zeta_{nl}^{(\alpha)}$ and replace ρ from [Equation 86](#) with the expression:

$$\tilde{\rho} := \frac{2\zeta_{nl}^{(\alpha)} r}{a_B} \quad (\text{species } \alpha) \tag{87}$$

Then, the radial part of the wavefunction is:

$$f(r) \propto e^{-\tilde{\rho}/2} L_{n-l-1}^{(2l+1)}(\tilde{\rho}) \tag{88}$$

with the normalization condition:

$$\int_0^\infty r^2 (f(r))^2 dr = 1 \quad (89)$$

Substituting this into the definition of $B_{nl}^{(\alpha)}$ (see [Equation 83 on page 154](#)) yields the following results in terms of the dimensionless quantity $x := \frac{a_B}{\zeta_{nl}^{(\alpha)}} K$:

$$B_{21}^{(\alpha)}(K) = \frac{16}{\sqrt{3}} \left(\frac{a_B}{\zeta_{21}^{(\alpha)}} \right)^{5/2} \frac{1}{(1+x^2)^3} \quad (90)$$

$$B_{31}^{(\alpha)}(K) = 16\sqrt{2} \left(\frac{a_B}{\zeta_{31}^{(\alpha)}} \right)^{5/2} \frac{1-x^2}{(1+x^2)^4} \quad (91)$$

$$B_{41}^{(\alpha)}(K) = \frac{32}{\sqrt{15}} \left(\frac{a_B}{\zeta_{41}^{(\alpha)}} \right)^{5/2} \frac{5-14x^2+5x^4}{5(1+x^2)^5} \quad (92)$$

A slightly different choice of model orbitals (replace $L_{n-l-1}^{(2l+1)}(\tilde{p})$ with \tilde{p}^{n-l-1} in [Equation 88](#)) leads to the functions:

$$B_{31}^{(\alpha)}(K) = 16 \frac{\sqrt{2}}{\sqrt{5}} \left(\frac{a_B}{\zeta_{31}^{(\alpha)}} \right)^{5/2} \frac{5-x^2}{5(1+x^2)^4} \quad (93)$$

$$B_{41}^{(\alpha)}(K) = \frac{32}{\sqrt{35}} \left(\frac{a_B}{\zeta_{41}^{(\alpha)}} \right)^{5/2} \frac{5-3x^2}{5(1+x^2)^5} \quad (94)$$

$B_{21}^{(\alpha)}$ is unchanged because of $L_0^{(3)} = 1$. Up to a constant prefactor, $B_{31}^{(\alpha)}(K)$ is identical to the overlap integral $B(K)$ for germanium of [1]. To keep spin-orbit parameters from [1] usable, Sentaurus Band Structure implements the overlap integrals using only the leading terms of the Laguerre polynomials.

Normalizing the Spin-Orbit Term

It is customary (compare to [2] for example) to drop all real constant prefactors from the spin-orbit matrix element of [Equation 84 on page 155](#), to replace the spin vector-operator $S = \frac{\hbar}{2}\sigma$ with the dimensionless operator $\frac{1}{2}\sigma$ (where σ is the formal vector of Pauli spin matrices), and to replace $B_{nl}^{(\alpha)}$ with the dimensionless function:

$$\tilde{B}_{nl}^{(\alpha)}(K) := \frac{B_{nl}^{(\alpha)}(K)}{\lim_{K' \rightarrow 0} B_{nl}^{(\alpha)}(K')} \quad (95)$$

Then, the spin-orbit contribution to the atomic form factor (with \mathbf{K} and \mathbf{K}' treated as dimensionless vectors in units of $2\pi/a_0$) reads (obvious indices omitted):

$$\langle \mathbf{K}'\chi | V_{LS} | \mathbf{K}\chi \rangle = -i\mu \tilde{B}(K)\tilde{B}(K')\mathbf{K}' \times \mathbf{K} \cdot \langle \chi' | \sigma | \chi \rangle \quad (96)$$

where μ is an energy parameter that controls the magnitude of the splitting. For correct scaling of the spin-orbit term in strained materials, it is necessary to return to [Equation 84](#): The matrix element is proportional to $1/\Omega_c^{\text{refc}}$. Therefore, under strain or in alloys, the spin-orbit terms must be multiplied by $\Omega_c^{\text{refc}}/\Omega_c^{\text{strained}}$.

Strained Materials

This section presents details about strained materials.

Bulk Strain and Internal Strain

Homogeneous strain has two effects on a crystalline material:

- It deforms the shape of the unit cell according to $\mathbf{r}' = (1 + \varepsilon)\mathbf{r}$ where ε is the bulk strain tensor.
- It can rearrange the ions within a unit cell in a nonbulk-like way (internal strain) [5].

To illustrate the origin of internal strain, consider a silicon atom and its four neighbors in a relaxed silicon crystal. The silicon atom sits at the center of a regular tetrahedron (circumcenter and barycenter coincide in a regular tetrahedron) whose corners are the sites of the four neighbor atoms. Now, this coordination tetrahedron is deformed by applying strain. Applying the bulk strain tensor ε both to the corner atoms and the central atom moves the central atom to the barycenter of the deformed tetrahedron.

For non-orthorhombic strain ε , the barycenter τ_0 of the deformed tetrahedron is different from its circumcenter τ_1 , which means that the bonds to the four neighbor atoms have acquired unequal lengths.

Note:

Sentaurus Band Structure uses tensor strain, which differs from 'engineering strain' (strain as a six-component vector; Voigt notation) by a factor of $\frac{1}{2}$ in the off-diagonal terms.

This bond-length distortion results in a force that tends to move the central atom towards the circumcenter. It is, however, opposed by noncentral (bond-bending) and nonnearest neighbor force-components. Therefore, the equilibrium position of the central atom is expected to be somewhere between the barycenter and circumcenter.

In crystals of diamond or zinc-blende structure, the effect of internal strain can be empirically described by a single scalar parameter ζ (*internal strain parameter*), where $\zeta = 0$ corresponds to a pure bulk-like deformation of the unit cell (central atom at the barycenter of the deformed tetrahedron) and $\zeta = 1$ corresponds to equal bond lengths (central atom at

the circumcenter of the deformed tetrahedron); intermediate values of ζ refer to intermediate positions:

$$\tau(\zeta) = (1 - \zeta)\tau_0 + \zeta\tau_1 = \tau_0 + \underbrace{\zeta(\tau_1 - \tau_0)}_{=: \delta(\zeta)} \quad (97)$$

where you have introduced the ‘internal strain displacement’ δ as the deviation of the interatomic separation τ from the value predicted by bulk elasticity. Crystals of lower symmetry might need additional parameters to describe the effect of internal strain.

Empirical Pseudopotential Method for Strained Materials

To apply EPM to strained materials, the rigid-ion approximation [5] is invoked, that is, it is assumed that the pseudopotential around an atom does not change shape as the crystal is deformed. The effect of strain on the EPM pseudo-Hamiltonian in this approximation can be seen easily by considering [Equation 64](#) and [Equation 63](#). Deformation of the spanning vectors of the unit cell leads to different reciprocal lattice vectors $\mathbf{G} = (1 + \varepsilon)^{-1}\mathbf{G}_0$, which shifts the evaluation points of the atomic form-factors $V_\alpha(\mathbf{q})$ in [Equation 64](#) to different momentum transfer vectors \mathbf{q} .

From [Equation 63](#), you can see that, in the absence of internal strain displacement ($\delta = 0$), the value of the structure factor of the strained crystal at a deformed momentum transfer vector $\mathbf{q} = (1 + \varepsilon)^{-1}\mathbf{q}_0$ is the same as the vector of the structure factor of the undeformed crystal at the momentum transfer vector \mathbf{q}_0 :

$$\delta = 0 \Rightarrow \tilde{\tau} = (1 + \varepsilon)\tau \Rightarrow \tilde{\tau} \cdot \tilde{\mathbf{q}} = \tau \cdot ((1 + \varepsilon)(1 + \varepsilon)^{-1}\mathbf{q}_0) = \tau \cdot \mathbf{q}_0 \quad (98)$$

A nonzero internal strain displacement vector δ , on the other hand, might give rise to potential contributions from \mathbf{q} -vectors, which in the relaxed crystal would be suppressed by symmetry (for example, $\|\mathbf{q}\| = \frac{2\pi}{a_0}\{\sqrt{4}, \sqrt{12}\}$ in a diamond lattice under non-orthorhombic strain).

Since there are analytic expressions for the nonlocal potential and the spin-orbit term at arbitrary vectors \mathbf{K} and \mathbf{K}' , the new evaluation points raise no issues beyond the question of correct plane-wave normalization in strained materials (compare to [Plane-Wave Normalization in Strained Crystals on page 160](#)). The local pseudopotential, however, needs additional attention.

Local Pseudopotential in Strained Crystals

To describe the local pseudopotential in an unstrained diamond-structure crystal, it is sufficient to know the atomic form-factors at $\|\mathbf{q}\| \in \frac{2\pi}{a_0}\{\sqrt{3}, \sqrt{8}, \sqrt{11}\}$.

For zinc-blende, there are additional contributions from $\|\mathbf{q}\| \in \frac{2\pi}{a_0}\{\sqrt{4}, \sqrt{12}\}$.

Chapter 11: Empirical Pseudopotential Method

Strained Materials

In strained crystals, however, the atomic form-factor is needed at a multitude of $\|\tilde{\mathbf{q}}\|$ values both in the vicinity of the old sampling points and (in the presence of internal strain displacement) around \mathbf{q} locations that previously were suppressed by the crystal symmetry.

This requires an analytic expression for the local form-factor $\hat{V}_\alpha^{\text{ion}}(\|\tilde{\mathbf{q}}\|)$. In the literature, there are two choices for expanding the local pseudopotential sampling data to a functional form: cubic spline interpolation [1] and an analytic ansatz by Friedel *et al.* [6].

Cubic Spline Interpolation of Local Pseudopotential

In this approach, the local pseudopotential $V(q)$ for arbitrary q is obtained by putting a cubic spline through the pseudopotential sample values of the unstrained crystal (for example, $V(q)$ at $q \in \frac{2\pi}{a_0} \{\sqrt{3}, \sqrt{8}, \sqrt{11}\}$ for diamond structures).

The spline is made unique by requiring:

- $V(0) = -\frac{2}{3}\epsilon_F$
- $(V(3k_F) = 0)$
- $V'(3k_F) = 0$
- $V''(0) = 0$

with the Fermi wavevector k_F as in [Equation 76](#) and Fermi energy $\epsilon_F = \frac{\hbar^2 k_F^2}{2m_0}$.

This is the method used for the treatment of strain by Rieger and Vogl [1]. There is no obvious reason why the interpolation spline through this minimal parameter set should give the correct strain response of the band structure. However, in the presence of orthorhombic strain, it yields good agreement with theoretical valence-band splitting results published by Van de Walle [7].

Additional sample values can be added to the spline to adjust the strain response without altering the unstrained band structure. If $V(0)$ is expressly supplied, it overrides the default value of $V(0) = -\frac{2}{3}\epsilon_F$.

The above cubic spline is typically the C2 spline with maximum smoothness where the continuity for the samples and the first-order and second-order derivatives at knots is fulfilled at the same time. However, the C1 spline also can be used. For the C1 spline, the continuity for the samples and the first-order derivative at knots is fulfilled, while the conservation of the second-order derivative at knots is not necessarily fulfilled. Therefore, the C1 spline is uniquely determined by not only the sample, but also the slope at knots. It is worth noting that the slope at knots in the C2 spline is not an independent parameter, but it is determined by the spline. Whereas, the slope at knots in the C1 spline is an independent parameter that

is used to determine the spline. This gives an additional degree of freedom for adjusting the strain response of the band structure.

Friedel Interpolation Formula for Local Pseudopotential

Friedel et al. [6] propose an analytic expression for the local pseudopotential of diamond-structure crystals with six free parameters a_1, \dots, a_6 :

$$V(q) = \frac{1}{\Omega_c \exp(a_3(q^2 - a_4)) + 1} \left(\tanh\left(\frac{a_5 - q^2}{a_6}\right) + 1 \right) \quad (99)$$

This allows interpolation of $V(q)$ at the three q values that feature in unstrained silicon and still leaves freedom for adjusting the derivatives $V'(q)$ and, therefore, the strain response.

Plane-Wave Normalization in Strained Crystals

When going from an unstrained crystal to a strained crystal, the volume of the unit cell changes by a factor $F := (\Omega_c^{\text{strained}} / \Omega_c^{\text{ref}}) = \det(1 + \underline{\varepsilon})$. The local pseudopotential function $V(q)$ of the unstrained crystal corresponds to matrix elements obtained from plane waves normalized on the unstrained cell volume $\langle \mathbf{r} | \mathbf{K} \rangle^{\text{(ref)}} = \exp(i\mathbf{K} \cdot \mathbf{r}) / \Omega_c^{\text{ref}}$.

In the strained crystal, plane waves must be normalized to the deformed unit cell volume $\Omega_c^{\text{strained}}$. Consequently, the local form-factor of the strained crystal is:

$$V^{\text{strained}}(q) = \frac{1}{F} V^{\text{unstrained}}(q) \quad (100)$$

No special treatment is needed for the kinetic energy and the nonlocal potential (both terms are directly computed on the deformed crystal). However, the spin-orbit coupling term needs some attention: In going from [Equation 84 on page 155](#) to the normalized expression of [Equation 96 on page 156](#), a factor of $1/\Omega_c$ was discarded. Depending on the details of the implementation of [Equation 96](#), the unit conversion of \mathbf{K} and \mathbf{K}' might introduce an additional factor proportional to $\Omega_c^{2/3}$. To recover the $1/\Omega_c$ scaling behavior of [Equation 84](#), the spin-orbit matrix elements in a strained crystal must be multiplied by $F^{-5/3}$.

Alloys and the Virtual Crystal Approximation

You can treat the band structure of statistical alloys such as $\text{Si}_{1-x}\text{Ge}_x$ by putting *virtual* $\text{Si}_{1-x}\text{Ge}_x$ atoms on the lattice sites of a diamond lattice with appropriate lattice spacing. These virtual atoms are statistical mixtures of the two constituent species involved; the atomic form-factor is simply the linear interpolation of the form factors of the constituent species.

As discussed in [Bulk Strain and Internal Strain on page 157](#), it is necessary to enforce correct normalization of the plane waves on a unit cell whose volume differs from that of the

Chapter 11: Empirical Pseudopotential Method

Strained Materials

reference crystals (pure silicon or pure germanium). In contrast to the situation in strained crystals where the volume ratio could be inferred from the strain tensor, correct plane-wave normalization for alloys requires storing the *reference volume* for each atomic species.

EPM Model Defaults

By default, Sentaurus Band Structure uses cubic spline interpolation for the local pseudopotential. Nonlocal potential contributions and spin-orbit coupling are activated.

[Table 17](#) summarizes the default parameters for Si and Ge.

Table 17 Default EPM parameters

Equation symbol	Sentaurus Band Structure name	Unit	Value		Parameter description
			Si	Ge	
a_0	latticeConstant	Å	5.43	5.65	Unstrained lattice constant
Ω_a	volume	Å^3	$a_0^3/8$	$a_0^3/8$	Volume per ion
N_{val}	nValence	1	4	4	Valence electrons per ion
$V_{\text{loc}}\left(\sqrt{3}\frac{2\pi}{a_0}\right)$	vloc(3)	Ry	-0.2241	-0.221	Local form factor at $\ \mathbf{q}\ ^2 = 3\left(\frac{2\pi}{a_0}\right)^2$
$V_{\text{loc}}\left(\sqrt{8}\frac{2\pi}{a_0}\right)$	vloc(8)	Ry	0.0520	0.019	Local form factor at $\ \mathbf{q}\ ^2 = 8\left(\frac{2\pi}{a_0}\right)^2$
$V_{\text{loc}}\left(\sqrt{11}\frac{2\pi}{a_0}\right)$	vloc(11)	Ry	0.0724	0.056	Local form factor at $\ \mathbf{q}\ ^2 = 11\left(\frac{2\pi}{a_0}\right)^2$
μ_{LS}	muls	Ry	0.00018	0.000965	Spin-orbit splitting strength
ζ_{LS}	zetaLS	$1/r_{\text{Bohr}}$	4.6	5.34	Radial scaling factor for analytic core wavefunctions
n_{LS}	nLS	1	2	3	Principal quantum number of the shell that dominates spin-orbit coupling
R_0	r0	Å	1.06	0.0	$l = 0$ nonlocal well radius

Chapter 11: Empirical Pseudopotential Method

Group Velocities and Effective Masses

Table 17 Default EPM parameters (Continued)

Equation symbol	Sentaurus Band Structure name	Unit	Value		Parameter description
			Si	Ge	
α_0	A0	Ry	0.03	0.0	$l = 0$ nonlocal well depth
β_0	B0	1	0.0	0.0	Energy-dependent correction to the $l = 0$ nonlocal well depth
R_2	R2	Å	0.0	1.22	$l = 2$ nonlocal well radius
α_2	A2	Ry	0.0	0.275	$l = 2$ nonlocal potential strength
–	nonLocalWell	–	Square	Gaussian	Nonlocal well shape
ζ	zeta	1	0.53	0.45	Internal strain parameter
–	Qcutoff	$2\pi/a_0$	$\sqrt{11.5}$	$\sqrt{12.44}$	Cut-off for the local pseudopotential form factor

Group Velocities and Effective Masses

Many applications (for example, Monte Carlo simulation) require the knowledge of group velocities:

$$\mathbf{v}_n(\mathbf{k}) = \nabla_{\mathbf{k}} \tilde{\epsilon}_n(\mathbf{k}) \quad (101)$$

Some applications also need the reciprocal effective-mass tensor:

$$\left[\frac{1}{m_n^*} \right]_{\alpha, \beta} = \pm \frac{1}{\hbar^2} \frac{\partial^2 \tilde{\epsilon}_n}{\partial k_\alpha \partial k_\beta} \quad (102)$$

The positive sign is used for electrons, the negative sign is used for holes. Obtaining these quantities by taking numeric derivatives of the band energies might encounter difficulties, for example, at crossing points of different bands. Therefore, the Hellman–Feynman theorem [8] is invoked to express the group velocities in terms of \mathbf{k} -derivatives of the Hamiltonian rather than derivatives of the band energies:

$$\mathbf{v}_n = \nabla_{\mathbf{k}} \langle n\mathbf{k}|H|n\mathbf{k} \rangle = \langle n\mathbf{k}|\nabla_{\mathbf{k}} H|n\mathbf{k} \rangle \quad (103)$$

Chapter 11: Empirical Pseudopotential Method

Group Velocities and Effective Masses

Sentaurus Band Structure uses numeric derivatives of the Hamiltonian matrix elements for evaluation of [Equation 103](#), which is considerably faster than evaluating the derivatives analytically.

For the evaluation of the effective masses, the degenerate perturbation theory is used. To obtain the effective-mass tensor at a given wavevector \mathbf{k}_0 , vector H is used (\mathbf{k}_0 is the unperturbed Hamiltonian H^0), and the \mathbf{k} -dependent Hamiltonian is expanded in a power series around \mathbf{k}_0 ; terms up to second order in the k_α are retained:

$$H = H^0 + \underbrace{\sum_{\alpha} \delta k_{\alpha} \frac{\partial H}{\partial k_{\alpha}}}_{=:H^1} + \underbrace{\sum_{\alpha, \beta} \delta k_{\alpha} \delta k_{\beta} \frac{\partial^2 H}{\partial k_{\alpha} \partial k_{\beta}}}_{=:H^{(2)}} \quad (104)$$

The first-order change in band energy leads again to [Equation 103](#). The second-order change in energy is related to the effective masses:

$$\delta E_i^{(2)} = \sum_{\alpha, \beta} \langle i^0 | H^{(2)} | i^0 \rangle + \langle i^0 | H^1 (1 - | i^0 \rangle \langle i^0 |) | i^1 \rangle \quad (105)$$

The first-order change in the wavefunction, $| i^1 \rangle$, is given by:

$$(E_i^0 - E_j^0) \langle i^0 | i^1 \rangle = \langle j^0 | H^1 | i^0 \rangle, \forall i \neq j \quad (106)$$

To avoid a contradiction (zero LHS but nonzero RHS in [Equation 106](#)), it is essential that the $| i^0 \rangle$ form a basis that diagonalizes H^1 on all eigenspaces of H . In the absence of degeneracy beyond the trivial degeneracy of Kramers pairs of a single-particle spin-1/2 Hamiltonian in the absence of an external magnetic field [9], this is automatically the case. Otherwise, H^1 must be diagonalized explicitly on each degenerate eigenspace of H .

Although Sentaurus Band Structure supports this explicit subspace diagonalization, this approach might result in nonintuitive results (for example, linear combinations of bands with large off-diagonal mass components). Therefore, the preferred approach for extracting the (reciprocal) effective-mass tensor in the presence of degeneracy consists of applying infinitesimal strain to break the degeneracy followed by calculating the nondegenerate effective mass.

Substituting [Equation 106](#) and [Equation 104](#) into [Equation 105](#) yields:

$$\delta E_i^{(2)} = \sum_{\alpha, \beta} \delta k_{\alpha} \delta k_{\beta} \left(\langle i^0 | \frac{\partial^2 H}{\partial k_{\alpha} \partial k_{\beta}} | i^0 \rangle + 2\Re \left(\sum_{j \neq i} \frac{\langle i^0 | \frac{\partial H}{\partial k_{\alpha}} | j^0 \rangle \langle j^0 | \frac{\partial H}{\partial k_{\beta}} | i^0 \rangle}{E_i^0 - E_j^0} \right) \right) \quad (107)$$

and the effective masses take the form:

$$\left[\frac{1}{m_i^*} \right]_{\alpha, \beta} = \frac{1}{\hbar^2} \sum_{\alpha, \beta} \left(\langle i^0 | \frac{\partial^2 H}{\partial k_{\alpha} \partial k_{\beta}} | i^0 \rangle + 2\Re \left(\sum_{j \neq i} \frac{\langle i^0 | \frac{\partial H}{\partial k_{\alpha}} | j^0 \rangle \langle j^0 | \frac{\partial H}{\partial k_{\beta}} | i^0 \rangle}{E_i^0 - E_j^0} \right) \right) \quad (108)$$

Chapter 11: Empirical Pseudopotential Method

References

If the potential is pure local, $\nabla_{\mathbf{k}} H$ becomes proportional to the momentum operator \mathbf{p} , and the second-order derivative of H reduces to:

$$\frac{\partial^2 H}{\partial k_\alpha \partial k_\beta} = \hbar^2 \delta_{\alpha, \beta} \quad (\text{local potential}) \quad (109)$$

Velocities are in units of $eV/ha_{x/y/z}$, where $a_{x/y/z}$ are the strained lattice constants according to the diagonal part of the strain tensor. Reciprocal effective masses are in units of $1/m_0$ where m_0 is the free electron mass.

References

- [1] M. M. Rieger and P. Vogl, "Electronic-band parameters in strained $\text{Si}_{1-x}\text{Ge}_x$ alloys on $\text{Si}_{1-y}\text{Ge}_y$ substrates," *Physical Review B*, vol. 48, no. 19, pp. 14276–14287, 1993.
- [2] J. R. Chelikowsky and M. L. Cohen, "Nonlocal pseudopotential calculations for the electronic structure of eleven diamond and zinc-blende semiconductors," *Physical Review B*, vol. 14, no. 2, pp. 556–582, 1976.
- [3] F. Herman and S. Skillman, *Atomic Structure Calculations*, Englewood Cliffs, New Jersey: Prentice-Hall, 1963.
- [4] J. P. Walter *et al.*, "Calculated and Measured Reflectivity of ZnTe and ZnSe," *Physical Review B*, vol. 1, no. 6, pp. 2661–2667, 1970.
- [5] L. Kleinman, "Deformation Potentials in Silicon. I. Uniaxial Strain," *Physical Review*, vol. 128, no. 6, pp. 2614–2621, 1962.
- [6] P. Friedel, M. S. Hybertsen, and M. Schlüter, "Local empirical pseudopotential approach to the optical properties of Si/Ge superlattices," *Physical Review B*, vol. 39, no. 11, pp. 7974–7977, 1989.
- [7] C. G. Van de Walle, "Strain effects on the valence-band structure of SiGe," *Properties of Silicon Germanium and SiGe:Carbon*, EMIS Datareviews Series, no. 24, E. Kaspar and K. Lyutovich (eds.), London: INSPEC, The Institution of Electrical Engineers, pp. 135–139, 2000.
- [8] R. P. Feynman, "Forces in Molecules," *Physical Review*, vol. 56, no. 4, pp. 340–343, 1939.
- [9] E. O. Kane, "Energy Band Structure in *p*-type Germanium and Silicon," *Journal of Physics and Chemistry of Solids*, vol. 1, no. 1-2, pp. 82–99, 1956.

12

Analytic Bands for Bulk Crystals

This chapter discusses analytic bands for bulk crystals.

Conduction Bands

The conduction band in silicon is usually approximated by three pairs of equivalent minima located close to the X-point of the Brillouin zone ($0.15\frac{2\pi}{a}$ away from the X-point in the direction of the Γ -point). It is commonly accepted that, close to the minima, electron dispersion is well described by the effective mass approximation with two masses: longitudinal mass $m_l = 0.914 m_0$ and transverse mass $m_t = 0.196 m_0$.

To describe a deviation of the density-of-states from pure parabolic expression, the nonparabolicity parameter α is introduced.

Note:

An applied strain modifies the crystal structure and, for a relatively small strain, the lattice constant can be defined for each crystal axis: $a_x = a(1 + \varepsilon_{xx})$, $a_y = a(1 + \varepsilon_{yy})$, and $a_z = a(1 + \varepsilon_{zz})$ where ε_{xx} , ε_{yy} , and ε_{zz} are diagonal components of the strain tensor. Therefore, components of the \mathbf{k} -vector are scaled by a corresponding component of the lattice constant.

Without shear strain (zero off-diagonal components of the strain tensor), changes in the conduction bands are trivial and are mostly defined by a constant strain-dependent energy shift for each band. However, the presence of shear strain could greatly disturb conduction bands, and it is described here.

Due to simplifications in band physics, the provided models describe conduction bands well only in a vicinity of the band minima of each six silicon valleys. Therefore, for the defined \mathbf{k} -vector, this analytic option selects one of the six valleys and computes bands for the selected valley only.

Two-Band $\mathbf{k} \cdot \mathbf{p}$ Model

As it is well described in the literature [1], the two-band $\mathbf{k} \cdot \mathbf{p}$ model considers the first conduction band $\Delta_1(i = 1)$ and the second conduction band $\Delta_2(i = 2)$ that are degenerate exactly at the X-point without stress.

Since the minimum of the conduction band is only $k_0 = 0.15\frac{2\pi}{a}$ away from the X-point, the dispersion around the minimum can be well described by the degenerate $\mathbf{k} \cdot \mathbf{p}$ perturbation theory, which only includes these two bands.

In Equation 110, only the pair of equivalent conduction-band valleys along the [001] direction is considered, but the electron dispersion of other valleys can be expressed similarly [1]:

$$E_i(\mathbf{k}) = \frac{\hbar^2 k_z^2}{2m_l} + \frac{\hbar^2(k_x^2 + k_y^2)}{2m_t} + (-1)^{i-1} \sqrt{\left(\frac{\hbar}{m_0} k_z p\right)^2 + \left(\Xi_s \varepsilon_{xy} - \frac{\hbar^2 k_x k_y}{M m_t}\right)^2} + \delta E_C \quad (110)$$

where:

- $\delta E_C = \Xi_d(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}) + \Xi_u \varepsilon_{zz}$ for the valley in the [001] direction.
- $i = 1, 2$ is the band index.
- k_z is negative with $k_z = 0$ at the X-point.
- M is the parameter of the $\mathbf{k} \cdot \mathbf{p}$ model related to coupling between bands.
- ε_{xy} is the shear-stress component positive for tensile stress in the [110] direction.
- Ξ_s is the shear deformation potential.
- Ξ_d is the dilation deformation potential.
- Ξ_u is the uniaxial deformation potential.
- The matrix element p is a dependent variable and is defined by a position of the band minima of the first band at $k_z = -k_0$ without the stress: $p = k_0 \hbar / m_l$.

Ellipsoidal Model

This model provides pure parabolic bands with a stress-induced change of effective masses and band minima position. Analytic derivation of these changes is performed in [1] and is based on an evaluation of Equation 110.

To simplify final expressions, the dimensionless off-diagonal strain is introduced:
 $\eta = 4\Xi_u \varepsilon_{xy} / \Delta_{bs}$ where $\Delta_{bs} = 0.53$ eV is an energy gap between the bands (Δ_1 and Δ_2) at $k_z = -k_0$ with no stress condition.

Chapter 12: Analytic Bands for Bulk Crystals

Conduction Bands

With stress, the first band minima position k_{\min}^1 moves along the k_z -axis in the direction of the X-point as follows:

$$k_{\min}^1 = \begin{cases} -k_0 \sqrt{1 - \eta^2}, & |\eta| \leq 1 \\ 0, & |\eta| > 1 \end{cases} \quad (111)$$

The band minima of the second band are reflected through the X-point ($k_{\min}^2 = -k_{\min}^1$) to have it similar to the no-stress condition.

The minima of the first band move down in energy with the following approximation:

$$\Delta E_{\text{shear}}^1 = \begin{cases} -\frac{\Delta}{4}\eta^2, & |\eta| \leq 1 \\ -\frac{\Delta}{4}(2|\eta| - 1), & |\eta| > 1 \end{cases} \quad (112)$$

The energy minima of the second band are estimated as $\Delta E_{\text{shear}}^2 = \Delta E_{\text{shear}}^1 + \eta\Delta$.

The effective masses m_l and m_t are modified with shear strain as well. Evaluating [Equation 110](#) at the band minima of the first band [1] gives two branches for the transverse mass across the m_{t1} direction and along the m_{t2} stress direction:

$$m_{t1}(\eta)/m_t = \begin{cases} \left(1 - \frac{\eta}{M}\right)^{-1}, & |\eta| \leq 1 \\ \left(1 - \frac{\text{sign}(\eta)}{M}\right)^{-1}, & |\eta| > 1 \end{cases} \quad (113)$$

$$m_{t2}(\eta)/m_t = \begin{cases} \left(1 + \frac{\eta}{M}\right)^{-1}, & |\eta| \leq 1 \\ \left(1 + \frac{\text{sign}(\eta)}{M}\right)^{-1}, & |\eta| > 1 \end{cases} \quad (114)$$

$$m_l(\eta)/m_l = \begin{cases} (1 - \eta^2)^{-1}, & |\eta| \leq 1 \\ (1 - 1/|\eta|)^{-1}, & |\eta| > 1 \end{cases} \quad (115)$$

Finally, the parabolic electron dispersion of two bands is expressed as:

$$E_i(\mathbf{k}) = \frac{\hbar^2(k_z - k_{\min}^i)^2}{2m_l(\eta)} + \frac{\hbar^2 k_{t1}^2}{2m_{t1}(\eta)} + \frac{\hbar^2 k_{t2}^2}{2m_{t2}(\eta)} + \Delta E_{\text{shear}}^i + \delta E_C \quad (116)$$

where k_{t1} and k_{t2} are components of vector \mathbf{k} in the directions (-110) and (110), respectively, for valleys along the (001) direction. For valleys of other directions, the electron dispersion is similar.

Nonparabolicity Model

Generally, nonparabolic electron dispersion $E^{np}(\mathbf{k})$ of a band with the effective mass m is introduced by $E^{np}(1 + \alpha E^{np}) = \hbar^2 \|\mathbf{k}\|^2 / 2m$, where α is the nonparabolicity parameter. Considering that the band minima energy (E_{\min}^i) differs for each of the Δ_1 and Δ_2' bands, the nonparabolic dispersion can be written as:

$$E_i^{np}(\mathbf{k}) = \frac{1}{2\alpha}(-1 + \sqrt{1 + 4(E_i(\mathbf{k}) - E_{\min}^i)\alpha}) + E_{\min}^i \quad (117)$$

where $E_i(\mathbf{k})$ is the electron dispersion defined by [Equation 110](#) or [Equation 116](#).

In the literature [1], it is derived that the nonparabolicity parameter α depends on the shear strain η , and this is accounted for in the ellipsoidal band model:

$$\alpha(\eta)/\alpha = \frac{1 + 2(\eta/M)^2}{1 - (\eta/M)^2} \quad (118)$$

Model Parameters

[Table 18](#) lists all parameters of the analytic conduction-band model with default values that can be modified. The nonparabolicity parameters `el_alpha` and `kp_alpha` take a list of two values: the first value is used for the first conduction band and the second value is used for the second conduction band.

Table 18 Model parameters

Symbol	Parameter name	Default value	Unit
m_l/m_0	<code>ml</code>	0.914	—
m_t/m_0	<code>mt</code>	0.196	—
α (ellipsoidal)	<code>el_alpha</code>	{0.5 0.5}	eV ⁻¹
α (k · p)	<code>kp_alpha</code>	{0.15 0.15}	eV ⁻¹
$(1 - k_0)a/2\pi$	<code>k0</code>	0.85	—
M	<code>M</code>	1.2	—
Ξ_d	<code>xi_d</code>	0.77	eV
Ξ_u	<code>xi_u</code>	9.17	eV

Chapter 12: Analytic Bands for Bulk Crystals

Valence Bands

Table 18 Model parameters (Continued)

Symbol	Parameter name	Default value	Unit
$\Xi_{u'}$	xi_s	7	eV
Δ_{bs}	dbs	0.53	eV

Valence Bands

The analytic band model for valence bands in diamond-type and zinc-blende-type semiconductors is based on the six-band $\mathbf{k} \cdot \mathbf{p}$ approach. This approach is computationally inexpensive, but it considers both the warped and nonparabolic nature of the valence bands as well as arbitrary strain. While this method can be applied for arbitrary \mathbf{k} in the Brillouin zone, this method is most accurate near the Γ -point and loses accuracy a few hundred meV away from the band edge.

Theory

The $\mathbf{k} \cdot \mathbf{p}$ method is a perturbative approach for solving the one-electron Schrödinger equation, typically around an extremum of the band structure [2]. Using the Bloch theorem, the wavefunction for a state in band n with wavevector \mathbf{k} can be written as:

$$\phi_{n\mathbf{k}} = u_{n\mathbf{k}}(\mathbf{r}) \exp(i\mathbf{k} \cdot \mathbf{r}) \quad (119)$$

where $u_{n\mathbf{k}}$ is the periodicity of the crystal lattice. Considering both spin-orbit coupling (H_{so}) and strain (H_ε), the Schrödinger equation for $u_{n\mathbf{k}}$ is:

$$\left(\frac{\|\mathbf{p}\|^2}{2m_0} + V + H_{so} + \frac{\hbar \mathbf{k} \cdot \mathbf{p}}{m_0} + \frac{\hbar^2 k^2}{2m_0} + H_\varepsilon \right) u_{n\mathbf{k}} = E_{n\mathbf{k}} u_{n\mathbf{k}} \quad (120)$$

At the Γ -point ($\mathbf{k} = \mathbf{0}$) in the absence of strain, this Hamiltonian reduces to the first three terms, and its solution for states near the band gap generates a set of six valence band states (Γ_v) and a set of conduction band states (Γ_c).

While the combined set of valence and conduction band states at Γ without strain can serve as a basis for solving Equation 120 for arbitrary \mathbf{k} and strain, a more convenient formulation can be obtained by treating the last three terms of the Hamiltonian using the Löwdin perturbation method [2]. In this approach, the six Γ_v states are treated directly, while their coupling to the Γ_c states is treated using first-order perturbation theory.

Within the Γ_v basis, the Hamiltonian is then reduced to a 6×6 matrix including a spin degeneracy of two. While the elements of this Hamiltonian matrix can be computed explicitly from perturbation theory, the resulting coefficients are typically taken as fitting parameters

Chapter 12: Analytic Bands for Bulk Crystals

Valence Bands

that can be extracted from experimental measurements or from more rigorous band-structure approaches such as the empirical pseudopotential method (EPM).

Luttinger–Kohn or Bir–Pikus Hamiltonian

In the Γ_γ basis of total angular momentum (Luttinger–Kohn basis) that diagonalizes the spin-orbit Hamiltonian at the Γ -point in the absence of strain, the 6×6 Luttinger–Kohn Hamiltonian augmented by the Bir–Pikus strain terms (in the crystallographic coordinate system) is given by [3] the following equation:

$ \frac{3}{2}, \frac{3}{2} \rangle$	$ \frac{3}{2}, \frac{1}{2} \rangle$	$ \frac{3}{2}, -\frac{1}{2} \rangle$	$ \frac{3}{2}, -\frac{3}{2} \rangle$	$ \frac{1}{2}, \frac{1}{2} \rangle$	$ \frac{1}{2}, -\frac{1}{2} \rangle$		
$H_{LK} = -$	$P + Q$	$-S$	R	0	$-\frac{S}{\sqrt{2}}$	$\sqrt{2}R$	$ \frac{3}{2}, \frac{3}{2} \rangle$
	$P - Q$	0	R	$-\sqrt{2}Q$	$\sqrt{\frac{3}{2}}S$		$ \frac{3}{2}, \frac{1}{2} \rangle$
		$P - Q$	S	$\sqrt{\frac{3}{2}}S^*$	$\sqrt{2}Q$		$ \frac{3}{2}, -\frac{1}{2} \rangle$
		$P + Q$	$-\sqrt{2}R^*$	$-\frac{S^*}{\sqrt{2}}$			$ \frac{3}{2}, -\frac{3}{2} \rangle$
		$P + \Delta$	0				$ \frac{1}{2}, \frac{1}{2} \rangle$
		$P + \Delta$					$ \frac{1}{2}, -\frac{1}{2} \rangle$

where:

$$\bullet \quad P = P_k + P_\varepsilon \quad Q = Q_k + Q_\varepsilon \quad R = R_k + R_\varepsilon \quad S = S_k + S_\varepsilon$$

Chapter 12: Analytic Bands for Bulk Crystals

Valence Bands

Luttinger–Kohn:

$$\begin{aligned}
 P_k &= \frac{\hbar^2}{2m_0} \gamma_1 (k_x^2 + k_y^2 + k_z^2) & \text{Bir–Pikus:} \\
 Q_k &= \frac{\hbar^2}{2m_0} \gamma_2 (k_x^2 + k_y^2 - 2k_z^2) & P_\epsilon = -a(\epsilon_{xx} + \epsilon_{yy} + \epsilon_{zz}) \\
 R_k &= \frac{\hbar^2}{2m_0} \sqrt{3} [-\gamma_2(k_x^2 - k_y^2) + 2i\gamma_3 k_x k_y] & Q_\epsilon = -\frac{b}{2}(\epsilon_{xx} + \epsilon_{yy} - 2\epsilon_{zz}) \\
 S_k &= \frac{\hbar^2}{2m_0} 2\sqrt{3}\gamma_3(k_x - ik_y)k_z & R_\epsilon = \frac{\sqrt{3}}{2}b(\epsilon_{xx} - \epsilon_{yy}) - id\epsilon_{xy} \\
 & & S_\epsilon = -d(\epsilon_{zx} - i\epsilon_{yz})
 \end{aligned}$$

- $\gamma_1, \gamma_2, \gamma_3$ are the Luttinger parameters.
- a, b, d are the Bir–Pikus deformation potentials.
- Δ is the spin-orbit split-off energy.

The free electron mass is given by m_0 , and ϵ_{ij} is the i, j -th element of the physical strain tensor. Only the upper-right part of H_{LK} is given since the matrix is Hermitian. The minus sign in front of the matrix indicates that the hole energies grow negative away from Γ .

The states written as $|j, m\rangle$ are states with total angular momentum j and z -projected momentum m . In the absence of strain, the states $|\frac{3}{2}, \pm\frac{3}{2}\rangle$, $|\frac{3}{2}, \pm\frac{1}{2}\rangle$, and $|\frac{1}{2}, \pm\frac{1}{2}\rangle$ represent the heavy holes, light holes, and split-off holes, respectively. In the presence of strain, these states become mixed and this simple identification cannot be made.

Model Parameters

[Table 19](#) lists the user-adjustable parameters for the six-band $k \cdot p$ model for Si and Ge. The default values were extracted from Sentaurus Band Structure EPM calculations (exception: values for the a deformation potential are taken from Van de Walle [4]).

Table 19 Parameters of six-band $k \cdot p$ model

Symbol	Parameter name	Si default value	Ge default value	Unit
γ_1	gamma1	4.306	10.536	1
γ_2	gamma2	0.345	3.107	1
γ_3	gamma3	1.44	4.397	1

Chapter 12: Analytic Bands for Bulk Crystals

References

Table 19 Parameters of six-band $k \cdot p$ model (Continued)

Symbol	Parameter name	Si default value	Ge default value	Unit
Δ_{so}	Delta	0.0434	0.297	eV
a	a_v	2.46	1.25	eV
b	b	-2.316	-2.067	eV
d	d	-5.514	-3.836	eV

References

- [1] V. Sverdlov *et al.*, “Effects of Shear Strain on the Conduction Band in Silicon: An Efficient Two-Band $k \cdot p$ Theory,” in *Proceedings of the 37th European Solid-State Device Research Conference (ESSDERC)*, Munich, Germany, pp. 386–389, September 2007.
- [2] P. Y. Yu and M. Cardona, *Fundamentals of Semiconductors: Physics and Materials Properties*, Berlin: Springer, 3rd ed., 2005.
- [3] C. Y.-P. Chao and S. L. Chuang, “Spin-orbit-coupling effects on the valence-band structure of strained semiconductor quantum wells,” *Physical Review B*, vol. 46, no. 7, pp. 4110–4122, 1992.
- [4] C. G. Van de Walle, “Strain effects on the valence-band structure of SiGe,” *Properties of Silicon Germanium and SiGe:Carbon*, EMIS Datareviews Series, no. 24, E. Kaspar and K. Lyutovich (eds.), London: INSPEC, The Institution of Electrical Engineers, pp. 135–139, 2000.

13

Subband and Mobility Calculations

This chapter describes the features and physical models for computing subband structure and inversion mobility.

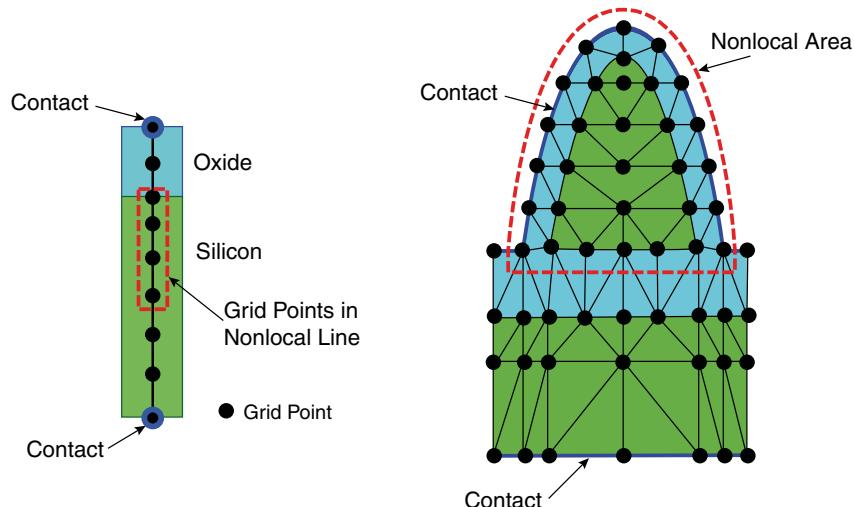
Device Structure

Subband and mobility calculations are performed on device structures that are typically MOS capacitors. These structures are read from a TDR file that can be created, for example, by Sentaurus Process.

Regions and Materials

As shown in [Figure 42](#), the different materials of a device structure are represented as regions. Each region has one material and there can be multiple regions of the same material. The materials silicon, polysilicon, and oxide (SiO_2) are supported by default using the names Silicon, Polysilicon, and Oxide, respectively.

Figure 42 (Left) One-dimensional device structure and (right) 2D device structure



Chapter 13: Subband and Mobility Calculations

Device Structure

Each material has a material type that is either semiconductor or insulator. You can introduce new materials into Sentaurus Band Structure by using the `Material` command (see [Material on page 361](#)). Each region has a unique name that can be used to reference the region during model specification. Each region consists of a set of points on which various models are defined, such as the vacuum potential and the electron density.

Contacts

Contacts are used to apply bias to a device and are read automatically from the TDR file. The workfunction of a contact can be modified by using the `Physics` command (see [Physics for Contacts on page 367](#)).

Nonlocal Lines and Nonlocal Areas

The calculation of the confined carrier density by the solution to the Schrödinger equation and the calculation of the inversion mobility are confined to *nonlocal lines* for 1D devices and *nonlocal areas* for 2D devices.

A nonlocal line or nonlocal area represents a part of the device structure. The geometry of a nonlocal line or nonlocal area is defined by using the `Math` command (see [Math on page 362](#)). For each nonlocal line or nonlocal area, one Schrödinger solver can be associated using the `Physics` command. In addition, one mobility calculator can be associated with a nonlocal line or nonlocal area by using the `Physics` command (see [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#)).

Device Coordinate Axes

The convention used to define the device coordinate axes depends on the dimensionality of the device structure.

Axes for 1D Devices

As shown in [Figure 43](#), the 1D device direction is along the z-axis. The z-axis also defines the surface orientation. This direction relative to the crystallographic coordinate system can be specified with the `surfaceOrientation` argument of the `Physics` command (see [Physics for Top-Level Parameters on page 365](#)).

For mobility calculations for 1D devices, the in-plane device axes are given by the x- and y-axes. The x-axis relative to the crystallographic coordinate system can be specified using the `xDirection` argument of the `Physics` command. The specified `surfaceOrientation` and `xDirection` must be orthogonal.

Chapter 13: Subband and Mobility Calculations

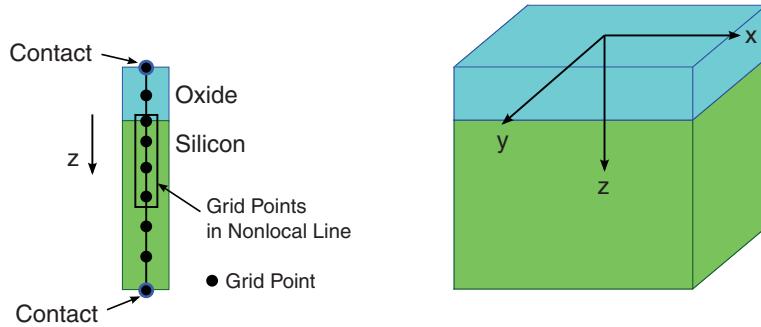
Device Structure

The y-axis is then formed from the cross product of the z- and x-axes. [Table 20](#) lists some common `surfaceOrientation` and `xDirection` combinations.

Table 20 Common surface orientation and x-axis directions

Surface orientation	surfaceOrientation	xDirection
(100)	[list 0 0 1]	[list 1 0 0]
(110)	[list 0 1 1]	[list 1 0 0]
(111)	[list 1 1 1]	[list 1 1 -2]

Figure 43 Axes for 1D devices



Axes for 2D Devices

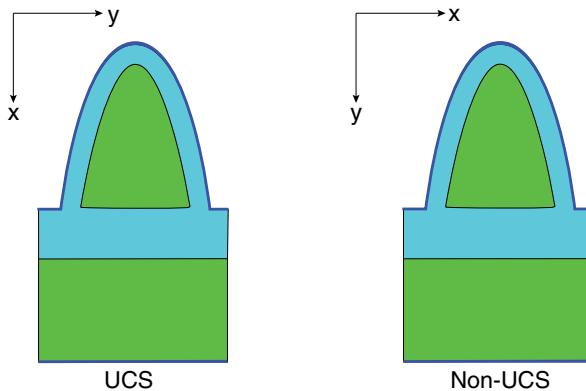
For 2D devices, the axes of the device are always labeled as *x* and *y*. As shown in [Figure 44](#), two different conventions are used depending on whether the unified coordinate system (UCS) is used. For UCS structures, the horizontal axis is the *y*-axis and the vertical axis is the *x*-axis. For non-UCS structures, the axis labels are exchanged.

The coordinate system used, that is, UCS or non-UCS, is read from the TDR file. The direction of the *x*-axis and *y*-axis, relative to the crystallographic coordinate system, can be specified using the `xDirection` and `yDirection` arguments of the `Physics` command (see [Physics for Top-Level Parameters on page 365](#)).

Chapter 13: Subband and Mobility Calculations

Files Used by Subband and Mobility Calculations

Figure 44 Axes for 2D devices



Axes for Wurtzite Crystal Semiconductors

For devices that consist of wurtzite semiconductor materials, you can specify the direction relative to the crystallographic coordinate system using the hexagonal Bravais–Miller indexing system, which is a four-index system.

Note:

The rotation of the device axes retains the same dependence on the dimensionality of the device structure as discussed [Axes for 2D Devices on page 175](#).

To simplify the orientation definition of the device axes with the Bravais–Miller indexing system, any two of the three device axes can be specified regardless of the device dimensionality. The two specified axes, however, must be orthogonal. Internally, the correct axes for the dimensionality of the device structure are determined automatically.

The Bravais–Miller indices can be specified using any two of the `wurtziteXOrientation`, `wurtziteYOrientation`, and `wurtziteZOrientation` arguments of the `Physics` command (see [Physics for Top-Level Parameters on page 365](#)).

Note:

The default orientation of the Cartesian axes within the hexagonal axes is such that the a_2 -axis is aligned with the y -axis, and the c -axis is aligned with the z -axis.

Files Used by Subband and Mobility Calculations

Sentaurus Band Structure uses several files to compute subband structure and mobility. With the name of the command file given by `<base>.cmd`, the various files are described here.

Chapter 13: Subband and Mobility Calculations

Files Used by Subband and Mobility Calculations

Input Files

The input files for Sentaurus Band Structure are:

- Command file (`<base>.cmd`): This is the main command (or input) file. It consists of the commands described in this manual for Sentaurus Band Structure along with intrinsic Tcl commands.
- Default physics file (`defaultPhysics.tcl`): This Tcl script sets the default models and parameters. It is read automatically during the `LoadDevice` command and is loaded from the `SBAND_LIB` directory.
- Default material definitions (`materialDB_util.tcl`): This Tcl script loads the user interface commands for the material database and initializes the default set of materials. This file is read automatically at the start of the Sentaurus Band Structure session.
- Input device structure (`*.tdx`): This file contains device structure and doping information.

Output Files

The output files for Sentaurus Band Structure are:

- Output log file (`<base>.log`): This file contains a log of the commands and additional information generated during the simulation.
- Bias log file (`<base>.plt`): This file contains the contact voltage biases and various simulation results after each call to the `Solve` command and each call to the `AddToFile` command. The default file name is based on the command file name. For example, if the command file is `mobility.cmd`, by default the bias log file is named `mobility.plt`. To change the name of the active bias log file, use the `logFile` argument of the `Solve` command (see [Solve on page 386](#)).

Table 21 Quantities saved to the bias log file by default

Quantity	Description	Unit
<code>V(<contact>)</code>	Applied bias on contact	V
<code>Q(<contact>)</code>	Charge on contact	1D: C/ μm^2 , 2D: C/ μm
<code>Ninv_in_well0</code>	Integrated eDensity in semiconductor well 0	1D: cm^{-2} , 2D: cm^{-1}
<code>Pinv_in_well0</code>	Integrated hDensity in semiconductor well 0	1D: cm^{-2} , 2D: cm^{-1}

Chapter 13: Subband and Mobility Calculations

Poisson Equation

Table 21 Quantities saved to the bias log file by default (Continued)

Quantity	Description	Unit
NewtonIterations	Number of Newton iterations for solving Poisson equation	1

- TDR file of real-space models (*.tdr): The `Save` command saves a TDR file with user-specified models as a function of the z-coordinate for 1D structures and the xy coordinates for 2D structures. The file name is user specified.
- TDR file of k -space models (*.tdr): The `SaveK` command saves a TDR file with selected models over 2D k -space for 1D device structures and 1D k -space for 2D device structures. The file name is user specified.

Poisson Equation

For a self-consistent solution, the Poisson equation must be solved in conjunction with models for the electron and hole carrier densities:

$$\nabla \cdot (\epsilon \nabla \psi_{\text{vac}}) = -e(N_D^+ - N_A^- + p - n) - \sigma_i \cdot \delta(s - s_0) + q_{\text{polar}} \cdot \delta(s - s_0) \quad (121)$$

where the solution variable, ψ_{vac} , is the vacuum potential. The various charge densities on the right-hand side (RHS) are the net active doping concentration $N_D^+ - N_A^-$, the hole density p , the electron density n , a representative interface charge density σ_i , and the polarization-induced charge q_{polar} .

The dielectric permittivity is given by ϵ , and the absolute value of the charge on an electron is denoted by e . The various quantities in the Poisson equation that can be accessed for visualization and extraction using model keywords are listed in [Table 22](#). The specification of charges at interfaces is described in [Interface Charge on page 195](#). The specification of polarization-induced charges at interfaces is described in [Polarization-Induced Charge on page 200](#).

Table 22 Model keywords for quantities in the Poisson equation

Model keyword	Unit	Description
DopingConcentration	cm^{-3}	Net active doping concentration
eDensity	cm^{-3}	Electron density
hDensity	cm^{-3}	Hole density

Chapter 13: Subband and Mobility Calculations

Poisson Equation

Table 22 Model keywords for quantities in the Poisson equation (Continued)

Model keyword	Unit	Description
Permittivity	1	Dielectric permittivity relative to the vacuum permittivity
VacuumPotential	V	Vacuum potential

Boundary Conditions

At an insulator–contact boundary, a Dirichlet boundary condition is used:

$$\psi_{\text{vac}} = V_{\text{app}} - \Phi_M \quad (122)$$

where V_{app} is the applied bias at the contact and Φ_M is the contact workfunction that can be set using the `workfunction` argument of the `Physics` command (see [Physics for Contacts on page 367](#)). The workfunction value is in units of eV.

At a semiconductor–contact boundary, a charge-neutrality condition is enforced:

$$(N_D^+ - N_A^- + p - n) = 0 \quad (123)$$

The vacuum potential is solved in all device regions and is continuous across all internal device boundaries.

Electric Field

The electric field is computed from the vacuum potential:

$$\vec{E} = -\nabla\psi_{\text{vac}} \quad (124)$$

The electric field is computed in all materials and can be accessed for visualization or extraction purposes by the model keyword `ElectricField`.

Convergence

The Poisson equation is a nonlinear equation that is solved using the Newton method. Solving the Poisson equation with carrier densities computed using one of the Schrödinger equations can lead to degraded convergence.

Several arguments in the `Math` command can improve and handle convergence issues (see [Math on page 362](#)). In particular, you can use `iterations` to set the maximum number of Newton iterations to allow for convergence. It is recommended to set this argument to a relatively large value, approximately 30.

Chapter 13: Subband and Mobility Calculations

Valley Models

Criteria for convergence can be set by `potentialUpdateTolerance` and `residualTolerance`. They set the criteria that must be met for convergence on the infinity norm of the potential update and the residual of the Poisson equation. Both criteria must be met for convergence.

To prevent the iterative solution from changing too much in one Newton iteration, you can use `potentialUpdateClamp` to set the maximum-allowed change in the potential.

In some cases, convergence can be difficult to achieve. In these cases, you can use an algorithm that adaptively dampens the potential updates to improve convergence. To activate this algorithm, set `damping=1`.

Furthermore, by setting `maxInnerIter` to a value greater than 1 (default is 1), the predictor loop is activated. This can also help to improve convergence, especially if the simulation is started with a poor initial guess.

The behavior of Sentaurus Band Structure when the Poisson equation fails to converge can be set by using `doOnFailure`. By default, the tool generates a Tcl error that can be treated using the Tcl `catch` command in the command file. If the error is not handled in this way, then the tool stops. This behavior can be suppressed by setting `doOnFailure=0`, in which case, no error is generated and the simulation continues to the next command without stopping.

Valley Models

The conduction and valence bands in most semiconductors are composed of multiple valleys or multiple bands or both. For example, in silicon, the conduction band is composed of states from three pairs of Δ -valleys, while the valence band is composed of three bands at the Γ -point. A single valley is represented using a particular valley model (`ValleyModel`). A valley model can represent multiple bands. Valley models serve two main purposes:

- To hold or compute model parameters that are then used by a Schrödinger solver or a bulk density model
- To automatically generate a set of bulk density and band-edge models

Valley models are multimodels, that is, more than one valley model can be defined for each region. To be able to refer to a particular `ValleyModel`, you must specify a name for the `ValleyModel`.

Common Syntax

All valley models share a common syntax, as shown in this example:

```
Physics material=Silicon ValleyModel=Model16 name=Delta1 \
degeneracy=2 useForEBulkDensity=1 <paraName1>=<value1> ...
```

The `ValleyModel` argument specifies a particular model. A name should be associated with this model so that the valley can be later referred to by other models. The valley degeneracy must also be specified by `degeneracy`. Each particular model will have a set of parameters that can also be set.

Automatic Generation of Bulk Density and Band-Edge Models

One reason to create a valley model is to provide parameters for computing the bulk carrier density for the valley. Specifying `useForEBulkDensity` or `useForHBulkDensity` causes a corresponding bulk carrier-density model to be created for computing the bulk carrier density for the valley. In addition, a model for the band edge of the valley is created automatically for visualization purposes.

For valley models that model multiple bands, a bulk density model and band-edge model for each band is created automatically. For example, using the previous `ValleyModel` example with the name of `Delta1`, the following models are created automatically: `Delta1_Density` and `Delta1_BandEdge`. These can be accessed for visualization or extraction purposes.

Ellipsoidal Valleys

Some valleys, such as the Δ -valleys in silicon, can be modeled as ellipsoids. The following models are available:

- [ConstantEllipsoid Valley Model](#)
- [2kpEllipsoid Valley Model](#)

ConstantEllipsoid Valley Model

The valley model `ConstantEllipsoid` models an ellipsoidal valley in which the effective masses and band-edge shifts of the valley are constant. The three principal axes and masses can be specified independently. When used for the parabolic Schrödinger equation, the nonparabolicity for in-plane dispersion can be specified using the `alpha` argument; while the nonparabolicity for quantization can be specified using the `alphaZ` argument.

This model can be specified using (see [Physics for Valley Models on page 369](#)):

```
Physics material=Silicon ValleyModel=ConstantEllipsoid name=Delta1 \
degeneracy=2 kl=[list 1 0 0] kt1=[list 0 1 0] kt2=[list 0 0 1] \
ml=0.9 mt1=.19 mt2=.19 alpha=0.0 Eshift=0.0 useForEBulkDensity=1
```

The three principal axes of the ellipsoid are specified in crystallographic coordinates by using `kl`, `kt1`, and `kt2` and their corresponding effective masses. Specifying `useForEBulkDensity=1` means an `eEllipsoidalDensity` model is created automatically.

Chapter 13: Subband and Mobility Calculations

Valley Models

This model uses the parameters of the `ValleyModel`, except for `alpha`, to compute the bulk density in the valley.

Table 23 Parameters and default values for ConstantEllipsoid valley model

Parameter	Value in all materials	Unit	Description
<code>a0</code>	<code>5.43e-8</code>	cm	Relaxed lattice constant
<code>alpha</code>	<code>0.0</code>	eV^{-1}	Nonparabolicity parameter
<code>alphaz</code>	<code>0.0</code>	eV^{-1}	Nonparabolicity parameter for quantization
<code>Eshift</code>	<code>0.0</code>	eV	Energy shift relative to relaxed band edge
<code>k1</code>	<code>{1 0 0}</code>	1	Longitudinal principal axis
<code>kt1</code>	<code>{0 1 0}</code>	1	First transverse principal axis
<code>kt2</code>	<code>{0 0 1}</code>	1	Second principal axis
<code>ml</code>	<code>1.0</code>	1	Longitudinal mass
<code>mt1</code>	<code>1.0</code>	1	First transverse mass
<code>mt2</code>	<code>1.0</code>	1	Second transverse mass

2kpEllipsoid Valley Model

This valley model provides a more accurate, strain-dependent, ellipsoidal model for Δ -valleys in silicon. This model treats the effective masses and band-edge shifts as a function of strain using the [Ellipsoidal Model on page 166](#). This model is based on two-band $k \cdot p$ theory and can be specified by using a `Physics` command of the form:

```
Physics material=Silicon ValleyModel=2kpEllipsoid name=Delta1 \
degeneracy=2 longAxis=100 useForEBulkDensity=1
```

Specifying `useForEBulkDensity=1` means an `eEllipsoidalDensity` model is created automatically. This model uses the model parameters of the `ValleyModel`, except for `alpha` and `alphaz`, to compute the bulk density in the valley. Note that this `ValleyModel` represents only one ellipsoidal valley. In silicon, for example, three ellipsoidal valleys should be specified, each with a degeneracy of two. To indicate the crystallographic axis along which the valley lies, you must use the `longAxis` argument.

Table 24 Parameters and default values for 2kpEllipsoid valley model

Parameter	Value in all materials	Unit	Description
a0	5.43e-8	cm	Relaxed lattice constant
alpha	0.5	eV ⁻¹	Nonparabolicity parameter
alphaz	0.0	eV ⁻¹	Nonparabolicity parameter for quantization
dbs	0.53	eV	Energy gap at Δ between first and second bands
Eshift	0.0	eV	Energy shift relative to relaxed band edge
k0	0.15	1	Location of relaxed band minimum from the X-point
longAxis	100	1	Indicates along which crystal axis the longitudinal axis of this valley is aligned; options are 100, 010, and 001
M	1.2	1	Coupling parameter
ml	0.916	1	Longitudinal mass
mt	0.194	1	Transverse mass
xi_d	0.77	eV	Linear deformation potential at the Δ -point
xi_s	7.0	eV	Linear deformation potential at the X-point
xi_u	9.17	eV	Linear deformation potential at the Δ -point

Valleys Based on k·p Theory

Models based on k·p theory allow for a more detailed treatment of band dispersion including warping and anisotropic nonparabolicity. The following models based on this approach are available:

- [The 6kpValley Model](#)
- [The 3kpValley Model](#)
- [The 8kpValley Model](#)
- [The 2kpValley Model](#)

The 6kpValley Model

The hole band structure in most semiconductors is very warped but can be modeled analytically using six-band $k \cdot p$, or Luttinger–Kohn, theory. The valley model `6kpValley` is based on the Luttinger–Kohn model (see [Luttinger–Kohn or Bir–Pikus Hamiltonian on page 170](#)) and represents three bands at the Γ -point.

This model is controlled by the following:

- Luttinger–Kohn parameters (`gamma1`, `gamma2`, and `gamma3`) determine the band dispersion
- Spin-orbit split-off energy (`delta`)
- Deformation potentials (`a_v`, `b`, and `d`) determine the strain response

This model can be created using a `Physics` command of the form:

```
Physics material=Silicon ValleyModel=6kpValley name=Gamma \
degeneracy=1 useForHBulkDensity=1
```

Specifying `useForHBulkDensity=1` means a `6kpDensity` model is created automatically for each of the three bands.

Table 25 Parameters and default values for 6kpValley valley model

Parameter	Value in all materials	Unit	Description
<code>a_v</code>	2.1	eV	Absolute deformation potential
<code>a0</code>	5.43e-8	cm	Relaxed lattice constant
<code>b</code>	-2.33	eV	Deformation potential
<code>d</code>	-4.75	eV	Deformation potential
<code>delta</code>	0.044	eV	Spin-orbit split-off energy
<code>gamma1</code>	4.27	1	Luttinger–Kohn parameter
<code>gamma2</code>	0.315	1	Luttinger–Kohn parameter
<code>gamma3</code>	1.387	1	Luttinger–Kohn parameter

The 3kpValley Model

The three-band $k \cdot p$ valley model is a simplified version of the six-band $k \cdot p$ valley model. In the case of weak spin-orbit interaction, the resulting (subband) dispersions are very similar and, in the special case of no spin-orbit interaction (`delta=0.0`), the results are identical.

Chapter 13: Subband and Mobility Calculations

Valley Models

Using the `3kpValley` model decreases the computational burden significantly, since spin degeneracy is implicitly taken into account and, therefore, the size of the Hamiltonian is reduced.

The `3kpValley` model uses the same parameters as the `6kpValley` model, but not `delta` (see [Table 25](#)).

The 8kpValley Model

The eight-band $k \cdot p$ valley model describes the lowest conduction band and three valence bands around the Γ -point. It is a suitable band structure model for direct bandgap materials.

This model can be created using a `Physics` command of the form:

```
Physics material=GaAs ValleyModel=8kpValley name=Gamma degeneracy=1 \
useForHBulkDensity=0
```

The `8kpValley` model uses the same parameters as the `6kpValley` model, but also has the additional parameters `a_c`, `E_p`, and `m_c` (see [Table 26](#)).

The renormalized inverse effective mass of the conduction band is denoted by A_c . Since its value is often not given in the literature, it is computed internally using the following convention [1]:

$$A_c := \frac{1}{m_c} - \frac{2E_p}{3E_g} - \frac{E_p}{3(E_g + \Delta_{so})} \quad (125)$$

The valence band parameters are renormalized internally as well [2] (see [Equation 183 on page 220](#) for details).

Table 26 Parameters and default values for 8kpValley valley model

Parameter	Value in all materials	Unit	Description
<code>a_c</code>	-10.62	eV	Deformation potential of the conduction band
<code>a_v</code>	-0.85	eV	Deformation potential
<code>a0</code>	5.65e-8	cm	Relaxed lattice constant
<code>b</code>	-1.85	eV	Deformation potential
<code>d</code>	-5.10	eV	Deformation potential
<code>delta</code>	0.341	eV	Spin-orbit split-off energy
<code>E_p</code>	23.81	eV	Optical matrix parameter
<code>gamma1</code>	7.05	1	Luttinger–Kohn parameter

Table 26 Parameters and default values for 8kpValley valley model (Continued)

Parameter	Value in all materials	Unit	Description
gamma2	2.35	1	Luttinger–Kohn parameter
gamma3	3.0	1	Luttinger–Kohn parameter
mc	0.067	1	Bulk effective mass of the conduction band

Spurious Solutions

The 8kpValley model is prone to spurious solutions, but only if certain rules for its input parameters are violated. Therefore, when the 8kpValley model is used in a Schrödinger equation as explained in [Using the Eight-Band \$k \cdot p\$ Schrödinger Solver on page 222](#), the user-defined input parameters are checked internally for consistency. If there is a violation, then a warning message is printed to screen.

A detailed discussion of spurious solutions in the 8kpValley model can be found in the literature [\[1\]\[3\]](#). In most cases, an artificial reduction of E_D must be considered to circumvent spurious solutions.

The 2kpValley Model

This model is used to model the Δ -valleys in the silicon conduction band using a warped, nonparabolic, strain-dependent dispersion, going beyond the simple ellipsoid approximation (see [Two-Band \$k \cdot p\$ Model on page 166](#)).

Note:

Only the 2kp Schrödinger solver can utilize this valley model.

The argument `valleyDir` of the `Physics` command is the identifier for the three different Δ -valleys in k -space. Valid input values are $\{0, 1, 2\}$.

See [Using the Two-Band \$k \cdot p\$ Schrödinger Solver on page 219](#) for an example of the proper definition of the 2kpValley model.

Table 27 Parameters and default values for 2kpValley valley model

Parameter	Value in all materials	Unit	Description
a0	5.43e-8	cm	Relaxed lattice constant
delta_Ec	0.128	eV	Difference between the relaxed band minimum and the conduction band energy at the X-point

Chapter 13: Subband and Mobility Calculations

Valley Models

Table 27 Parameters and default values for 2kpValley valley model (Continued)

Parameter	Value in all materials	Unit	Description
k_0	0.15	$2\pi/a_0$	Location of relaxed band minimum from the X-point
M	1.2	1	Coupling parameter
m_l	0.916	1	Longitudinal mass
m_t	0.194	1	Transverse mass
χ_i_d	0.77	eV	Dilation deformation potential
χ_i_s	7.0	eV	Shear deformation potential
χ_i_u	9.17	eV	Uniaxial deformation potential

Default Valley Models in Silicon

Table 28 lists the set of valley models and corresponding bulk density and band-edge models that are created by default for all silicon and polysilicon regions.

Table 28 Default valley models for silicon

Name	ValleyModel	Description
Delta1	2kpEllipsoid	Longitudinal direction along [100], $m_l=0.914$, $m_t=0.196$
Delta2	2kpEllipsoid	Longitudinal direction along [010], $m_l=0.914$, $m_t=0.196$
Delta3	2kpEllipsoid	Longitudinal direction along [001], $m_l=0.914$, $m_t=0.196$
Gamma	6kpValley	Three bands at gamma, default parameters from Table 25

Chapter 13: Subband and Mobility Calculations

Electrostatic Models

Electrostatic Models

This section describes the electrostatic models used in the solution of the Poisson equation.

Permittivity

The dielectric permittivity is modeled as a constant value relative to the vacuum permittivity and is set using:

```
Physics material=Silicon Permittivity epsilon=11.7
```

Table 29 Default permittivity values

Model keyword	Parameter	Silicon	Polysilicon	Oxide
Permittivity	epsilon	11.7	11.7	3.9

Relaxed Band Models

Band models are used in the calculation of the carrier densities by either bulk models or the solution of a Schrödinger equation. The specification and calculation of the needed band edges for these models have two parts: the specification of the relaxed electron affinity, χ , and the band gap, E_g , and then strain-dependent shifts computed by various valley models or by strain-dependent Schrödinger equations.

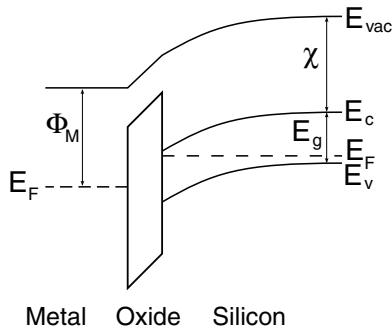
The relaxed affinity and band gap are modeled as constant values over a region and are independent of temperature and doping. The specification of the parameters for these models can be performed using:

```
Physics material=Silicon Affinity chi=4.5  
Physics material=Silicon Bandgap Eg=1.24
```

Table 30 Parameters and default values for relaxed band models

Model keyword	Parameter	Unit	Silicon	Polysilicon	Oxide
Affinity	chi	eV	4.0727	4.0727	0.90
Bandgap	Eg	eV	1.242	1.242	9.0

Figure 45 Band diagram for a bulk MOS capacitor



The relaxed conduction-band and valence-band energies can be computed directly from the vacuum potential, the relaxed affinity, and the band gap as indicated in [Table 31](#) and [Figure 45](#).

Table 31 Default valley models for silicon

Model keyword	Equation	Unit
ConductionBandEnergy	$-(\psi_{vac} + \chi)$	eV
ValenceBandEnergy	$-(\psi_{vac} + \chi + E_g)$	eV

Quasi-Fermi Levels

In addition to the band models, the quasi-Fermi levels are used in the calculation of the carrier densities. The quasi-Fermi levels are defined only for semiconductor regions and are modeled as fixed values tied to a particular contact value. The contact that is used to set the value of the quasi-Fermi level in a particular region is determined by looking at the connectivity of the region to the contact. The contact that is connected to a particular semiconductor region by other semiconductor regions is used to set the value of the quasi-Fermi level in that region.

If a semiconductor region is floating, in that there is no semiconducting path from a contact to that region, then the quasi-Fermi level is set to 0. This case often occurs in double-gate-type structures. [Table 32](#) lists the different models related to quasi-Fermi levels. The models for energy are simply the negative of the models for potential. These models are used internally by Sentaurus Band Structure and can be saved for visualization purposes.

Chapter 13: Subband and Mobility Calculations

Electrostatic Models

Table 32 Model keywords for models related to quasi-Fermi levels

Model keyword	Units	Description
eQuasiFermiEnergy	eV	Electron quasi-Fermi energy
hQuasiFermiEnergy	eV	Hole quasi-Fermi energy
eQuasiFermiPotential	V	Electron quasi-Fermi potential
hQuasiFermiPotential	V	Hole quasi-Fermi potential

Carrier Density

Several models are provided for computing the carrier densities for electrons and holes. Distinctions are made between models that treat confined carriers and those that model bulk densities, as well as between models that treat strain or do not treat strain.

Note:

Carrier densities are computed only in semiconductor regions.

Bulk Versus Confined Carrier Density

When a Schrödinger equation is defined for a carrier over a segment of the device, as determined by a nonlocal line, the carrier density over the nonlocal line is computed by integrating the subband dispersion as described in [Calculating the Confined Carrier Density on page 205](#). Outside of the nonlocal line, or for a carrier not treated by a Schrödinger equation, the carrier density is computed by a bulk density model.

The automatic treatment of the carrier density as confined or bulk is handled by the carrier density models with the names `eHybridDensity` and `hHybridDensity`. These models switch automatically from a confined density calculation to a bulk density model outside of the nonlocal line.

The density model that is used for the bulk calculation is set using `eBulkDensity` and `hBulkDensity`. For example:

```
Physics material=Silicon eDensity=eHybridDensity  
Physics material=Silicon eBulkDensity=eFermiDensity
```

This example sets the primary model for computing the electron density to `eHybridDensity`, which automatically applies the confined calculation to all nodes on the nonlocal line. Outside of the nonlocal line, `eHybridDensity` automatically applies the bulk density model called `eFermiDensity`.

Chapter 13: Subband and Mobility Calculations

Electrostatic Models

Due to the complicated nature of electron and hole bands under strain, several model choices for the bulk density are available.

Fermi–Dirac Bulk Density Model

Assuming parabolic dispersion within a set of valleys or bands with the same band extrema, the carrier density for electrons and holes under a Fermi–Dirac distribution function is given by:

$$\begin{aligned} n &= N_C \left(\frac{T}{300\text{ K}} \right)^{3/2} F_{1/2} \left(\frac{E_{Fn} - E_C}{k_B T} \right) \\ p &= N_V \left(\frac{T}{300\text{ K}} \right)^{3/2} F_{1/2} \left(\frac{E_V - E_{Fp}}{k_B T} \right) \end{aligned} \quad (126)$$

where:

- N_C and N_V are the conduction and valence effective band density-of-states (DOS) at 300 K, respectively.
- E_{Fn} and E_{Fp} are the electron and hole quasi-Fermi energies, respectively.
- E_C and E_V are the conduction and valence band edges, respectively.
- $F_{1/2}$ is the Fermi–Dirac integral of the order one-half.
- k_B is the Boltzmann constant.
- T is the ambient temperature.

The values of N_C and N_V are independent of strain, and they can be set using the following Physics commands:

```
Physics material=Silicon eBulkDensity=eFermiDensity Nc=1.0e19
Physics material=Silicon hBulkDensity=hFermiDensity Nv=2.0e19
```

Table 33 lists the default values for N_C and N_V .

Table 33 Default parameter values for Fermi–Dirac bulk density model

Model name	Parameter	Unit	Silicon	Polysilicon
eFermiDensity	Nc	cm ⁻³	2.8567e19	2.8567e19
hFermiDensity	Nv	cm ⁻³	3.1046e19	3.1046e19

Bulk Density Model Based on an Ellipsoidal Valley

For one band of an ellipsoidal valley model, such as `ConstantEllipsoid`, the bulk carrier densities, assuming parabolic dispersion and Fermi–Dirac statistics, can be computed using the models `eEllipsoidalDensity` and `hEllipsoidalDensity`, and are given by:

$$\begin{aligned} n &= 2g_V(m_1m_2m_3)^{1/2} \left(\frac{k_B T}{2\pi\hbar^2} \right)^{3/2} F_{1/2} \left(\frac{E_{Fn} - (E_C + \Delta E_C)}{k_B T} \right) \\ p &= 2g_V(m_1m_2m_3)^{1/2} \left(\frac{k_B T}{2\pi\hbar^2} \right)^{3/2} F_{1/2} \left(\frac{(E_V + \Delta E_V) - E_{Fp}}{k_B T} \right) \end{aligned} \quad (127)$$

where:

- g_V is the valley degeneracy.
- m_1 , m_2 , and m_3 are the principal masses of the ellipsoidal valley.
- E_C , E_V are the relaxed conduction and valence band edges, respectively.
- ΔE_C , ΔE_V are the shifts of the valley band edge relative to the relaxed conduction or valence band, respectively.

The valley-dependent parameters of this model, such as the masses and the band-edge shift, are defined or computed by the `ValleyModel` with the name specified by the `valley` argument.

Bulk Hole Density Model Based on 6kp

The valence band in silicon is composed of three bands at the Γ -point. In relaxed silicon, these are usually referred to as the heavy-hole, light-hole, and split-off bands. Under general strain, however, the bands and effective masses become mixed, making it difficult to label the bands in this way.

Due to the large warping of the valence bands in silicon, a simple ellipsoidal band is not an accurate model for the valence-band dispersion. Instead, a Fermi–Dirac model in conjunction with a more accurate calculation of the hole carrier-concentration effective mass, m_{cc} , in each band can be selected using the `6kpDensity` model. This model obtains its parameters from the `6kpValley` valley model specified by name with the `valley` argument and for the band specified by the `band` argument.

To compute the hole m_{cc} for one of the valence bands for arbitrary strain, the band structure provided by the six-band $k \cdot p$ method is integrated explicitly assuming Boltzmann statistics [4]. In this approximation, m_{cc} is given by:

$$m_{cc}^{3/2} = \frac{2}{\sqrt{\pi}} (k_B T)^{-3/2} \int_0^{\infty} dE m_{DOS}^{3/2}(E) \sqrt{E} e^{-E/k_B T} \quad (128)$$

Chapter 13: Subband and Mobility Calculations

Electrostatic Models

where the energy-dependent DOS mass, m_{DOS} , is given by:

$$m_{\text{DOS}}^{3/2} = \frac{\sqrt{2}\pi^2\hbar^3}{\sqrt{E}} \frac{1}{(2\pi)^3} \int_0^{2\pi} d\varphi \int_0^\pi d\theta \sin\theta \cdot \left[\left| \frac{\partial k}{\partial E} \right| k^2 \right] \quad (129)$$

The band structure-related integrand is computed from the inverse six-band $k \cdot p$ method in polar k -space coordinates. The integrals for m_{cc} and m_{DOS} are evaluated using optimized quadrature rules.

Using m_{cc} , the density within the treated band is given by:

$$p = 2(m_{\text{cc}})^{3/2} \left(\frac{k_B T}{2\pi\hbar^2} \right)^{3/2} F_{1/2} \left(\frac{(E_V + \Delta E_V) - E_{Fp}}{k_B T} \right) \quad (130)$$

where E_V is the relaxed valence band edge, and ΔE_V is the shift of the band edge for the treated band relative to E_V .

Multivalley Bulk Density Model

The conduction and valence bands in most semiconductors are composed of multiple valleys or multiple bands or both. For example, in silicon, the conduction band is composed of states from three pairs of Δ -valleys, while the valence band is composed of three bands at the Γ -point. Most of the bulk density models compute the density for only a single valley or band. To obtain the total carrier density, these individual densities must be summed.

This summation is performed automatically using the models `eMultiValleyDensity` and `hMultiValleyDensity` for electrons and holes, respectively. These models automatically sum the density from all density models created by valley models in which `useForEBulkDensity` or `useForHDensity` has been specified.

Default Models for Carrier Density

To treat both confined and bulk carrier densities for multiple valleys and bands, the following set of models is activated by default:

```
Physics material=semiconductor eDensity=eHybridDensity
Physics material=semiconductor eBulkDensity=eMultiValleyDensity
Physics material=semiconductor hDensity=hHybridDensity
Physics material=semiconductor hBulkDensity=hMultiValleyDensity
```

The actual bulk density models that are used for electrons and holes are determined by the default valley models.

Maxwell–Boltzmann Statistics

By default, the models that compute carrier densities use Fermi–Dirac statistics. Optionally, you can use Maxwell–Boltzmann statistics. In that case, the Fermi–Dirac integrals are

Chapter 13: Subband and Mobility Calculations

Electrostatic Models

replaced by exponential functions. This option can be selected on a region basis using the `CarrierStatistics` model in the `Physics` command. For example, to activate Maxwell–Boltzmann statistics in all regions, specify:

```
Physics material=all CarrierStatistics=Boltzmann
```

To set Fermi–Dirac statistics explicitly, specify `CarrierStatistics=Fermi`.

Doping Concentration

The main doping quantities are:

- Net doping concentration (`DopingConcentration`) is used in the Poisson equation.
- Total doping concentration (`TotalConcentration`) is used for Coulomb scattering.

The values of these doping quantities are determined by a set of rules that depend on which fields are read from the input TDR file.

By default, during the `LoadDevice` command, Sentaurus Band Structure reads the active doping concentration of all individual dopants that it recognizes. The `Dopant` command is used to specify properties of new dopants including quantities such as the dopant name, the element symbol, and the type of dopant (see [Dopant on page 358](#)). After a dopant has been specified, its active concentration is loaded from the TDR file if it is present. For example, for boron, a field with the model keyword `BoronActiveConcentration` will be created with values from the TDR file. By default, the following dopants have been specified for all semiconductor materials: boron, arsenic, and phosphorus.

To deactivate the loading of individual dopants, specify `ignoreDopants=1` in the `LoadDevice` command (see [LoadDevice on page 360](#)). From the doping-related fields read from the TDR file, other models are created such as `DonorConcentration`, `AcceptorConcentration`, `DopingConcentration`, and `TotalConcentration`.

Based on the value of `ignoreDopants`, the contents of the TDR file, and the known dopants specified with the `Dopant` command, [Table 34](#) lists the rules that are used to load and create doping information.

Table 34 Rules for loading and computing doping-related models

Model keyword	No dopants read from TDR file	At least one dopant read from TDR file
AcceptorConcentration	$(\text{TotalConcentration} - \text{DopingConcentration})/2$	Sum of active acceptor dopants read from TDR file
DonorConcentration	$(\text{TotalConcentration} + \text{DopingConcentration})/2$	Sum of active donor dopants read from TDR file

Chapter 13: Subband and Mobility Calculations

Electrostatic Models

Table 34 Rules for loading and computing doping-related models (Continued)

Model keyword	No dopants read from TDR file	At least one dopant read from TDR file
DopingConcentration	Read from DopingConcentration or NetActive in TDR file	DonorConcentration - AcceptorConcentration
TotalConcentration	DonorConcentration	DonorConcentration + AcceptorConcentration

If required, the doping that is loaded from the TDR file can be replaced by a constant value within a region using a `Physics` command of the following type:

```
Physics region=sil BoronActiveConcentration=ConstantModel value=1.0e18
```

Net Density

The bulk net density within a semiconductor region is given simply by the RHS of the Poisson equation ([Equation 121 on page 178](#)), excluding the interface charge, in units of cm^{-3} . This model can be saved for visualization purposes using the model keyword `NetDensity`.

Interface Charge

Either a fixed charge or trapped charge at region interfaces can be specified using `TrapModel` in the `Physics` command (see [Physics for Trap Models on page 374](#)). More than one value of the interface charge can be specified by issuing multiple `Physics` commands. The net interface charge at a particular region interface is then the sum of all the specified charges.

Region interfaces can be specified in terms of region names using the `regionInterface` argument or in terms of materials using the `materialInterface` argument using a slash (/) to separate the regions (or materials) on either side of the interface.

For example, to specify the interface between a region named `sil` and a region named `ox1`, the interface would be specified as `regionInterface="sil/ox1"`. The order of the specified regions does not matter. Likewise, you can use `materialInterface` as a shorthand for all interfaces between regions of particular materials. For example:

"Silicon/Oxide"

Chapter 13: Subband and Mobility Calculations

Electrostatic Models

Fixed Interface Charge

A fixed interface charge is specified using `TrapModel=FixedCharge`. This model has only the parameter `conc` that specifies the interface charge in units of cm^{-2} . For example:

```
Physics regionInterface="sil/ox1" TrapModel=FixedCharge conc=1e12 \
    name=Qss1
Physics regionInterface="sil/ox1" TrapModel=FixedCharge conc=-1e12 \
    name=Qss2
```

Here, two fixed charges are specified at the same interface. The fixed charge can be specified with either a positive or negative concentration. In addition, each fixed charge can be given a unique name, which allows the value of the fixed charge to be changed later in the simulation by specifying this name. For example:

```
Physics regionInterface="sil/ox1" TrapModel name=Qss1 conc=2e12
```

During initial specification, if `name` is not specified, then a default name is created.

Interface Traps

An interface trap is specified using either `TrapModel=Donor` for donor-type traps or `TrapModel=Acceptor` for acceptor-type traps. Donor traps are positively charged when occupied by a hole; acceptor traps are negatively charged when occupied by an electron. The interface trapped-charge concentration in each type of trap is given by:

$$\text{Donor: } Q_{it} = e \int_{E_{\min}}^{E_{\max}} D_{it}(E) f_D^+(E) dE \quad (131)$$

$$\text{Acceptor: } Q_{it} = -e \int_{E_{\min}}^{E_{\max}} D_{it}(E) f_A^-(E) dE \quad (132)$$

where $D_{it}(E)$ is the energy profile of the interface trap density.

The distribution functions for trap occupancy are given by $f_D^+(E)$ for the hole occupancy of a donor trap and $f_A^-(E)$ for the electron occupancy of an acceptor trap:

$$f_D^+(E) = \frac{1}{1 + g \cdot e^{(E_F - E)/k_B T}} \quad (133)$$

$$f_A^-(E) = \frac{1}{1 + g \cdot e^{(E - E_F)/k_B T}} \quad (134)$$

Chapter 13: Subband and Mobility Calculations

Electrostatic Models

where:

- E_F is the quasi-Fermi energy of the carrier specified by the `carrierType` argument.
- g is a degeneracy parameter. Its default value is 1, which makes $f_D^+(E)$ and $f_A^-(E)$ both a standard equilibrium Fermi–Dirac distribution function. Other common values for g are $g=2$ for donor traps and $g=4$ for acceptor traps.

Energy Profile

You use `DitProfile` to specify the energy profile of the interface trap density. Different types of profile are available (see [Physics for Trap Models on page 374](#)). For all profiles, energies are specified relative to the relaxed valence band. All profiles are treated as a continuous distribution in energy and taken to have positive D_{it} . The sign of the trapped charge is determined by the trap type, either donor or acceptor:

- `DitProfile=exponential` defines an exponential profile to either side of the `EnergyMid` argument. It is parameterized by the `conc`, `EnergyMid`, and `EnergySig` arguments:

$$\text{conc} \cdot \exp\left(-\left|\frac{E - \text{EnergyMid}}{\text{EnergySig}}\right|\right)$$

`EnergyMid` is defined relative to the relaxed valence band.

- `DitProfile=Gaussian` defines a Gaussian profile that is parameterized by the `conc`, `EnergyMid`, and `EnergySig` arguments:

$$\text{conc} \cdot \exp\left(-\frac{(E - \text{EnergyMid})^2}{2 \cdot \text{EnergySig}}\right)$$

`EnergyMid` is defined relative to the relaxed valence band.

- `DitProfile=table` defines a table profile, which is specified by `DitTable` as a Tcl list of `{Energy Dit}` pairs in which the energy is given in eV relative to the relaxed valence band, and D_{it} is given in $\text{eV}^{-1} \text{cm}^{-2}$. To produce a continuous D_{it} distribution from this list, linear interpolation on $\log(D_{it})$ is performed.
- `DitProfile=uniform` defines a uniform trap energy profile with a density given by the `conc` argument.

Energy Range

The energy range for the trap profile is defined using the `Emin` and `Emax` arguments. For analytic profiles, the default values of these parameters are the relaxed valence band and the relaxed conduction band, respectively, that is, the trap profile is contained within the relaxed band gap. You can extend the trap profile into the valence band or into the

Chapter 13: Subband and Mobility Calculations

Electrostatic Models

conduction band by setting `Emin` and `Emax` appropriately. For the table profile, the minimum and maximum energies from the specified `DitTable` list are used.

For convenience, `Emin`, `Emax`, and `EnergyMid` can be specified with one of the following values:

- `CondBand` = Relaxed conduction band edge
- `ValBand` = Relaxed valence band edge
- `MidBandGap` = Middle of the relaxed band gap
- Numeric value = Value interpreted as relative to the relaxed valence band edge

The material or region that provides the reference band edges for defining the trap energy profile can be specified using `refMaterial` or `refRegion`, respectively. By default at a semiconductor–insulator interface, the semiconductor band edges are taken as reference. At a semiconductor–semiconductor interface, you must specify the reference region or material.

Note:

Interface traps at insulator–insulator interfaces are not supported.

Visualization of Interface Charge

The spatial profile of a trapped interface charge can be saved to a TDR file using the `Save` command with the `eInterfaceTrappedCharge` and `hInterfaceTrappedCharge` model parameters for the total negative trapped charge and the total positive trapped charge, respectively. These quantities include the charge from fixed and trapped interface charges in units of cm^{-2} .

For interface traps, the energy profile of D_{it} and other quantities for a specific trap can be saved to a TDR file by using the `SaveDitProfile` command (see [SaveDitProfile on page 385](#)). With this command, you specify the trap by name and a point in the structure near the interface node for which the profile quantities are saved. Sentaurus Band Structure locates the nearest interface point and saves the quantities listed in [Table 35](#) to the TDR file.

Table 35 Interface trap profile–related quantities saved to TDR file

Quantity	Description
<code>DistributionFunction</code>	Distribution function for trap occupancy in units of [1]
<code>Ec</code>	Indicated by 1 at the energy of the relaxed conduction band
<code>Ev</code>	Indicated by 1 at the energy of the relaxed valence band
<code>Ef</code>	Indicated by 1 at the energy of the quasi-Fermi energy

Chapter 13: Subband and Mobility Calculations

Electrostatic Models

Table 35 Interface trap profile-related quantities saved to TDR file (Continued)

Quantity	Description
Energy	Energy relative to the relaxed valence band in units of eV
TrapDensity	Interface trap density in units of $\text{eV}^{-1}\text{cm}^{-2}$
TrappedChargeDensity	Density of trapped charge in units of $\text{eV}^{-1}\text{cm}^{-2}$

Effective Field for Universal Mobility

The effective field concept is useful for simplifying the modeling and characterization of inversion-layer mobility in silicon MOSFETs [5]. Experimentally, the effective field is defined as a linear combination of the inversion sheet density (N_{inv}) and the depletion sheet density (N_{dep}) as:

$$E_{\text{eff}} = \frac{e}{\epsilon_s} (\eta N_{\text{inv}} + N_{\text{dep}}) \quad (135)$$

where ϵ_s is the permittivity of the semiconductor. The weighting factor η has been experimentally determined to differ for electrons and holes. Models have been defined to compute the integrand of this charge-based effective field. The weighting factor η can be set, for example, for electrons, by using:

```
Physics material=Silicon eEffIntegrand nu=0.5
```

[Table 36](#) lists the model keywords and default values for η .

Table 36 Charge-based effective field models

Model keyword	Parameter	Unit	Silicon
eEffIntegrand	nu	1	0.5
hEffIntegrand	nu	1	0.333

To compute the actual effective field value, these models must be integrated over the appropriate region using the [Extract](#) command (see [Extract on page 359](#)). For example:

```
Set Eeff [Extract model=eEffIntegrand region=sil integral]
```

Chapter 13: Subband and Mobility Calculations

Electrostatic Models

Alternatively, you can also define the effective field in terms of a weighted integral of the local electric field, E_z :

$$E_{\text{eff}} = \frac{\int dx n(x) E_z(x)}{\int dx n(x)} \quad (136)$$

where n can be either the electron density or the hole density, whichever is inverted.

The calculation of the effective field in this approach is treated by providing a model for the integrand of the numerator called `eDensityTimesEField` for electrons and `hDensityTimesEField` for holes. The actual value of the effective field can then be computed by integrating over the appropriate region and dividing by the inversion sheet density.

For example:

```
set nTimesE [Extract model=eDensityTimesEField region=sil integral]
set Ninv [Extract model=eDensity region=sil integral]
set Eeff [expr $nTimesE/$Ninv]
```

Strain

The strain tensor in the crystallographic coordinate system is set using the `CrystalStrain` model keyword. Only constant strain within a region is supported. The strain tensor can be computed using the elasticity features of Sentaurus Band Structure.

The following example shows how to apply 1 GPa of uniaxial stress along <110>:

```
SiliconCrystal name=mySi
mySi apply uniaxialStrain dir=[list 1 1 0] stress=1.0e9
set strainTensor [mySi get strain]
Physics material=Silicon CrystalStrain strain=$strainTensor
```

The strain tensor is used internally by many different models. In addition, the strain tensor can be saved for visualization using the `CrystalStrain` model keyword.

Polarization-Induced Charge

For wurtzite semiconductor materials, the spontaneous and piezoelectric polarization-induced charges are captured at all interfaces of each wurtzite semiconductor.

Chapter 13: Subband and Mobility Calculations

Electrostatic Models

The polarization vector \mathbf{P} is expressed as:

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = activation \times (\mathbf{P}_{sp} + \mathbf{P}_{pz}) \quad (137)$$

where \mathbf{P}_{sp} is the spontaneous polarization vector, \mathbf{P}_{pz} is the piezoelectric polarization vector, and *activation* is a nonnegative real calibration parameter with a default value of 1. Here, the spontaneous polarization vector is defined as:

$$\mathbf{P}_{sp} = \begin{bmatrix} 0 \\ 0 \\ P_{sp} \end{bmatrix} \quad (138)$$

The piezoelectric polarization vector as a function of the strain tensor is written as:

$$\mathbf{P}_{pz} = \begin{bmatrix} 0 & 0 & 0 & 0 & e_{15} & 0 \\ 0 & 0 & 0 & e_{15} & 0 & 0 \\ e_{31} & e_{31} & e_{33} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ \epsilon_{yz} \\ \epsilon_{xz} \\ \epsilon_{xy} \end{bmatrix} \quad (139)$$

The spontaneous and piezoelectric polarization vectors are defined in the crystal coordinate system. The polarization vector \mathbf{P} is first computed in crystal coordinates and is converted to the simulation coordinate system afterwards.

To capture the spontaneous or piezoelectric polarization charge, the electrostatic models must be defined for each relevant material or region. The model can be added using the following syntax:

```
Physics {material=String | region=String} \
    SpontaneousPolarization=SpontaneousPolarizationModel \
    Psp=Double

Physics {material=String | region=String} \
    PiezoelectricPolarization=PiezoelectricPolarizationModel \
    e15=Double e31=Double e33=Double
```

Chapter 13: Subband and Mobility Calculations

Electrostatic Models

After the polarization models are defined for a material or region, the model type can be omitted from the subsequent command. Therefore, the arguments can be adjusted with the following syntax:

```
Physics {material=String | region=String} \
    SpontaneousPolarization Psp=Double

Physics {material=String | region=String} \
    PiezoelectricPolarization e15=Double e31=Double e33=Double
```

The value of the spontaneous polarization charge, in units of C/cm², can be specified with the SpontaneousPolarization argument of the Physics command. For example:

```
Physics material=GaN \
    SpontaneousPolarization=SpontaneousPolarizationModel Psp=-0.034
```

The value of the piezoelectric polarization charge–strain tensor (e_{ij}), in units of C/cm², can be specified with the PiezoelectricPolarization argument of the Physics command. For example:

```
Physics material=GaN \
    PiezoelectricPolarization=PiezoelectricPolarizationModel \
    e15=0.3255 e31=-0.5274 e33=0.8946
```

The calibration parameter *activation* can be specified on a per-region or per-material basis by using the PolarizationActivation option of the Physics command. For example:

```
Physics material=GaN PolarizationActivation activation=1.0
```

The final polarization-induced charge is then computed according to $q_{\text{polar}} = \mathbf{n} \cdot \mathbf{P}$, where \mathbf{n} is the outward normal of the respective semiconductor interface.

Visualization of Polarization-Induced Charges

The spatial profile of all polarization-induced charges can be saved to a TDR file using the Save command with the SpontaneousPolarizationCharge, PiezoelectricPolarizationCharge, and PolarizationCharge model parameters. These models represent the spontaneous polarization charge, the piezoelectric polarization charge, and the sum of both polarization charge models, respectively. The quantities written to the TDR file are in units of cm⁻³.

In addition, you can save the polarization vectors to a TDR file using the Save command with the SpontaneousPolarization and PiezoelectricPolarization model parameters. These represent the spontaneous and piezoelectric polarization-induced charge as a vector quantity and are in units of C/cm². You can also save the activation parameter to a TDR file using the PolarizationActivation model parameter.

Calculating Subbands

The confined carrier density, the subband dispersion, and the wavefunctions within an inversion layer can be computed using different Schrödinger solvers. These solvers compute the subband dispersion and wavefunctions in k -space over previously defined nonlocal lines or nonlocal areas. These results are then used to directly compute the confined carrier density. Models for the minimum energy and wavefunction of each subband are created automatically for visualization or extraction purposes.

Polar Grid for 2D k-Space

For 1D device structures, the Schrödinger solvers are solved over a 2D k -space grid in polar coordinates. To reduce the computation time, these solves are performed only over the angular section of the polar grid that is unique from a symmetry perspective. This part of the polar grid is called the *irreducible wedge* (IW).

Figure 46 (Left) Polar grid over which a 1D Schrödinger equation is solved and (right) example of subband dispersion with irreducible wedge (IW) highlighted; in example, IW symmetry is IW4 and $\phi_0 = -\pi/4$

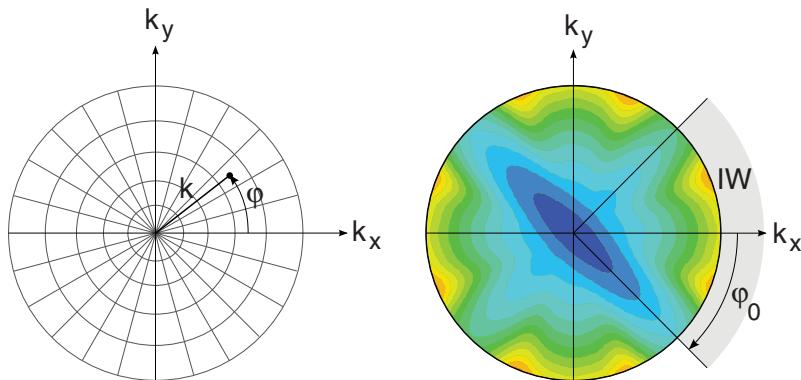


Figure 46 shows a polar grid and an example of a subband dispersion with its associated IW. The IW is described by two parameters: the IW symmetry that gives the number of equivalent sections that span 2π , and the starting angle for the IW, ϕ_0 , relative to the k_x -axis. The direction of the k_x -axis is specified using the `xDirection` argument of the `Physics` command. The example in Figure 46 has an IW symmetry of IW4 and $\phi_0 = -\pi/4$.

The supported IW symmetries are IW12, IW8, IW4, and IW2. The particular IW symmetry and ϕ_0 that should be used during a calculation depends on the carrier, the surface orientation, and the applied strain. Some Schrödinger solvers compute the IW symmetry and ϕ_0 parameter automatically, while others require you to specify these arguments in the `Physics` command. This is discussed in detail for each type of Schrödinger solver. For

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

interpolation of the subband dispersion and other quantities over the polar grid, the Fourier-spline approach is used [6].

You specify the polar grid by using a set of arguments of the `Physics nonlocal=String` command (see [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#)). The specified polar grid must be sufficiently fine to properly resolve the dispersion for all the computed subbands. For the radial k -grid, the grid points can be specified as uniform by using `Nk` and `Kmax`. Alternatively, the radial grid can be given as an explicit list of k -values using `kGrid`. The φ -grid is always uniformly spaced, and the total number of points spanning 2π is specified by using `Nphi`.

Note:

`Nphi` must be consistent with the IW symmetry, that is, `Nphi` must be divisible by the IW symmetry. If this is not the case, then `Nphi` is adjusted upwards automatically to the next smallest value that is consistent with the IW symmetry.

Grid for 1D k-Space

For 2D device structures, the solution of the 2D Schrödinger equation requires computing the dispersion on a 1D grid in k -space. You specify this k -grid in the `Physics nonlocal=String` command using either the `Nk` and `Kmax` arguments or the `kGrid` argument (see [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#)).

Ladders

The bands of many semiconductors are often composed of more than one valley. For example, the conduction band in silicon is composed of three pairs of Δ -valleys. The solution of a Schrödinger equation in each valley produces a *ladder* of subbands of increasing energy. The valleys to be included in the solution of a particular Schrödinger equation can be specified with the `Physics` command using the `valleys` argument. The required number of subbands to compute for each ladder is specified with the `Nsubbands` argument (see [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#)).

Dynamically Varying Subband Numbers

The number of subbands resolved within the solution of a particular Schrödinger equation controls the balance between accurate and efficient simulations. A sufficiently high number of subbands must be resolved to accurately capture the confined carrier density and the carrier transport properties. However, if too many subbands are requested, then the efficiency of the simulation degrades.

Being able to select the optimal number of subbands is difficult, without first having run the simulation with an excess number of subbands.

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

To overcome this challenge, a user-friendly feature entitled *dynamic subbands* allows the simulation to dynamically determine the optimal number of subbands required.

To activate this feature for the solution of a particular Schrödinger equation, specify the `useDynamicSubbands` argument in the `Physics` command (see [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#)).

To aid model visualization and extraction, you can use the `getSubbandNumbers` option of the `Physics` command to request the number of subbands dynamically determined during a simulation. It returns the number of subbands per ladder to the Tcl environment (see [Physics for Top-Level Parameters on page 365](#)).

Calculating the Confined Carrier Density

The confined carrier density is computed directly from the subband dispersion and wavefunctions using:

$$n(z) = \sum_v n_v(z) \quad \text{for 1D confinement} \quad (140)$$

$$n_v(z) = \frac{g_v}{(2\pi)^2} \int dk f_{FD}(E_v(k)) |\Psi_v(k; z)|^2$$

$$n(x, y) = \sum_v n_v(x, y) \quad \text{for 2D confinement} \quad (141)$$

$$n_v(x, y) = \frac{g_v}{(2\pi)^2} \int dk f_{FD}(E_v(k)) |\Psi_v(k; (x, y))|^2$$

where:

- g_v includes the degeneracy for both the spin and valley degeneracy of subband v .
- f_{FD} is the Fermi–Dirac distribution function.
- E_v is the subband dispersion.
- Ψ_v is the wavefunction.

In general, the sum over subbands, v , includes multiple valleys or ladders.

The accurate evaluation of the integrand in [Equation 140](#) possibly requires a finer radial grid than that used for computing the dispersion. For this reason, a separate k -grid can be specified in the `Physics` command for evaluating the carrier density using `NkForNinv` or `kGridForNinv` (see [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#)).

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

If you selected Maxwell–Boltzmann statistics in the first semiconductor region in the nonlocal line or nonlocal area, then the Fermi–Dirac distribution function is replaced by a Maxwell–Boltzmann distribution function.

Calculating the Thermal Injection Velocity

The thermal injection velocity for valley v along the direction \hat{u} is computed as the occupancy-weighted average of the group velocity over one-half of k -space.

Due to inversion symmetry, this can be computed as an integral over all k -space using:

$$v_{\text{inj}}^v = \frac{g_v \cdot s_v \sum_{i \in \text{Subbands}} \frac{1}{(2\pi)^d} \int f_{\text{FD}}(E_i(\vec{k})) \left| \frac{1}{\hbar} \nabla E_i(\vec{k}) \cdot \hat{u} \right|^d dk^d}{g_v \cdot s_v \sum_{i \in \text{Subbands}} \frac{1}{(2\pi)^d} \int f_{\text{FD}}(E_i(\vec{k})) dk^d} \quad (142)$$

where:

- g is the valley degeneracy.
- s is the spin degeneracy.
- f is the equilibrium distribution function, that is, either Fermi–Dirac or Maxwell–Boltzmann.
- E_i is the subband dispersion for subband i .
- d is the dimensionality of the k -space dispersion. For 1D confinement, $d = 2$. For 2D confinement, $d = 1$.

The overall injection velocity for all valleys is computed from the sum of the per-valley injection velocities weighted by the valley occupancies. The calculation of the thermal injection velocity can be initiated using the `ComputeVinj` command (see [ComputeVinj on page 357](#)).

Calculating the Transport Mass

The transport mass is computed from the occupancy-weighted inverse transport mass. The occupancy-weighted inverse transport mass is computed as one component of the inverse mass tensor integrated over all k -space and weighted by the equilibrium Fermi–Dirac or Maxwell–Boltzmann distribution function.

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

For valley v , the Im component of the occupancy-weighted inverse mass tensor is:

$$W_{\text{Im}}^v = \frac{g_v \cdot s_v \sum_{i \in \text{Subbands}} \frac{1}{(2\pi)^d} \int f_{\text{FD}}(E_i(\vec{k})) \frac{1}{\hbar^2} \frac{\partial^2}{\partial k_i \partial k_m} E_i(\vec{k}) dk^d}{g_v \cdot s_v \sum_{i \in \text{Subbands}} \frac{1}{(2\pi)^d} \int f_{\text{FD}}(E_i(\vec{k})) dk^d} \quad (143)$$

where:

- g is the valley degeneracy.
- s is the spin degeneracy.
- f is the equilibrium distribution function, that is, either Fermi–Dirac or Maxwell–Boltzmann.
- E_i is the subband dispersion for subband i .
- d is the dimensionality of the k -space dispersion. For 1D confinement, $d = 2$. For 2D confinement, $d = 1$.

The overall inverse transport mass for all valleys is computed from the sum of the per-valley inverse masses weighted by the valley occupancies. The calculation of the transport mass can be initiated using the [ComputeMass](#) command (see [ComputeMass on page 355](#)).

Automatically Created Models for Visualization and Extraction

All Schrödinger solvers automatically create, for each subband, a set of models for the subband energy minimum, the wavefunction value and norm at the center of the polar grid, and the carrier density profile in each subband. These models can be accessed for visualization or extraction purposes.

The names of the models start with the name of the valley or ladder for which the subband belongs and then the subband index starting at 0.

For example, the lowest-lying subband computed for the valley named `Delta1` has the following models created automatically:

- `Delta1_0_SubbandEnergy`
- `Delta1_0_SubbandDensity`
- `Delta1_0_Wavefunction`
- `Delta1_0_WavefunctionValue`

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

The next higher-lying subband within the same ladder has the following models created automatically:

- Delta1_1_SubbandEnergy
- Delta1_1_SubbandDensity
- Delta1_1_Wavefunction
- Delta1_1_WavefunctionValue

For the wavefunction values, the models that are saved to the TDR file are expanded automatically to include both the real and imaginary parts for the parabolic Schrödinger solver, as well as the different vector components for the $k \cdot p$ Schrödinger solvers. For example, when the wavefunction value for a `Delta1_0` subband from the parabolic Schrödinger solver is requested, the following models are saved to the TDR file:

`Delta1_0_WavefunctionValue_real, Delta1_0_WavefunctionValue_imag`

For the $k \cdot p$ Schrödinger solvers, such as the `6kp` Schrödinger solver, each vector component, as well as the real and imaginary parts, are saved automatically. For example, requesting the wavefunction value for the `Gamma_0` subband from the `6kp` Schrödinger solver causes the following models to be saved to the TDR file:

`Gamma_0_WavefunctionValue_0_real, Gamma_0_WavefunctionValue_0_imag,`
`Gamma_0_WavefunctionValue_1_real, Gamma_0_WavefunctionValue_1_imag,`
`Gamma_0_WavefunctionValue_2_real, Gamma_0_WavefunctionValue_2_imag,`
`Gamma_0_WavefunctionValue_3_real, Gamma_0_WavefunctionValue_3_imag,`
`Gamma_0_WavefunctionValue_4_real, Gamma_0_WavefunctionValue_4_imag,`
`Gamma_0_WavefunctionValue_5_real, Gamma_0_WavefunctionValue_5_imag`

The subband dispersion as a function of k -space can be saved to a TDR file using the `SaveK` command using the model name: `valley-name_subband-index_Dispersion`. For example, for a valley named `Delta1`, the name of the dispersion model for the lowest-lying subband is `Delta1_0_Dispersion`.

Parabolic Schrödinger Solver

The Parabolic Schrödinger solver is based on the solution of the parabolic Schrödinger equation with a correction for nonparabolicity computed using perturbation theory. This solver is most relevant for electrons. For an ellipsoidal valley, the parabolic Schrödinger equation is given by:

$$-\frac{\hbar^2}{2}(\nabla \cdot \underline{W}_c \cdot \nabla \Psi) + (\tilde{E}_b - E)\Psi = 0 \quad (144)$$

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

where:

- \hbar is the reduced Planck's constant.
- \underline{W}_c is the inverse mass tensor for the ladder for the confinement direction or directions.
- Ψ is the wavefunction.
- E is the eigenenergy at the center \mathbf{k} -point of the dispersion.
- \tilde{E}_b is the confining band-edge energy, which is typically computed in terms of the relaxed conduction or valence band edge plus some fixed or strain-dependent shift.

Dirichlet boundary conditions are applied to the boundaries of the nonlocal line or nonlocal area over which the Schrödinger equation is solved. The band parameters used in the Schrödinger equation are obtained from the valley models specified with the `valleys` parameter of the `Physics` command. These valley models must be one of the ellipsoidal models described in [Ellipsoidal Valleys on page 181](#).

Dispersion

For a parabolic band, the eigenenergy for a subband, E_v^0 , is combined with a parabolic dispersion function to compute the total dispersion, $E_v(\mathbf{k})$, over the \mathbf{k} -space grid:

$$E_v(\mathbf{k}) = E_v^0 + \gamma_v(\mathbf{k}) \quad (145)$$

where the parabolic dispersion function is given by $\gamma_v(\mathbf{k}) = \frac{\hbar^2}{2}(\mathbf{k} \cdot \underline{W}_d \cdot \mathbf{k})$, and \underline{W}_d is the inverse mass tensor for dispersion, and \mathbf{k} is the in-plane wavevector, both of which are specified relative to the in-plane device coordinates.

When nonparabolicity is considered, both the in-plane dispersion and the subband minimum energy can be altered. As described here, for computing this nonparabolic correction, three models are provided. The required model can be set using the `correction` argument. Setting `alpha=0` and `alphaZ=0` for the associated ellipsoidal valley suppresses any of these corrections.

Setting `correction=1` uses a model based on [7] in which only the in-plane dispersion is modified, resulting in a total dispersion of the form:

$$E_v(\mathbf{k}) = E_v^0 + \frac{1}{2\alpha}(-1 + \sqrt{1 + 4\alpha\gamma_v(\mathbf{k})}) \quad (146)$$

where α is the nonparabolicity parameter. This is the default model.

Setting `correction=2` uses a model based on [8], which adds a perturbative correction to the subband minimum energy and results in a total dispersion of the form:

$$E_v(\mathbf{k}) = E_v^0 + \alpha_z \langle (E_v^0 - \tilde{E}_b) \rangle_v + \frac{1}{2\alpha}(-1 + \sqrt{1 + 4\alpha\gamma_v(\mathbf{k})}) \quad (147)$$

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

where $\langle g \rangle_v = \int dS g(\vec{x}) |\Psi_v(\vec{x})|^2$. In this formulation, the in-plane dispersion is the same as in the first model in which the nonparabolicity is specified with the `alpha` parameter. The impact of nonparabolicity on the minimum subband energy is characterized separately by `alphaz`.

Setting `correction=3` uses a model based on [9] in which an alternative perturbative correction corrects both the subband minimum energy and in-plane dispersion, resulting in a total dispersion of the form:

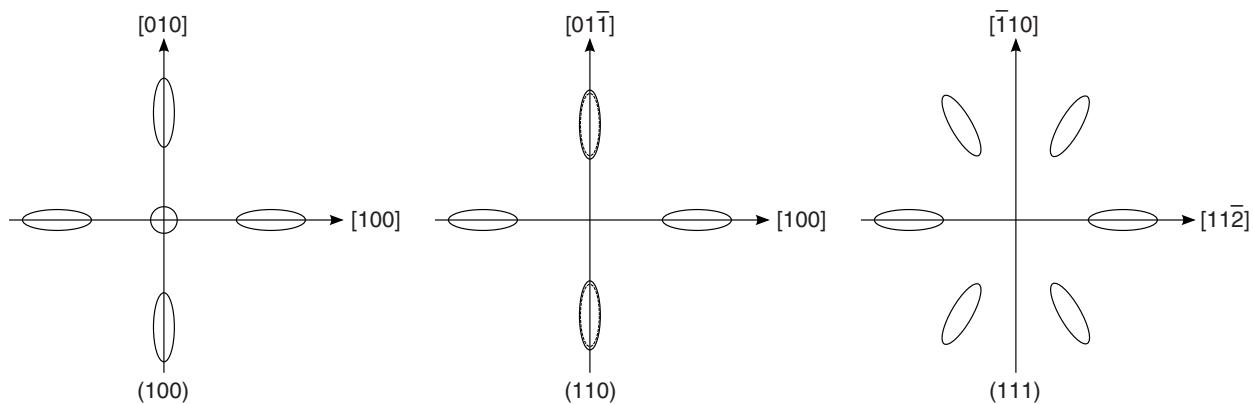
$$E_v(\mathbf{k}) = \tilde{\langle E_b \rangle}_v + \frac{-1 + \sqrt{1 + 4\alpha(\gamma_v(\mathbf{k}) + E_v^0 - \tilde{\langle E_b \rangle}_v)}}{2\alpha} \quad (148)$$

Surface Orientation and Effective Masses

Changes in surface orientation alter the inverse mass tensor that goes into the parabolic Schrödinger equation as well as the inverse mass tensor that is used for the dispersion. These tensors are computed automatically by transforming the inverse mass tensor of each ellipsoidal valley based on the user-specified device axes [10]. As an example, Figure 47 shows the orientations of the six Δ -valley constant-energy ellipses for (100), (110), and (111) surface orientations and typical x-directions.

For 1D confinement, the IW symmetry of the elliptical valleys is always IW4. The ϕ_0 parameter describing the start of the IW is computed automatically, based on the principal axes of each ellipsoidal valley as determined by the transformed inverse mass tensor.

Figure 47 In-plane constant-energy ellipses for 1D confinement for silicon Δ -valleys for (100), (110), and (111) surface orientations; the directions of the k_x - and k_y -axes for typical x-directions are also shown



Chapter 13: Subband and Mobility Calculations

Calculating Subbands

Using the Parabolic Schrödinger Solver

The `Parabolic` Schrödinger solver is selected using a `Physics` command of the type:

```
Physics nonlocal=NL1 eSchrodinger=Parabolic correction=1 \
valleys=[list Delta1 Delta2 Delta3] Nk=22 Kmax=0.3 Nphi=16
```

This example specifies to solve the `Parabolic` Schrödinger equation on the previously defined nonlocal line named `NL1`. Note that the valley names listed in `valleys` should each refer to one of the ellipsoidal valley models described in [Ellipsoidal Valleys on page 181](#). In this example, the nonparabolic correction model is set to model 1. Parameters for the polar grid for 1D confinement over which the subband dispersion is computed are specified as well. For 2D confinement, the `Nphi` argument is ignored.

In this example, a uniformly spaced \mathbf{k} -grid is specified with a maximum k -value of 0.3 and 22 points. The φ -grid is always uniformly spaced and is specified here to have 16 points.

Note:

The arguments `iwSymmetry` and `phi0` do not need to be specified since they are computed automatically by the `Parabolic` Schrödinger solver.

Confined $\mathbf{k}\cdot\mathbf{p}$ Schrödinger Solvers

In contrast to the `Parabolic` Schrödinger solver, which is based on the single-band effective mass approximation, the confined $\mathbf{k}\cdot\mathbf{p}$ Schrödinger solvers are based on a perturbation method involving multiple bands.

In an extended crystal and in the absence of a confining potential well, this perturbation method results in a bulk $\mathbf{k}\cdot\mathbf{p}$ Hamiltonian operator. For example, an expansion in terms of the six valence-band states around the Γ -point results in the Luttinger–Kohn or Bir–Pikus Hamiltonian of the equation in [Luttinger–Kohn or Bir–Pikus Hamiltonian on page 170](#). This operator is a 6×6 matrix, whose elements are functions of the \mathbf{k} -vector and the strain tensor ε .

In an inversion layer, the symmetry of the system is reduced by the presence of a surface and a confining potential well $V(\vec{x})$: Translational invariance along the confining direction or directions is lost.

This situation is addressed by the formal substitution $k_z \rightarrow -i\hbar\partial/\partial z$ for 1D confinement and $k_x \rightarrow -i\hbar\partial/\partial x, k_y \rightarrow -i\hbar\partial/\partial y$ for 2D confinement in the $\mathbf{k}\cdot\mathbf{p}$ Hamiltonian. This turns the Hamiltonian into a differential operator operating on a vector valued envelope wavefunction Ψ with one component per band of the bulk $\mathbf{k}\cdot\mathbf{p}$ Hamiltonian.

Arbitrary Surface Orientations

In the discussion of analytic bulk band structures (see [Chapter 12 on page 165](#)), coordinates were always assumed to be in the *crystallographic coordinate system*, that is,

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

the x-, y-, and z-axes are aligned with the (100) directions of the bulk crystal. For arbitrary orientation, it is necessary to define the *device coordinate system*. For 1D confinement, the confinement direction is along the z'-axis, and the x'- and y'-axes span the surface parallel plane. For 2D confinement, the x'- and y'-axes define the confinement plane. Primed coordinates refer to the device coordinate system; whereas, unprimed coordinates refer to crystal coordinates.

For arbitrary orientations, you must transform the bulk $\mathbf{k} \cdot \mathbf{p}$ Hamiltonian from its representation in the crystallographic coordinate system (H_{bulk}) to its representation in device coordinates (H'_{bulk}) before replacing the confined wavevectors with confinement operators. The details of this transformation depend on the particular choice of the underlying bulk $\mathbf{k} \cdot \mathbf{p}$ model and are discussed for the case of six-band $\mathbf{k} \cdot \mathbf{p}$ for holes and two-band $\mathbf{k} \cdot \mathbf{p}$ for electrons.

In both cases, subband energies E and envelope wavefunctions $\vec{\Psi}$ are found by solving the eigenvalue problem (shown here for 1D confinement):

$$\underbrace{\left(H'_{\text{bulk}} \Big|_{\mathbf{k}' = (k'_x, k'_y, -i\hbar\partial/\partial z')^T} + V(z') \right)}_{=: H'_{\text{confined}} \Big|_{\mathbf{k} = (k'_x, k'_y)^T}} \vec{\Psi}_{vk'_x k'_y}(z') = E_{vk'_x k'_y} \vec{\Psi}_{vk'_x k'_y}(z') \quad (149)$$

Reordering the Subband Dispersion

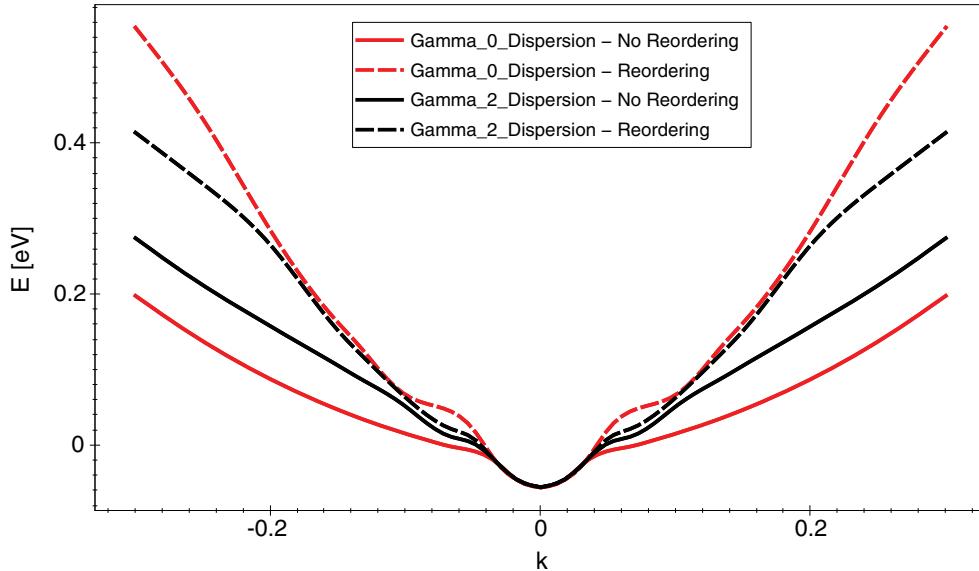
For 2D confined systems, an algorithm is available to reorder the numerically computed subband dispersion, following the concepts outlined in [11].

As shown in [Figure 48](#), reordering a $6kp$ subband dispersion allows the formation of light hole-like subbands. See the `reorderDispersion` argument in [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#).

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

Figure 48 Reordering of a $6kp$ subband dispersion allows the formation of light hole-like subbands



Confined Six-Band $k \cdot p$ Schrödinger Solver

The implementation of the confined six-band $k \cdot p$ method follows [1][12]. The 6×6 $k \cdot p$ Hamiltonian associated with the Luttinger–Kohn basis states can be expressed based on the 3×3 spin degenerate $k \cdot p$ Hamiltonian associated with the *chemical basis* states.

Within the crystallographic coordinate system x, y, z , the 3×3 spin degenerate $k \cdot p$ Hamiltonian associated with the chemical basis states $|p_x\rangle, |p_y\rangle, |p_z\rangle$ is expressed as follows [1]:

$$H_{\text{bulk}}^{3 \times 3} = \begin{pmatrix} k_x L k_x + k_y M k_y + k_z M k_z & k_x N^+ k_y + k_y N^- k_x & k_x N^+ k_z + k_z N^- k_x \\ k_y N^+ k_x + k_x N^- k_y & k_x M k_x + k_y L k_y + k_z M k_z & k_y N^+ k_z + k_z N^- k_y \\ k_z N^+ k_x + k_x N^- k_z & k_z N^+ k_y + k_y N^- k_z & k_x M k_x + k_y M k_y + k_z L k_z \end{pmatrix} \quad (150)$$

where the parameters L, M, N relate to the $k \cdot p$ parameters $\gamma_1, \gamma_2, \gamma_3$ as follows:

$$L = \frac{\hbar^2}{2m_0}(-\gamma_1 - 4\gamma_2) \quad M = \frac{\hbar^2}{2m_0}(-\gamma_1 + 2\gamma_2) \quad N = \frac{\hbar^2}{2m_0}(-6\gamma_3) = N^+ + N^- \quad (151)$$

Here, N^- and N^+ are given by [1]:

$$N^- = M - \frac{\hbar^2}{2m_0} \quad N^+ = N - N^- \quad (152)$$

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

Note that, since N^+ and N^- differ, the operator ordering established by:

$$H_{\text{bulk}}^{3 \times 3} \left(k_x \leftarrow -i\hbar \frac{d}{dx}, k_y \leftarrow -i\hbar \frac{d}{dy}, k_z \leftarrow -i\hbar \frac{d}{dz} \right) \quad (153)$$

forms a nonsymmetric matrix form. This nonsymmetric-operator ordering approach can eliminate spurious solutions for the wavefunction within a heterostructure quantum well [1].

It is important to note that the six (including spin degeneracy) top valence bands at the Γ -point consist of p orbitals of the lattice atoms. The Luttinger–Kohn Hamiltonian (see [Luttinger–Kohn or Bir–Pikus Hamiltonian](#)) is expressed in terms of total angular momentum $|j, m_j\rangle$ states, which are linear combinations of the chemical basis states $|p_x, +\rangle, |p_y, +\rangle, |p_z, +\rangle, |p_x, -\rangle, |p_y, -\rangle$, and $|p_z, -\rangle$ where + and – denote the spin-up and spin-down states, respectively. Therefore, the $6 \times 6 \mathbf{k} \cdot \mathbf{p}$ Hamiltonian associated with the Luttinger–Kohn basis states can be expressed, based on $H_{\text{bulk}}^{3 \times 3}$, as follows:

$$H_{\text{bulk}} = H_{\text{bulk}}^{6 \times 6} = A \begin{pmatrix} H_{\text{bulk}}^{3 \times 3} & 0 \\ 0 & H_{\text{bulk}}^{3 \times 3} \end{pmatrix} A^\dagger \quad (154)$$

where the 6×6 unitary matrix A connects the Luttinger–Kohn basis states to the chemical basis states (compared to [13]).

To transform the bulk Hamiltonian correctly between the crystal and device coordinates, it is important to note that the chemical basis states have vector-like transformation properties. Therefore, for any fixed wavevector \mathbf{k}' , the Hamiltonian in device coordinates can be expressed as:

$$H_{\text{bulk}}'^{3 \times 3} \Big|_{\mathbf{k}'} = U H_{\text{bulk}}^{3 \times 3} \Big|_{\mathbf{k} = U^T \mathbf{k}'} U^T \quad (155)$$

where the real 3×3 rotation matrix U transforms between crystal and device coordinates (for example, $U \hat{\mathbf{n}} = (0 \ 0 \ 1)^T$).

Consequently, the Luttinger–Kohn Hamiltonian associated with the device coordinate system can be expressed as:

$$H_{\text{bulk}}' \Big|_{\mathbf{k}'} = A \begin{pmatrix} H_{\text{bulk}}^{3 \times 3} \Big|_{\mathbf{k}'} & 0 \\ 0 & H_{\text{bulk}}^{3 \times 3} \Big|_{\mathbf{k}'} \end{pmatrix} A^\dagger = A \begin{pmatrix} U & 0 \\ 0 & U \end{pmatrix} \begin{pmatrix} H_{\text{bulk}}^{3 \times 3} \Big|_{\mathbf{k} = U^T \mathbf{k}'} & 0 \\ 0 & H_{\text{bulk}}^{3 \times 3} \Big|_{\mathbf{k} = U^T \mathbf{k}'} \end{pmatrix} \begin{pmatrix} U^T & 0 \\ 0 & U^T \end{pmatrix} A^\dagger \quad (156)$$

For 1D confinement, to apply the $k_z' \rightarrow -i\hbar\partial/\partial z'$ substitution, it is useful to express the $\mathbf{k} \cdot \mathbf{p}$ Hamiltonian as a polynomial in k_z' :

$$H_{\text{bulk}}' = k_z'^{(2)} k_z' + k_z'^{(1+)} + H^{(1-)} k_z' + H^{(0)} \quad (157)$$

In the case of a six-band $\mathbf{k} \cdot \mathbf{p}$, the $H^{(i)}$ take the form:

$$H^{(2)} = M^{zz} \quad (158)$$

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

$$H^{(1+)} = M^{zy}k'_y + M^{zx}k'_x \quad (159)$$

$$H^{(1-)} = k'_y M^{yz} + k'_x M^{xz} \quad (160)$$

$$H^{(0)} = k'_x M^{xx}k'_x + k'_y M^{yy}k'_y + k'_x M^{xy}k'_y + k'_y M^{yx}k'_x + H'_{\text{bulk}} \Big|_{k'=0} \quad (161)$$

where $M^{\alpha\beta}$ is the component of the expansion of the Hamiltonian:

$$H'_{\text{bulk}} = H'_{\text{bulk}} \Big|_{k'=0} + \sum_{\alpha \in \{x, y, z\}} \sum_{\beta \in \{x, y, z\}} k'_\alpha M^{\alpha\beta} k'_\beta \quad (162)$$

For an arbitrary surface orientation, $M^{\alpha\beta}$ can be determined as follow:

$$M^{\alpha\beta} = A \begin{pmatrix} U & 0 \\ 0 & U \end{pmatrix} \begin{pmatrix} H_{\text{bulk}}^{3 \times 3, \alpha\beta} & 0 \\ 0 & H_{\text{bulk}}^{3 \times 3, \alpha\beta} \end{pmatrix} \begin{pmatrix} U^T & 0 \\ 0 & U^T \end{pmatrix} A^\dagger \quad (163)$$

where:

$$H_{\text{bulk}}^{3 \times 3, \alpha\beta} = \begin{pmatrix} U_{\alpha x} L U_{\beta x} + U_{\alpha y} M U_{\beta y} + U_{\alpha z} M U_{\beta z} & U_{\alpha x} N^+ U_{\beta y} + U_{\alpha y} N^- U_{\beta x} & U_{\alpha x} N^+ U_{\beta z} + U_{\alpha z} N^- U_{\beta x} \\ U_{\alpha y} N^+ U_{\beta x} + U_{\alpha x} N^- U_{\beta y} & U_{\alpha x} M U_{\beta x} + U_{\alpha y} L U_{\beta y} + U_{\alpha z} M U_{\beta z} & U_{\alpha y} N^+ U_{\beta z} + U_{\alpha z} N^- U_{\beta y} \\ U_{\alpha z} N^+ U_{\beta x} + U_{\alpha x} N^- U_{\beta z} & U_{\alpha z} N^+ U_{\beta y} + U_{\alpha y} N^- U_{\beta z} & U_{\alpha x} M U_{\beta x} + U_{\alpha y} M U_{\beta y} + U_{\alpha z} L U_{\beta z} \end{pmatrix} \quad (164)$$

Here, $U_{\alpha x}$, $U_{\alpha y}$, $U_{\alpha z}$ and $U_{\beta x}$, $U_{\beta y}$, $U_{\beta z}$ are components of the unitary rotation matrix U .

Note:

The following symmetry $H_{\text{bulk}}^{3 \times 3, \alpha\beta} = (H_{\text{bulk}}^{3 \times 3, \beta\alpha})^T$ holds.

Consequently, $M^{\alpha\beta}$ has the following symmetry: $M^{\alpha\beta} = (M^{\beta\alpha})^\dagger$.

Therefore, it can be shown that $H^{(1+)} = (H^{(1-)})^\dagger$ holds, and $H^{(2)}$, $H^{(0)}$ are Hermitian matrices. These important symmetries ensure that the eigenproblem of the quantum confinement system is of a Hermitian type, which in turn ensures a real (noncomplex) value for the eigen energies.

To reduce the computation time of the confined six-band $k \cdot p$ Schrödinger solver, the symmetry of the valence subband structure for 1D confinement can be detected automatically using the algorithm based on [14]. To use this feature, specify `iwSymmetry=AUTO` in the `Physics nonlocal=String` command (see [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#)). Internally, the `iwSymmetry` and `phi0` values, for a given configuration of the surface/channel orientation and strain, are determined automatically.

For 2D confinement, only the positive half of 1D k -space needs be considered due to inversion symmetry.

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

For 1D confinement, however, the `iwSymmetry` and `phi0` values in the `Physics nonlocal=String` command are expected to be either specified manually or computed using `iwSymmetry=AUTO`. [Table 37](#) lists the `iwSymmetry` and `phi0` settings for typical cases.

Table 37 Typical symmetry settings for 1D-confined six-band $k \cdot p$

surfaceDirection	xDirection	strain	iwSymmetry	phi0
(001)	[100]	–	IW8	0
(001)	[110]	–	IW8	0
(110)	[1̄10]	–	IW4	0
(111)	[1̄10]	–	IW12	0
(111)	[1̄1̄2]	–	IW12	0
(001)	[100]	orthorhombic	IW4	0
(001)	[110]	orthorhombic	IW4	$\pi/4$
(001)	[100]	from [110] stress	IW4	$\pi/4$
(001)	[110]	from [110] stress	IW4	0
(110)	[1̄10]	from [1̄10] stress	IW4	0
(111)	[1̄10]	from [1̄10] stress	IW4	0
(111)	[1̄1̄2]	from [1̄10] stress	IW4	0
(001)	[110]	arbitrary	IW2	0
any	any	arbitrary	IW2	0

Using the Six-Band $k \cdot p$ Schrödinger Solver

You select the six-band $k \cdot p$ Schrödinger solver by specifying `hSchrodinger=6kp` in the `Physics nonlocal=String` command. For example, for 1D confinement, specify:

```
Physics nonlocal=NL1 hSchrodinger=6kp valleys=[list Gamma] \
Nk=22 Kmax=0.3 iwSymmetry=AUTO Nphi=48 \
useKdependentWF=1 useNonsymBulkKPHamil=1
```

This example specifies to solve the $6kp$ Schrödinger equation on the previously defined nonlocal line named `NL1`. The valley name listed in `valleys` must refer to a `6kpValley` valley model:

```
Physics material=Silicon ValleyModel=6kpValley name=Gamma \
degeneracy=1 useForHBulkDensity=1
```

Parameters for the polar grid over which the subband dispersion is computed are specified as well. In this example, a uniformly spaced radial k -grid is specified with a maximum k -value of 0.3 and 22 points. Automatic symmetry is switched on with `iwSymmetry=AUTO`, and the symmetry of the subband is detected automatically. Within the irreducible φ -range determined by `iwSymmetry=AUTO`, the φ -grid is always uniformly spaced and is specified here to have 48 points. For 2D confinement, the `Nphi` argument is ignored.

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

Schrödinger solvers compute the subband dispersion at each point in the IW of the specified k -space grid. Wavefunctions also can be computed at each k -space grid point by specifying `useKdependentWF=1`. Alternatively, the calculation of the wavefunctions can be limited to only the k -vector of the subband minimum. This is indicated by specifying `useKdependentWF=0`, which is the default value.

The nonsymmetric-operator ordering approach is used in this example by specifying `useNonsymBulkKPHamil=1`, which is the default value.

If `useNonsymBulkKPHamil=0`, then the symmetric-operator ordering approach is used, which implies:

$$N^+ = N^- = \frac{N}{2} \quad (165)$$

Confined Three-Band $k \cdot p$ Schrödinger Solver

The 3×3 spin degenerate $k \cdot p$ Hamiltonian can be expressed using the chemical basis states as outlined in [Confined Six-Band \$k \cdot p\$ Schrödinger Solver on page 213](#).

Using the Three-Band $k \cdot p$ Schrödinger Solver

You can specify the three-band $k \cdot p$ valley model by setting `ValleyModel=3kpValley`. For example:

```
Physics material=Silicon ValleyModel=3kpValley name=Gamma \
    degeneracy=1 gamma1=3.6 gamma2=0.67 gamma3=1.21 \
    a_v=2.1 b=-2.33 d=-4.75 useForHBulkDensity=0
```

In addition, the `3kp` keyword is available for the `hSchrodinger` argument. For example:

```
Physics nonlocal=NL1 hSchrodinger=3kp valleys=Gamma
```

Confined Two-Band $k \cdot p$ Schrödinger Solver

The 2×2 $k \cdot p$ Hamiltonian for two nondegenerate conduction bands associated with the Z-valley is given by:

$$H_{\text{bulk}} = \frac{\hbar^2}{2} \begin{pmatrix} k_x \frac{1}{m_t} k_x + k_y \frac{1}{m_t} k_y + k_z \frac{1}{m_l} k_z - k_z \frac{1}{m_l} k_0 - k_0 \frac{1}{m_l} k_z & -2k_x \left(\frac{1}{Mm_t} \right)^+ k_y - 2k_y \left(\frac{1}{Mm_t} \right)^- k_x \\ -2k_y \left(\frac{1}{Mm_t} \right)^+ k_x - 2k_x \left(\frac{1}{Mm_t} \right)^- k_y & k_x \frac{1}{m_t} k_x + k_y \frac{1}{m_t} k_y + k_z \frac{1}{m_l} k_z + k_z \frac{1}{m_l} k_0 + k_0 \frac{1}{m_l} k_z \end{pmatrix} \quad (166)$$

In contrast to the valence bands of the six-band $k \cdot p$ method, the lowest conduction bands at the X-point consist of spherical s orbitals. This simplifies the relationship between H_{bulk} and H'_{bulk} to:

$$H'_{\text{bulk}}|_{k'} = H_{\text{bulk}}|_{k = (UU^T)^T k'} \quad (167)$$

where U is the same rotation matrix as in the six-band $k \cdot p$ case.

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

Here, U^V defined as:

$$U^{V=X} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad U^{V=Y} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad U^{V=Z} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (168)$$

is the rotation matrix used to rotate the X, Y, Z valley to the Z -valley.

Unlike the six-band case, the two-band bulk $\mathbf{k} \cdot \mathbf{p}$ Hamiltonian can contain first-order terms in \mathbf{k} . This can be observed in the expansion of H_{bulk} into k'_x, k'_y, k'_z :

$$H_{\text{bulk}} = H_{\text{bulk}}|_{k'=0} + \sum_{\alpha \in \{x, y, z\}} k'_\alpha M^{\alpha+} + \sum_{\alpha \in \{x, y, z\}} M^{\alpha-} k'_\alpha + \sum_{\alpha \in \{x, y, z\}} \sum_{\beta \in \{x, y, z\}} k'_\alpha M^{\alpha\beta} k'_\beta \quad (169)$$

For an arbitrary surface orientation, the component $M^{\alpha\beta}$ can be determined as follows:

$$M^{\alpha\beta} = \frac{\hbar^2}{2} \begin{pmatrix} \tilde{U}_{\alpha x} \frac{1}{m_t} \tilde{U}_{\beta x} + \tilde{U}_{\alpha y} \frac{1}{m_t} \tilde{U}_{\beta y} + \tilde{U}_{\alpha z} \frac{1}{m_l} \tilde{U}_{\beta z} - 2 \tilde{U}_{\alpha x} \left(\frac{1}{Mm_t} \right)^+ \tilde{U}_{\beta y} - 2 \tilde{U}_{\alpha y} \left(\frac{1}{Mm_t} \right)^+ \tilde{U}_{\beta x} \\ - 2 \tilde{U}_{\alpha y} \left(\frac{1}{Mm_t} \right)^+ \tilde{U}_{\beta x} - 2 \tilde{U}_{\alpha x} \left(\frac{1}{Mm_t} \right)^+ \tilde{U}_{\beta y} \quad \tilde{U}_{\alpha x} \frac{1}{m_t} \tilde{U}_{\beta x} + \tilde{U}_{\alpha y} \frac{1}{m_t} \tilde{U}_{\beta y} + \tilde{U}_{\alpha z} \frac{1}{m_l} \tilde{U}_{\beta z} \end{pmatrix} \quad (170)$$

and the components $M^{\alpha+}$ and $M^{\alpha-}$ are given by:

$$M^{\alpha+} = M^{\alpha-} = \frac{\hbar^2}{2} \begin{pmatrix} -\tilde{U}_{\alpha z} \frac{1}{m_l} k_0 & 0 \\ 0 & \tilde{U}_{\alpha z} \frac{1}{m_l} k_0 \end{pmatrix} \quad (171)$$

Here, $\tilde{U}_{\alpha x}, \tilde{U}_{\alpha y}, \tilde{U}_{\alpha z}$ and $\tilde{U}_{\beta x}, \tilde{U}_{\beta y}, \tilde{U}_{\beta z}$ are components of the unitary rotation matrix $U = UU^V$.

It can be easily shown that $M^{\alpha\beta} = (M^{\beta\alpha})^\dagger$ holds.

When the components $M^{\alpha\beta}$, $M^{\alpha+}$, and $M^{\alpha-}$ have been calculated, the matrices $H^{(2)}$, $H^{(1+)}$, $H^{(1-)}$, and $H^{(0)}$ can be calculated:

$$H^{(2)} = M^{zz} \quad (172)$$

$$H^{(1+)} = M^{zy} k'_y + M^{zx} k'_x + M^{z+} \quad (173)$$

$$H^{(1-)} = k'_y M^{yz} + k'_x M^{xz} + M^{z-} \quad (174)$$

$$H^{(0)} = k'_x M^{xx} k'_x + k'_y M^{yy} k'_y + k'_x M^{xy} k'_y + k'_y M^{yx} k'_x + k'_x M^{x+} + M^{x-} k'_x + k'_y M^{y+} + M^{y-} k'_y + H_{\text{bulk}}|_{k'=0} \quad (175)$$

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

Again, it can be shown that $H^{(1+)} = (H^{(1-)})^\dagger$ holds, and $H^{(2)}, H^{(0)}$ are Hermitian matrices. These important symmetries ensure that the eigenproblem of the quantum confinement system is of a Hermitian type, which in turn ensures a real (noncomplex) value for the eigenenergies.

For 2D confinement, only the positive half of 1D k -space needs be considered due to inversion symmetry. For 1D confinement, however, the `iwSymmetry` and `phi0` values in the `Physics nonlocal=String` command are expected to be either specified manually or computed using `iwSymmetry=AUTO`.

Using the Two-Band $k \cdot p$ Schrödinger Solver

You select the two-band $k \cdot p$ Schrödinger solver by specifying `eSchrodinger=2kp` in the `Physics nonlocal=String` command. For example, for 1D confinement, specify:

```
Physics nonlocal=NL1 eSchrodinger=2kp valleys=[list X1 X2 X3] \
Nk=22 Kmax=0.3 iwSymmetry=AUTO nPhi=48 \
useKdependentWF=1 useNonsymBulkKPHamil=1
```

This example specifies to solve the `2kp` Schrödinger equation on the previously defined nonlocal line named `NL1`. The valley names listed in `valleys` must refer to a `2kpValley` valley model:

```
Physics material=Silicon ValleyModel=2kpValley name=X1 valleyDir=0 \
degeneracy=1 useForEBulkDensity=1
Physics material=Silicon ValleyModel=2kpValley name=X2 valleyDir=1 \
degeneracy=1 useForEBulkDensity=1
Physics material=Silicon ValleyModel=2kpValley name=X3 valleyDir=2 \
degeneracy=1 useForEBulkDensity=1
```

The argument `valleyDir` must be used to define the k -space orientation of the respective Δ -valley.

As for the other $k \cdot p$ models, the `iwSymmetry=AUTO` functionality for detecting the symmetry of k -space is supported by the two-band $k \cdot p$ model.

If `useNonsymBulkKPHamil=0`, then the symmetric-operator ordering approach is used, which implies:

$$\left(\frac{1}{Mm_t}\right)^+ = \left(\frac{1}{Mm_t}\right)^- = \frac{1}{2} \left(\frac{1}{Mm_t}\right) \quad (176)$$

Confined Eight-Band $k \cdot p$ Schrödinger Solver

This implementation is very similar to the 6×6 $k \cdot p$ described in [Confined Six-Band \$k \cdot p\$ Schrödinger Solver on page 213](#). What differs here are the terms for the conduction band H_{cc} and the coupling between the conduction band and the valence band H_{cv} .

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

Choosing the chemical basis states $|s\rangle$, $|p_x\rangle$, $|p_y\rangle$, and $|p_z\rangle$ as a basis for the Hamiltonian, the second order in the \mathbf{k} -part of the conduction band Hamiltonian can be written as:

$$H_{cc}^{1 \times 1} = k' A_c k' \quad (177)$$

where the renormalized inverse effective mass A_c is defined as in [Equation 125 on page 185](#) and the possible non-alignment of the device coordinates with the crystal coordinates is taken into account by:

$$k' = U k \quad (178)$$

as in [Confined Six-Band \$k \cdot p\$ Schrödinger Solver on page 213](#). The coupling between the conduction band and the valence band is of the first order in \mathbf{k} and is given by:

$$H_{cv}^{1 \times 3} = [iP(U_{xx} + U_{xy} + U_{xz})k'_x \quad iP(U_{yx} + U_{yy} + U_{yz})k'_y \quad iP(U_{zx} + U_{zy} + U_{zz})k'_z] \quad (179)$$

$$H_{vc}^{3 \times 1} = [-ik'_x(U_{xx} + U_{xy} + U_{xz})P \quad -ik'_y(U_{yx} + U_{yy} + U_{yz})P \quad -ik'_z(U_{zx} + U_{zy} + U_{zz})P]^T \quad (180)$$

where P is defined as $P = \sqrt{\frac{\hbar^2}{2m_0}} E_p$.

The valence band Hamiltonian part of the second order in \mathbf{k} is formally given by the same expression as in the case of 6×6 $k \cdot p$:

$$H_{vv}^{3 \times 3} = \sum_{\alpha} \sum_{\beta} k'_{\alpha} M^{3 \times 3, \alpha\beta} k'_{\beta} \quad (181)$$

$$M^{3 \times 3, \alpha\beta} =$$

$$\begin{pmatrix} U_{\alpha x} \tilde{L} U_{\beta x} + U_{\alpha y} \tilde{M} U_{\beta y} + U_{\alpha z} \tilde{M} U_{\beta z} & U_{\alpha x} \tilde{N}^+ U_{\beta y} + U_{\alpha y} \tilde{N}^- U_{\beta x} & U_{\alpha x} \tilde{N}^+ U_{\beta z} + U_{\alpha z} \tilde{N}^- U_{\beta x} \\ U_{\alpha y} \tilde{N}^+ U_{\beta x} + U_{\alpha x} \tilde{N}^- U_{\beta y} & U_{\alpha x} \tilde{M} U_{\beta x} + U_{\alpha y} \tilde{L} U_{\beta y} + U_{\alpha z} \tilde{M} U_{\beta z} & U_{\alpha y} \tilde{N}^+ U_{\beta z} + U_{\alpha z} \tilde{N}^- U_{\beta y} \\ U_{\alpha z} \tilde{N}^+ U_{\beta x} + U_{\alpha x} \tilde{N}^- U_{\beta z} & U_{\alpha z} \tilde{N}^+ U_{\beta y} + U_{\alpha y} \tilde{N}^- U_{\beta z} & U_{\alpha x} \tilde{M} U_{\beta x} + U_{\alpha y} \tilde{M} U_{\beta y} + U_{\alpha z} \tilde{L} U_{\beta z} \end{pmatrix} \quad (182)$$

However, instead of the 6×6 $k \cdot p$ parameters L , M , and N , you use the renormalized parameters [\[2\]](#):

$$\tilde{L} = L + \frac{E_p}{E_g + \Delta_{so}/3} \quad \tilde{M} = M \quad \tilde{N} = N + \frac{E_p}{E_g + \Delta_{so}/3} \quad (183)$$

The relation between the parameters L , M , and N , and the Luttinger parameters γ_1 , γ_2 , and γ_3 is given by [Equation 151 on page 213](#). The definition of N^+ and N^- can be found in [Equation 152 on page 213](#).

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

The Hamiltonian part of the zeroth order in \mathbf{k} has three contributions:

- A diagonal entry from the band edge:

$$V^{4 \times 4} = \begin{pmatrix} E_c^{1 \times 1} & 0 \\ 0 & E_v^{3 \times 3} \end{pmatrix} \quad (184)$$

- A contribution from the strain ε :

$$H_{\text{strain}}^{4 \times 4} = \begin{pmatrix} a_c(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}) & 0 & 0 & 0 \\ 0 & l\varepsilon_{xx} + m(\varepsilon_{yy} + \varepsilon_{zz}) & n\varepsilon_{xy} & n\varepsilon_{xz} \\ 0 & n\varepsilon_{yx} & l\varepsilon_{yy} + m(\varepsilon_{xx} + \varepsilon_{zz}) & n\varepsilon_{yz} \\ 0 & n\varepsilon_{zx} & n\varepsilon_{zy} & l\varepsilon_{zz} + m(\varepsilon_{xx} + \varepsilon_{yy}) \end{pmatrix} \quad (185)$$

- From spin-orbit coupling. Before you can add the spin-orbit coupling, you must put all of the other terms together into:

$$H^{4 \times 4} = \begin{pmatrix} H_{cc}^{1 \times 1} & H_{cv}^{1 \times 3} \\ H_{vc}^{3 \times 1} & H_{vv}^{3 \times 3} \end{pmatrix} + V^{4 \times 4} + H_{\text{strain}}^{4 \times 4} \quad (186)$$

and finally write:

$$H^{8 \times 8} = \begin{pmatrix} H^{4 \times 4} & 0 \\ 0 & H^{4 \times 4} \end{pmatrix} + H_{\text{so}} \quad (187)$$

using a spin-dependent chemical basis system $|s, +\rangle$, $|s, -\rangle$, $|p_x, +\rangle$, $|p_x, -\rangle$, $|p_y, +\rangle$, $|p_y, -\rangle$, $|p_z, +\rangle$, and $|p_z, -\rangle$, where + and – denote the spin-up and spin-down states, respectively.

By transforming from the spin-dependent chemical basis states to the basis states of the total angular momentum [2], the spin-orbit coupling term H_{so} in Equation 187 becomes diagonal:

$$H^{8 \times 8} = AH^{8 \times 8}A^\dagger = A \begin{pmatrix} H^{4 \times 4} & 0 \\ 0 & H^{4 \times 4} \end{pmatrix} A^\dagger + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\Delta_{\text{so}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Delta_{\text{so}} \end{pmatrix} \quad (188)$$

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

The transformation matrix A in [Equation 188](#) is given by:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{2}{\sqrt{6}} & 0 & -\frac{1}{\sqrt{6}} & \frac{i}{\sqrt{6}} & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{3}} & 0 & \frac{1}{\sqrt{3}} & -\frac{i}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & \frac{i}{\sqrt{6}} & 0 & 0 & 0 & 0 & \frac{2}{\sqrt{6}} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{3}} & \frac{i}{\sqrt{3}} & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{3}} \end{pmatrix} \quad (189)$$

From the bulk Hamiltonian in [Equation 188](#), you obtain the Hamiltonian for the confined system by replacing the components of the k -vector with the respective derivatives, as described in [Equation 149 on page 212](#) and [Equation 153 on page 214](#).

Using the Eight-Band $k \cdot p$ Schrödinger Solver

You can use the `8kpValley` model described in [The 8kpValley Model on page 185](#) in a Schrödinger equation, which is associated with the carrier type. Using the keyword `eSchrodinger`, the electron subbands are calculated and, using the keyword `hSchrodinger`, the hole subbands are calculated.

For a self-consistent simulation where both the electron and the hole charges are computed from the eight-band $k \cdot p$ model, you must define two `8kpValley` models and use them with the respective Schrödinger equation object:

```
Physics material=GaAs ValleyModel=8kpValley name=Gamma1 \
degeneracy=1 useForHBulkDensity=0

Physics material=GaAs ValleyModel=8kpValley name=Gamma2 \
degeneracy=1 useForEBulkDensity=0

Physics nonlocal=NL1 hSchrodinger=8kp valleys=[list Gamma1] \
Nk=22 Kmax=0.3 iwSymmetry=AUTO nPhi=48 \
useKdependentWF=1 useNonsymBulkKPHamil=1

Physics nonlocal=NL1 eSchrodinger=8kp valleys=[list Gamma2] \
Nk=22 Kmax=0.3 iwSymmetry=AUTO nPhi=48 \
useKdependentWF=1 useNonsymBulkKPHamil=1
```

Chapter 13: Subband and Mobility Calculations

Calculating Subbands

As for the six-band $k \cdot p$ model, the `iwSymmetry=AUTO` functionality for detecting the symmetry of the k -space is supported by the eight-band $k \cdot p$ model.

If `useNonsymBulkKPHamil=0`, then the symmetric-operator ordering approach is used, which implies:

$$N^+ = N^- = \frac{N}{2} \quad (190)$$

Handling Subband Degeneracies

In calculations using the $k \cdot p$ band models, subband degeneracies often arise.

In some cases, these degeneracies can cause numeric issues that degrade accuracy or convergence. In these cases, it is often helpful to split the degeneracies slightly using the `degenSubbandEnergySplit` argument, which specifies a value (in eV) that is added to the diagonal of the $k \cdot p$ Hamiltonians to split degenerate subbands. By default, the value of this argument is zero. Effective splitting can be achieved with values as small as 1.0e-6 eV (see [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#)).

Interface Potential Spike

A delta-function spike in the confining band-edge potential can be specified at region interfaces using the `SpikeModel` argument in the `Physics` command (see [Physics for Interface Potential Spike Models on page 377](#)). The `SpikeModel` argument can specify multiple models, each model being associated with a unique list of valleys. The potential spike is included as part of the confining potential when solving one of the Schrödinger equations. The value of the spike potential is specified using the `value` argument in units of Vcm. For both the `eSchrodinger` and `hSchrodinger` solvers, a positive value of the spike potential creates a spike-like barrier at the interface, while a negative value creates a spike-like well at the interface.

Region interfaces can be specified in terms of region names using `regionInterface` or in terms of materials using `materialInterface`, using a slash (/) to separate regions (or materials) on either side of the interface.

For example, to specify a spike-like barrier of 2.0e-9 V cm at the region interface between regions `si1` and `si2` to be applied to the three default Δ -valleys in silicon, use the following command:

```
Physics regionInterface="si1/ox1" SpikeModel=ConstantSpike \
    value=2.0e-9 \
    valleys=[list Delta1 Delta2 Delta3] name=DeltaSpike
```

The only available model to use for `SpikeModel` is `ConstantSpike`, as shown in this example. This creates a constant spike-like potential along the entire specified interface. In addition, each `SpikeModel` can be given a unique name using `name`. This name can then be used in subsequent `Physics SpikeModel=String` commands to modify or remove an

Chapter 13: Subband and Mobility Calculations

Using Sentaurus Band Structure as an External Schrödinger Solver for Sentaurus Device

existing `SpikeModel`. In addition, this name is used to automatically generate a model for the spike potential that can be saved to a TDR file.

Visualizing the Interface Potential Spike

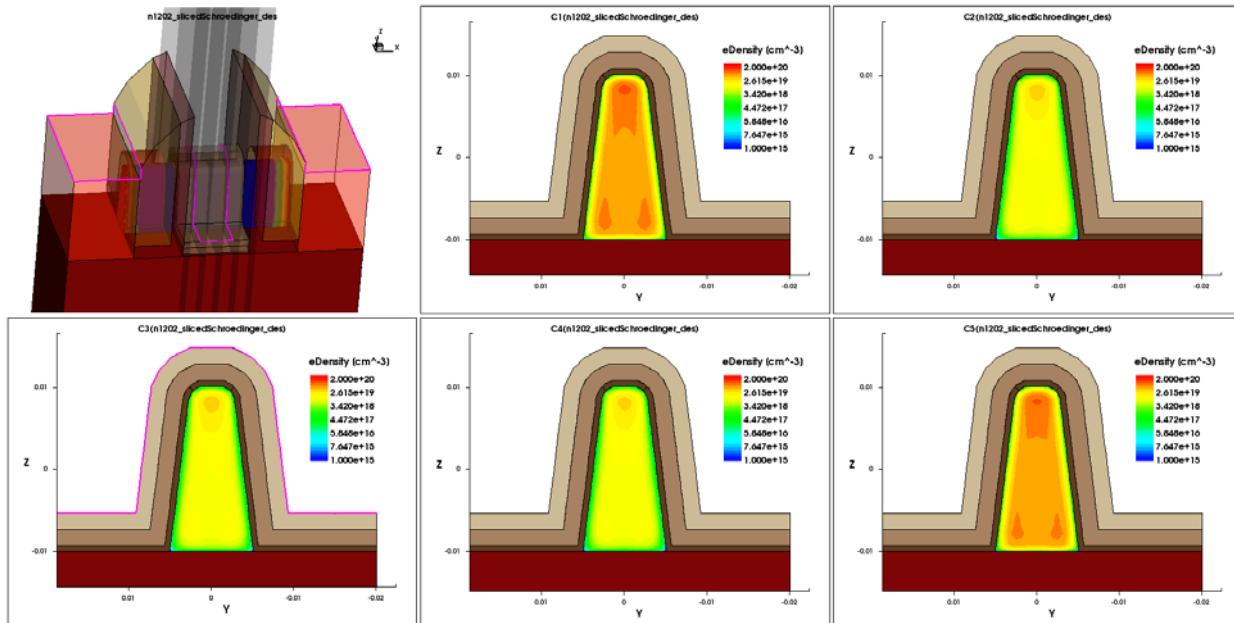
The spatial profile of a specified `SpikeModel` can be saved to a TDR file using the `Save` command. To save a particular `SpikeModel`, specify the `SpikeModel` name in the model list for the `models` argument. For example, to save the `SpikeModel` created in the previous example, with the name `DeltaSpike`, use the following command:

```
Save tdrFile=device.tdr models=[list DeltaSpike]
```

Using Sentaurus Band Structure as an External Schrödinger Solver for Sentaurus Device

Sentaurus Device and Sentaurus Band Structure can cooperate to run 3D device simulations with 2D Schrödinger confinement. In this cooperative simulation mode, Sentaurus Device handles the 3D device structure and solves a system of equations as specified in the `Solve` section of its command file. However, whenever Sentaurus Device needs to update the carrier densities, it calls Sentaurus Band Structure to provide quantum-mechanical density corrections on 2D slices normal to the main transport direction (see [Figure 49](#)).

Figure 49 Three-dimensional device structure with five 2D Schrödinger slices



Chapter 13: Subband and Mobility Calculations

Using Sentaurus Band Structure as an External Schrödinger Solver for Sentaurus Device

On the Sentaurus Device side, an `ExternalSchroedinger` block in the command file triggers the cooperative simulation mode (see *Sentaurus™ Device User Guide*, External 2D Schrödinger Solver). A Sentaurus Band Structure process can connect to a Sentaurus Device master process by executing the `ConnectToMasterProcess` Tcl command. For example:

```
set sliceIndex [lindex $argv 0]

set offsetVector [list 0.007 0.009 [expr (0.01 * $sliceIndex)]]

ConnectToMasterProcess connectionName=$SBandDeviceLink \
    slice=$sliceIndex timeout=600 \
    transform=[list {1 0 0} {0 1 0} {0 0 1}] \
    translate=$offsetVector
```

The `connectionName` must correspond to the name of the `ExternalSchroedinger` block in the Sentaurus Device command file, in this example:

```
ExternalSchroedinger "$SBandDeviceLink" {
    NumberOfSlices=2
    ...
}
```

Each Sentaurus Band Structure process is responsible for handling exactly one 2D slice, which is identified by a slice number between 0 and $N - 1$, where N is the number of slices requested in the `ExternalSchroedinger` block. The 3×3 matrix `transform` (units: dimensionless) and the 3D vector `translate` (units: μm) define an affine linear transformation that relates the 2D coordinates of the slice mesh to the 3D coordinates of the full device. Typically, the offset vector by which the mesh is displaced will be a function of the slice index.

The `timeout` (in seconds) specifies how long Sentaurus Band Structure will wait for Sentaurus Device to respond to connection requests and messages before terminating with an error message.

Beyond the `ConnectToMasterProcess` command, Sentaurus Band Structure behaves as usual: the `LoadDevice` command loads the (2D slice) structure, and a 2D Schrödinger simulator is set up in the same way as for a standalone 2D confined charge-density calculation.

When Sentaurus Device has finished its calculations, it returns control to Sentaurus Band Structure, and the usual execution of Tcl commands resumes.

Note:

The meshes of the 2D slices passed to Sentaurus Band Structure must satisfy the Delaunay criterion. A 2D cut through a 3D Delaunay mesh is usually *not* a Delaunay mesh.

Chapter 13: Subband and Mobility Calculations

Calculating Mobility

You can configure Sentaurus Device to start one Sentaurus Band Structure process per slice automatically. In this mode, the slice index is passed to Sentaurus Band Structure as a command-line argument (`[lindex $argv 0]`).

Calculating Mobility

The calculation of the inversion-layer mobility combines the subband dispersion computed by one of the Schrödinger solvers with a set of user-defined scattering models within either the Kubo–Greenwood formalism or using a direct solution of the linear Boltzmann transport equation. This provides a general microscopic approach for computing any component of the mobility tensor for arbitrary strain, orientation, and temperature.

Kubo–Greenwood Formalism

A linearization of the Boltzmann transport equation in the driving electric field gives the following Kubo–Greenwood expression for the ij component of the low-field mobility tensor in subband v :

$$\mu_{ij}^v = \frac{e}{\hbar^2 k_B T N_v} \frac{g_v}{(2\pi)^d} \int \frac{d\mathbf{k}}{\tau_i^v \partial E_v \partial k_j} f_0(E_v) [1 - f_0(E_v)] \quad (191)$$

where:

- g_v is the degeneracy.
- N_v is the subband inversion carrier density.
- τ_i^v is the total momentum relaxation time for subband v in direction (i).
- E_v is the subband energy dispersion.
- f_0 is the equilibrium Fermi–Dirac distribution function.
- d is the dimension of \mathbf{k} -space.

The total mobility is then given by a weighted average over all subband mobilities:

$$\mu_{ij}^{\text{Tot}} = \frac{1}{N_{\text{Tot}}} \sum_v N_v \mu_{ij}^v \quad (192)$$

where N_{Tot} is the total inversion sheet charge density.

Chapter 13: Subband and Mobility Calculations

Calculating Mobility

Inverse Momentum Relaxation Time

The inverse momentum relaxation time (IMRT) for a given scattering mechanism for transport along the i -th axis, for example, is given by an integral over final states by:

$$\begin{aligned}\frac{1}{\tau_i^v(\mathbf{k})} &= \sum_{v'} \int \frac{d\mathbf{k}'}{(2\pi)^d} S_{vk, v'k'} \times \Phi_i(v\mathbf{k}, v'\mathbf{k}') \\ &= \frac{2\pi}{\hbar} \sum_{v'} \int \frac{d\mathbf{k}'}{(2\pi)^d} |M_{vk, v'k'}|^2 \delta(E_{v'}(\mathbf{k}') - E_v(\mathbf{k}) \mp \hbar\omega) \times \Phi_i(v\mathbf{k}, v'\mathbf{k}')\end{aligned}\quad (193)$$

where $S_{vk, v'k'}$ is the transition rate between the initial wavevector \mathbf{k} in subband v and the final wavevector \mathbf{k}' in subband v' . The matrix element for scattering between initial and final states is denoted by $M_{vk, v'k'}$ and $E_v(\mathbf{k})$ is the subband dispersion.

For inelastic phonon processes, the phonon dispersion is assumed to be constant with energy $\hbar\omega$. The upper sign in the energy-conserving delta function refers to absorption, while the lower sign refers to emission. The Φ_i factor is the *momentum relaxation factor* and takes several forms depending on various approximations and assumptions [15]. Each scattering mechanism implements a particular model for this term or provides a parameter for choosing different models. The total IMRT is computed by summing the IMRT for each of the individual scattering mechanisms.

Linear Boltzmann Transport Equation

In the limit of the vanishing driving field, F , the carrier distribution function can be written as a small perturbation, $g_v(\mathbf{k})$, to the equilibrium Fermi–Dirac distribution, f_0 :

$$f_v(\mathbf{k}) = f_0(E_v) + F \cdot g_v(\mathbf{k}) \quad (194)$$

Inserting this form of the distribution function into the Boltzmann transport equation (BTE) and only keeping terms that are linear in the perturbation, the following linear BTE can be solved to give $g_v(\mathbf{k})$:

$$\sum_{v', k'} S_{v, v'k'}(\mathbf{k}, \mathbf{k}') \left[g_v(\mathbf{k}) \frac{1 - f_0(E_{v'}(\mathbf{k}'))}{1 - f_0(E_v(\mathbf{k}))} - g_{v'}(\mathbf{k}') \frac{f_0(E_{v'}(\mathbf{k}'))}{f_0(E_v(\mathbf{k}'))} \right] = -\frac{1}{\hbar} \frac{\partial f_0(E_v(\mathbf{k}))}{\partial \mathbf{k}} \quad (195)$$

where $S_{v, v'k'}(\mathbf{k}, \mathbf{k}')$ is the total transition rate from the initial state (v, \mathbf{k}) to the final state (v', \mathbf{k}') . After $g_v(\mathbf{k})$ has been found, the mobility in subband v can be computed using:

$$\mu_v = \frac{\tilde{g}_v \int \frac{d\mathbf{k}}{(2\pi)^d} \tilde{g}_v(\mathbf{k}) v_v(\mathbf{k}) \cdot \hat{\beta}}{N_v} \quad (196)$$

where:

- \tilde{g}_v is the valley and spin degeneracy.
- v_v is the group velocity.

Chapter 13: Subband and Mobility Calculations

Calculating Mobility

- β is the direction.
- N_v is the inversion charge in subband v .
- d is the k -space dimension, either one or two.

Mobility Calculators

Within Sentaurus Band Structure, the evaluation of IMRTs and the Kubo–Greenwood integral ([Equation 191](#)) or the solution of the linear BTE is handled by a mobility calculator defined on a nonlocal line or nonlocal area.

Two types of mobility calculator are available: `KGFromK` and `LinearBTE`. The `KGFromK` calculator evaluates [Equation 191](#) and the `LinearBTE` calculator evaluates [Equation 196](#) using optimized quadrature rules over the k -space grid defined in the `Physics` command.

For `KGFromK`, IMRTs are evaluated on the same k -space as the Kubo–Greenwood integral. For `LinearBTE`, [Equation 195](#) is solved on the same k -space grid as the mobility integral.

The mobility calculator for a nonlocal line or nonlocal area obtains the subband dispersions and wavefunctions from the Schrödinger equation assigned to the same nonlocal line or nonlocal area. The mobility calculator obtains the scattering models used in the evaluation of IMRTs from models defined by the `Physics ScatteringModel=String` command. Typically, multiple scattering models are defined.

Note:

Mobility can be computed from the `Parabolic` and $k \cdot p$ Schrödinger solvers (`2kp`, `3kp`, `6kp`, and `8kp`).

k-Space Grid

The k -space grid used to evaluate the mobility must be specified independently of the k -space grid used to calculate the subband dispersion from the Schrödinger equation. The grid spacing of the specified k -grid and φ -grid must be sufficiently fine to resolve the integrand in [Equation 191](#).

Using Mobility Calculators

Using a mobility calculator involves the following main steps:

1. Specify a set of scattering models (see [Scattering Models on page 231](#)).
2. Specify a mobility calculator and its parameters for a nonlocal line (see [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#)).

Chapter 13: Subband and Mobility Calculations

Calculating Mobility

A mobility calculator for electrons, for example, can be specified for a nonlocal line using a Physics command of the form:

```
Physics nonlocal=NL1 eMobilityCalculator=KGFromK Nk=128 \
Kmax=0.3 Nphi=32
```

This command specifies to use the `KGFromK` mobility calculator for electron mobility calculations on the nonlocal line `NL1`. Parameters for the polar grid used for the mobility evaluation are also specified.

3. Use the `ComputeMobility` command to compute the mobility (see [ComputeMobility on page 356](#)).

For example, to compute the `xx` component of the mobility tensor, use the command:

```
set MobilityXX [ComputeMobility xx]
```

This command returns the computed mobility value to the Tcl variable `MobilityXX`. In addition, the computed value is added automatically to the bias log file. To also add the mobilities in each valley to the bias log file, specify the `writeMobilityPerValley` option in the `ComputeMobility` command.

Visualization Quantities

The k -space quantities related to the calculation of mobility can be saved to a TDR file by using the `SaveK` command (see [SaveK on page 385](#)). For both mobility calculators, the IMRT can be saved using a model name of the form `valley-name_subband-index_IMRT`. For example: `Delta1_0_IMRT`

When using the `LinearBTE` calculator, the solution to the linear BTE can also be saved using the `LinearBTE` model name. For example: `Delta1_0_LinearBTE`

Coulomb Green's Function

The solution of the Green's function for the Poisson equation, known here as the Coulomb Green's function, is used in two aspects of mobility calculations:

- First, it computes the Lindhard dielectric screening function, which can be used to screen certain scattering mechanisms.
- Second, it computes the unscreened scattering matrix element for Coulomb scattering.

The Coulomb Green's function for a point charge at x_0 and an in-plane wavevector q is the solution of:

$$[\nabla \bullet \epsilon \nabla - \epsilon \cdot q^2] G(q, x, x_0) = -e \cdot \delta(x - x_0) \quad (197)$$

Chapter 13: Subband and Mobility Calculations

Calculating Mobility

where ϵ is the dielectric permittivity. In one dimension, two different sets of boundary conditions can be specified for the solution of this equation. For `null` boundary conditions, G is set to zero at metal contacts. This is important in high- κ /metal gate devices in which the metal gate provides additional screening. Another option is to use `mixed` boundary conditions, which mimic semi-infinite layers on the top and bottom of the device. This is a common approximation used for conventional gate stacks.

For this boundary condition, G must fulfill:

$$\frac{dG}{dz} = q \cdot G \quad (198)$$

The Coulomb Green's function is computed automatically as needed. The required boundary conditions can be set using the parameter `BC` for the model keyword `CoulombGreensFunction`. For example:

```
Physics material=all CoulombGreensFunction BC=mixed
```

By default in one dimension, `mixed` boundary conditions are used. For 2D structures, `null` boundary conditions are always used.

Spatial Mobility Profile

The mobility computed using the `ComputeMobility` command represents an effective mobility value for all subbands over the entire nonlocal line or nonlocal area. This mobility value has no explicit spatial dependency.

However, by weighting each subband mobility value by the subband inversion charge and the wavefunction norm, a spatial profile for the mobility can be computed [16]:

$$\mu(x) = \frac{\sum_i |\psi_i(x)|^2 n_i \mu_i}{n(x)} \quad (199)$$

where:

- ψ_i , n_i , and μ_i are the wavefunction, the inversion charge, and the mobility for subband i , respectively.
- $n(x)$ is the local electron density.

This spatial mobility profile can be saved to a TDR file using the model keywords `eMobility` and `hMobility`, for electrons and holes, respectively, in the `Save` command. For example, to save the electron mobility spatial profile, use:

```
Save tdrFile=device_out.tdr models=[list eMobility]
```

Scattering Models

Scattering models compute the transition rate for carriers to scatter from initial to final states due to various scattering mechanisms:

- Phonon Scattering
 - Alloy Disorder Scattering
 - Surface Roughness Scattering
 - Coulomb Scattering
-

Common Parameters

Several common parameters are used by different scattering models, including the list of allowed valleys involved in the transition, the type of transition, the approach for computing the momentum transfer, and, for some models, the approach for computing dielectric screening.

Valleys

All scattering models require a description of the allowed transitions between valleys. This description consists of a list of the allowed initial/final valley pairs, as given by the `valleys` argument of the `Physics` command, and a description of the type of transitions allowed, as given by `transitionType`. Each initial/final valley pair is entered as a Tcl list, for example, `{Delta1 Delta2}`. This indicates that transitions from the `Delta1` valley to the `Delta2` valley are allowed.

To allow transitions to occur between multiple valley pairs, a list of these valley pairs can be specified.

Note:

Reciprocal transitions must be listed explicitly. For example, to allow transitions also from `Delta2` to `Delta1` in this example, a list of the form `{Delta2 Delta1}` must be specified.

Transition Type

The type of transition must be specified using the `transitionType` argument. In general, transitions can occur in one of the following ways:

- Between states in the same subband (*intra-subband*)
- Between states that lie in the same subband or different subbands but are still in the same valley (*intravalley*)

- Between subbands that lie in different valleys (*intervalley*)

For intervalley transitions, two different types of transition are distinguished based on the equivalence of the valleys. The transition type partly determines the degeneracy of the final state. [Table 38](#) summarizes the allowed types and final-state degeneracy.

Table 38 Allowed transition types, with degeneracy of initial and final valleys denoted by g_v and g'_v , respectively

transitionType	value	Final-state degeneracy	Description
gIntervalley		$g_v - 1$	Only transitions between equivalent valleys are allowed
Intervalley		g'_v	Only transitions between non-equivalent valleys are allowed
Intrasubband	1		Only transitions within the initial subband are allowed
Intravalley	1		Only transitions within subbands from the same valley are allowed

For inelastic phonon scattering, two additional common parameters must be specified: the phonon energy and the inelastic type. As specified by `inelasticType`, the inelastic type indicates whether phonon absorption or emission occurs during the transition. The phonon energy dispersion is assumed to be constant and is specified using `hbarOmega`.

Momentum Relaxation Factor

For the Kubo–Greenwood approach, calculating the IMRT from the transition rate involves a factor that models the momentum transfer during scattering, that is, the momentum relaxation factor. This factor is given by:

$$\Phi(v\mathbf{k}, v'\mathbf{k}') = \frac{1 - f_0[E_v(\mathbf{k}')] }{1 - f_0[E_v(\mathbf{k})]} \cdot \{\text{mrFactor model}\} \quad (200)$$

where f_0 is the equilibrium Fermi–Dirac distribution function, and the `mrFactor` model is selected according to [Table 39](#). For isotropic scattering models, such as nonpolar phonon scattering, this factor is computed automatically. For anisotropic mechanisms, such as surface roughness and Coulomb scattering, different `mrFactor` models can be selected as summarized in [Table 39](#). The required model is selected with the `mrFactor` argument. For 1D devices, by default, `mrFactor=2` is used. When using the linear BTE approach, `mrFactor` is ignored.

Chapter 13: Subband and Mobility Calculations

Scattering Models

Table 39 Models for mrFactor

Model number	Equation for 1D simulation	Equation for 2D simulation	Description
1	1	1	Treat as isotropic
2	$1 - \cos(\theta' - \theta)$	$1 - \text{sign}(k \cdot k')$	Consider the change in angles of the wavevectors
3	$1 - \frac{\mathbf{k} \cdot \mathbf{k}'}{\mathbf{k} \cdot \mathbf{k}}$	$1 - \frac{k'}{k}$	Consider the change in wavevectors
4	$1 - \frac{\mathbf{v} \cdot \mathbf{v}'}{\mathbf{v} \cdot \mathbf{v}}$	$1 - \frac{v'}{v}$	Consider the change in group velocities
5	$1 - \frac{k'_x}{k_x}$	Same as 4	Consider the change in the x-component of the wavevectors

Screening

Some scattering mechanisms, namely, surface roughness, Coulomb, alloy, and polar-optical phonon scattering, can be screened by carriers in inversion layers. Different approaches for computing this dielectric screening are provided.

Tensor-Based Lindhard Screening

In the tensor-based approach to dielectric screening, the screened matrix elements, $M_{vv'}^{\text{scr}}$, for a particular scattering mechanism are related to the unscreened matrix elements, $M_{\mu\mu'}^{\text{scr}}$, using the fourth-rank dielectric tensor $\epsilon_{\mu\mu'}^{vv'}(q)$:

$$M_{\mu\mu'}(q) = \sum_{vv'} \epsilon_{\mu\mu'}^{vv'}(q) M_{vv'}^{\text{scr}}(q) \quad (201)$$

where q is the magnitude of the wavevector change between the initial and final states. Given the unscreened matrix elements and the dielectric tensor, the linear system above can be inverted to give the screened matrix elements. The dielectric tensor itself is given by:

$$\epsilon_{\mu\mu'}^{vv'}(q) = \delta_{\mu\nu} \delta_{\mu'\nu'} + \Pi_{vv'}(q) F_{\mu\mu'}^{vv'}(q) \quad (202)$$

where:

- $\mu\mu'$ are subbands from the same valley, and vv' are subbands from the same valley.
- The polarization is given by:

$$\Pi_{vv'}(q) = \frac{g_v}{(2\pi)^d} \int d\mathbf{k} \frac{f_0[E_v(\mathbf{k})] - f_0[E_v(\mathbf{k} + \mathbf{q})]}{E_v(\mathbf{k} + \mathbf{q}) - E_v(\mathbf{k})} \quad (203)$$

Chapter 13: Subband and Mobility Calculations

Scattering Models

where g_v is the degeneracy, E_v is the subband dispersion, and f_0 is the Fermi–Dirac distribution function.

- The screening form-factor is given by:

$$F_{\mu\mu'}^{vv'}(q) = \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} dx' \vec{\psi}_{\mu}(x) \cdot \vec{\psi}_{\mu'}^{\dagger}(x) G(q, x, x') \vec{\psi}_v(x') \cdot \vec{\psi}_{v'}^{\dagger}(x') \quad (204)$$

where ψ_v are the wavefunctions, and G is the Coulomb Green's function given by [Equation 197](#).

Treating all of the inter-subband contributions to the dielectric tensor is time-consuming. Whether a scattering transition between two subbands, μ and μ' , is screened is determined by the `tdfDegenTol` argument in the `Physics nonlocal=String` command when specifying the mobility calculator (see [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#)).

For two subbands that have a difference in minimum energies less than this tolerance, the inter-subband transition is screened. If the energy difference is greater than `tdfDegenTol`, then the scattering transition is treated as unscreened. By default, `tdfDegenTol` is set to `-1`, which indicates a special algorithm is used to identify nearly degenerate subband pairs. To set a specific tolerance value, use a positive value such as `1.0e-4 eV`.

Note:

Ensure that you always use the tensor-based approach to dielectric screening for double-gate structures to properly treat nearly degenerate subbands.

Scalar Lindhard Screening

For bulk or single-gate MOS capacitor structures, some simplifying approximations can be used to reduce the complexity of the dielectric screening calculation. In this scalar Lindhard dielectric function approach [\[17\]](#), all nondegenerate inter-subband matrix elements remain unscreened, while the intra-subband matrix elements are screened using:

$$|M_{vk, vk}|^2 = \frac{|M_{vk, vk}^{\text{unscr}}|^2}{\epsilon^2(q)} \quad (205)$$

where a scalar version of the dielectric function, $\epsilon(q)$, is given by:

$$\epsilon(q) = 1 + \sum_v F_{vv}(q) \Pi_{vv}(q) \quad (206)$$

where:

- ϵ is the permittivity of the semiconductor.
- F_{vv} is the screening form-factor ([Equation 204](#)).
- Π_{vv} is the polarization ([Equation 203](#)).

Chapter 13: Subband and Mobility Calculations

Scattering Models

To treat cases with nearly degenerate subbands, you set the `sdfDegenTol` argument in the `Physics nonlocal=String` command when specifying a mobility calculator (see [Physics of Nonlocal Lines or Nonlocal Areas on page 379](#)). This argument determines whether two subbands will be considered degenerate based on the difference in their minimum subband energies. Setting `sdfDegenTol` to a negative value means degenerate subbands are ignored when computing and applying the scalar dielectric function.

Thomas–Fermi Screening for Semiconductors

In this simple screening model, the complexity is even more reduced compared to scalar Lindhard screening. The Thomas–Fermi screening model for semiconductors can be derived from the more general Lindhard expression. In particular, the Thomas–Fermi expression is the limit of the Lindhard formula when the length scale of interest is much larger than the Fermi screening length, that is, the long-distance limit. The same approach as for the scalar Lindhard screening model ([Equation 205](#)) is used to screen the matrix element but with the modified dielectric function:

$$\varepsilon(q) = 1.0 + \left(\frac{1.0}{\epsilon_s - 1.0} + \frac{q^2}{k_f^2} \right)^{-1} \quad (207)$$

where:

- The Fermi radius is $k_f = \frac{a_0}{2\pi} (3\pi^2 n_{\text{avg}})^{\frac{1}{3}}$, in reduced units $2\pi/a_0$.
- ϵ_s is the relative permittivity of the semiconductor.
- n_{avg} is the averaged density [cm^{-3}] in the semiconductor; $n_{\text{avg}} = \frac{1}{V} \int d\mathbf{r} n(\mathbf{r})$.

For larger device structures, it is expected that the results of the simple Thomas–Fermi screening model and the more advanced Lindhard screening agree well, but the simulation time can be reduced significantly.

Using Screening

Screening can be activated for those scattering mechanisms that support it by using the `screening` argument. In general, the following options are available, although some scattering models allow only the scalar approach:

- `screening=Lindhard` selects the scalar Lindhard dielectric function.
- `screening=LindhardTDF` selects the tensor-based Lindhard dielectric function.
- `screening=ThomasFermi` selects the scalar Thomas–Fermi dielectric function.
- `screening=none` deactivates screening.

Scattering Model Availability by Device Dimension

Table 40 lists the availability of scattering models depending on the dimensionality of the simulated device.

Table 40 Available scattering models by device dimension

Scattering mechanism	Model name	Available for 1D devices	Available for 2D devices	Screening options (default: bold)
Phonon scattering				
Elastic acoustic phonon	ElasticAcousticPhonon	Yes	Yes	—
Ohashi acoustic phonon	OhashiAcousticPhonon	Yes	Yes	—
Inelastic phonon	InelasticPhonon	Yes	Yes	—
Tensor acoustic phonon	TensorElasticAcousticPhonon	Yes	Yes	—
Tensor optical phonon	TensorInelasticPhonon	Yes	Yes	—
Polar-optical phonon	POP	Yes	Yes	none Lindhard LindhardTDF ThomasFermi
Remote soft-optical phonon	RemoteSOPhonon	Yes	No	none Lindhard LindhardTDF ThomasFermi
Alloy disorder scattering				
Alloy disorder	ElasticAlloy	Yes	Yes	none Lindhard LindhardTDF ThomasFermi
Old alloy disorder	OldElasticAlloy	Yes	Yes	—

Chapter 13: Subband and Mobility Calculations

Scattering Models

Table 40 Available scattering models by device dimension (Continued)

Scattering mechanism	Model name	Available for 1D devices	Available for 2D devices	Screening options (default: bold)
Surface roughness scattering				
Isotropic Prange–Nee	IsotropicPrangeNee	Yes	No	none Lindhard
Surface roughness scattering (2D)	SRFor2D	No	Yes	none Lindhard LindhardTDF ThomasFermi
Surface roughness scattering from 6kp Schrödinger (1D)	SRFrom6kp or SRFor1D	Yes	No	none Lindhard LindhardTDF ThomasFermi
Surface roughness scattering from Parabolic Schrödinger (1D)	SRFromParabolic	Yes	No	none Lindhard LindhardTDF
Coulomb scattering				
Coulomb	Coulomb	Yes	Yes	none Lindhard LindhardTDF ThomasFermi

Phonon Scattering

Models for acoustic and inelastic phonon scattering for a heterostructure channel containing multi-SiGe layers are available.

Acoustic Phonon Scattering

Acoustic phonon scattering is treated as an elastic isotropic process within the energy equipartition approximation.

Chapter 13: Subband and Mobility Calculations

Scattering Models

As an example for SiGe, for both electrons and holes, the matrix element for the isotropic scattering is:

$$|M_{vk, v'k}|^2 = \sum_i \left\{ (1 - x_i) \frac{k_B T}{\rho^{Si} (u_l^{Si})^2} (D_{ac}^{Si})^2 + x_i \frac{k_B T}{\rho^{Ge} (u_l^{Ge})^2} (D_{ac}^{Ge})^2 \right\} F_{i, v, v'} \quad (208)$$

where:

- i is the layer index.
- x_i is the mole fraction of layer i .
- ρ^{Si} (ρ^{Ge}) is the mass density of Si (Ge).
- u_l^{Si} (u_l^{Ge}) is the longitudinal sound velocity of Si (Ge).
- D_{ac}^{Si} (D_{ac}^{Ge}) is the effective deformation potential of Si (Ge), combining multiple phonon branches and angular averaging.
- F is the wavefunction overlap form-factor given by:

$$F_{i, v, v'} = \int_{\text{i-th layer}} dr \left| \vec{\Psi}_v(r; \mathbf{k}) \cdot \vec{\Psi}_{v'}(r; \mathbf{k}) \right|^2 \quad (209)$$

where Ψ_v is the wavefunction for subband v . The integral is taken over layer i only, implying a layer separation of the form factor. The vector notation indicates that the wavefunction can, in general, have multiple components. Acoustic phonon scattering is typically limited to intravalley transitions. Because this mechanism is elastic and the matrix element is isotropic, the momentum relaxation factor is simply 1.

This model is specified with the name `ElasticAcousticPhonon`. You must specify separately in the Tcl file the Si and Ge parts within the brackets {} on the RHS of [Equation 208](#). For each part, [Table 41](#) lists the user-modifiable parameters for this model.

Table 41 Parameters for acoustic phonon scattering

Equation parameter	Parameter name	Unit	Description
$\sqrt{1 - x_i} D_{ac}^{Si}$ or $\sqrt{x_i} D_{ac}^{Ge}$	Dac	eV	Deformation potential
ρ^{Si} or ρ^{Ge}	density	g/cm ³	Mass density
u_l^{Si} or u_l^{Ge}	ul	cm/s	Longitudinal sound velocity

Ohashi Acoustic Phonon Scattering

The Ohashi acoustic phonon scattering model captures the spatial dependence of the acoustic deformation potential, D_{ac} , as reported in the literature [18][19]. It follows the typical approach for acoustic phonons and is treated as an elastic isotropic process within the energy equipartition approximation.

The matrix element for this model follows that of [Equation 208](#), with the expression for D_{ac} replaced with the Ohashi form. The Ohashi form for the acoustic deformation potential is calculated as follows:

$$D_{ac}^{\text{eff}} = \sum_{v,i} \frac{N_{v,i}}{N_s} \int D_{ac}(z) |\xi_{v,i}(z)|^2 dz \quad (210)$$

where:

- For a planar MOS structure, $D_{ac}(z) = \Delta D_{ac} \exp\left(-\frac{z}{L_{ac}}\right) + D_{ac}^{\min}$.
- For an SOI or a double-gate structure,

$$D_{ac}(z) = \left(\Delta D_{ac} \exp\left(-\frac{z}{L_{ac}}\right) + \Delta D_{ac} \exp\left(-\frac{T_{\text{body}} - z}{L_{ac}}\right) \right) + D_{ac}^{\min}.$$

- $N_{v,i}$ is the surface carrier density in the i -th subband in the v -th valley.
- N_s is the integrated carrier density.
- $\xi_{v,i}$ is the subband wavefunction in the i -th subband in the v -th valley.
- z is the distance from the interface.
- $\Delta D_{ac} = D_{ac}^{\max} - D_{ac}^{\min}$ is the difference between the minimum, or bulk, D_{ac} and the value at the interface.
- L_{ac} is the decay length of D_{ac} .
- T_{body} is the body thickness of an SOI or a double-gate structure. In the expression, the body thickness is extracted automatically from the geometry of the device.

Note:

When you specify the model for a 2D structure, the model applies the correction to the nearest interface only, replacing z with the distance to the nearest interface computed as $z' = \sqrt{x^2 + y^2}$.

Chapter 13: Subband and Mobility Calculations

Scattering Models

This model is specified with the name `OhashiAcousticPhonon`. [Table 42](#) lists the user-modifiable parameters for this model.

Table 42 Parameters for Ohashi acoustic phonon scattering

Equation parameter	Parameter name	Unit	Description
D_{ac}^{\min}	minDac	eV	Minimum, or bulk, deformation potential
D_{ac}^{\max}	maxDac	eV	Maximum deformation potential
L_{ac}	Lac	μm	Decay length for deformation potential
ρ	density	g/cm ³	Mass density
u_l	ul	cm/s	Longitudinal sound velocity

Inelastic Phonon Scattering

Inelastic phonon scattering is treated in an isotropic approximation and assumes constant phonon energy dispersion with energy $\hbar\omega_{Si}^{Si}$ and $\hbar\omega_{Ge}^{Ge}$.

The matrix element for SiGe, as an example, is given by:

$$|M_{vk, v'k}|^2 = \sum_i \left\{ (1 - x_i) |\tilde{M}_{vk, v'k}^{Si}|^2 + x_i |\tilde{M}_{vk, v'k}^{Ge}|^2 \right\} F_{i, v, v'} \quad (211)$$

where:

$$|\tilde{M}_{vk, v'k}^{Si}|^2 = \frac{\hbar^2 (DtK^{Si})^2}{2\rho_{Si} \hbar\omega_{Si}} \left[N(\hbar\omega_{Si}) + \frac{1}{2} \mp \frac{1}{2} \right] \quad (212)$$

$$|\tilde{M}_{vk, v'k}^{Ge}|^2 = \frac{\hbar^2 (DtK^{Ge})^2}{2\rho_{Ge} \hbar\omega_{Ge}} \left[N(\hbar\omega_{Ge}) + \frac{1}{2} \mp \frac{1}{2} \right] \quad (213)$$

Here, $N(\hbar\omega) = 1/(e^{\hbar\omega/k_B T} - 1)$ is the phonon occupancy. The upper and lower signs in [Equation 193](#) and [Equation 212](#) refer to absorption and emission processes, respectively.

This model is specified with the name `InelasticPhonon`. You must specify separately in the Tcl file the Si and Ge parts within the braces {} on the RHS of [Equation 211](#). For each part, the user-modifiable parameters are listed in [Table 43](#). Note that the scattering models for emission and absorption processes must be specified separately.

Table 43 Parameters for inelastic phonon scattering

Equation parameter	Parameter name	Unit	Description
$\sqrt{1 - x_t} D_t K^{\text{Si}}$ or $\sqrt{x_t} D_t K^{\text{Ge}}$	DtK	eV/cm	Deformation potential
ρ^{Si} or ρ^{Ge}	density	g/cm ³	Mass density
$\hbar\omega^{\text{Si}}$ or $\hbar\omega^{\text{Ge}}$	hbarOmega	eV	Phonon energy

Evaluating the Wavefunction Overlap Form-Factor

For the $6kp$ Schrödinger solver, the wavefunction overlap form-factor ([Equation 209](#)) is, in general, dependent on the wavevector of the initial and final states. This k -dependency can be treated in one of two ways as set by the `kdepFF` argument.

When `kdepFF=0`, the k -dependency of the form factor is ignored, and the form factor is always evaluated at the center of the k -space

grid used for the mobility calculation. For the $6kp$ Schrödinger solver, the overlap form-factor between pairs of subbands that are degenerate at the Γ -point is often very small. This suppresses inter-subband phonon scattering between these subband pairs.

To avoid this suppression, use the `useDonettiOverlapFix` argument in the `Math` command to replace the evaluation of the overlap form-factor between these degenerate subbands with the intra-subband form-factor from the subband of the initial state [\[17\]](#). This modification to the form-factor calculation is not activated by default. To activate this modification, set `useDonettiOverlapFix=1`.

When `kdepFF=1`, the dependency of the form factor on the initial wavevector is treated by evaluating the initial and final wavefunctions in [Equation 209](#) at the initial wavevector.

Tensor Acoustic Phonon Scattering

The general approach for acoustic phonon scattering makes an approximation to the deformation potential that results in a scalar form. For simulations that use the $k \cdot p$ band structure, there is an alternative that is based on the tensor form of Bir–Pikus deformation potential theory. The scattering model is still treated as an elastic isotropic process within the energy equipartition approximation.

Note:

If you use this model for band structures other than $k \cdot p$, then Sentaurus Band Structure will generate a runtime error message.

Chapter 13: Subband and Mobility Calculations

Scattering Models

The matrix element for this model is:

$$|M_{v'k', v k}|^2 = s \cdot \sum_l \frac{k_B T}{\rho u_l^2} F_{l, v', v} \quad (214)$$

where:

- l is the layer index.
- ρ is the mass density.
- u_l is the longitudinal sound velocity.
- s is a unitless factor that scales the overall squared matrix element. It is specified by using `scaleFactor` and is set to 1.0 by default.
- F is the wavefunction overlap form-factor, containing the deformation potential operator, and is given by:

$$F_{l, v', v} = \left| \int \Psi_{v'}^*(\mathbf{r}, \mathbf{k}') \cdot \mathbf{D}_{ac} \cdot \Psi_v(\mathbf{r}, \mathbf{k}) d\mathbf{r} \right|^2 \quad (215)$$

For the six-band $\mathbf{k} \cdot \mathbf{p}$ approach, \mathbf{D}_{ac} is given by:

$$\begin{aligned} \mathbf{D}_{ac} &= \begin{bmatrix} [\mathbf{D}'_{ac}] & 0 \\ 0 & [\mathbf{D}'_{ac}] \end{bmatrix} \\ \mathbf{D}'_{ac} &= \begin{bmatrix} l+2m & n & n \\ n & l+2m & n \\ n & n & l+2m \end{bmatrix} \end{aligned} \quad (216)$$

where l , m , and n are the valence-band deformation potentials related to the widely used a , b , and d by:

$$l = a + 2b \quad m = a - b \quad n = \sqrt{3}d \quad (217)$$

This model is specified with the name `TensorElasticAcousticPhonon`. [Table 44](#) lists the user-modifiable parameters for this model.

Table 44 Parameters for tensor acoustic phonon scattering

Equation parameter	Parameter name	Unit	Description
s	<code>scaleFactor</code>	—	Unitless scaling factor
ρ	<code>density</code>	g/cm ³	Mass density
u_l	<code>ul</code>	cm/s	Longitudinal sound velocity

Tensor Optical Phonon Scattering

Following the tensor acoustic phonon scattering model, optical phonon scattering models also generally make a scalar approximation to the deformation potential. With $\mathbf{k} \cdot \mathbf{p}$ bands, it is possible to reintroduce the tensor form for the optical deformation potential. The other usual optical phonon assumptions apply, that is, a constant phonon energy dispersion.

The matrix element for this model is given by:

$$|M_{\text{op};v,v'}(\mathbf{k}, \mathbf{k}')|^2 = s \cdot \sum_l \frac{\hbar^2 (D_l K)^2}{2\rho \hbar \omega_0 \Omega} \left(N(\hbar \omega_0) + \frac{1}{2} \mp \frac{1}{2} \right) F_{l,v',v} \quad (218)$$

where:

- l is the layer index.
- ρ is the mass density.
- $D_l K$ is the deformation potential.
- $\hbar \omega_0$ is the constant phonon energy.
- $N(\hbar \omega_0) = 1/(e^{\hbar \omega_0} - 1)$ is the phonon occupancy.
- s is a unitless factor that scales the overall squared matrix element. It is specified by using `scaleFactor` and is set to 1.0 by default.
- F is the wavefunction overlap form-factor, containing the optical permutation operator, and is given by:

$$F_{l,v',v} = \sum_{l=1}^3 \left| \int \Psi_{v'}^*(\mathbf{r}, \mathbf{k}') \cdot \mathbf{A}_l \cdot \Psi_v(\mathbf{r}, \mathbf{k}) d\mathbf{r} \right|^2 \quad (219)$$

For the six-band $\mathbf{k} \cdot \mathbf{p}$ approach, \mathbf{A}_l is given by:

$$\mathbf{A}_l = \begin{bmatrix} [A'_l] & 0 \\ 0 & [A'_l] \end{bmatrix} \quad (220)$$

and:

$$\mathbf{A}'_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \mathbf{A}'_y = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \mathbf{A}'_z = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (221)$$

Chapter 13: Subband and Mobility Calculations

Scattering Models

This model is specified with the name `TensorInelasticPhonon`. [Table 45](#) lists the user-modifiable parameters for this model.

Table 45 Parameters for tensor optical phonon scattering

Equation parameter	Parameter name	Unit	Description
s	scaleFactor	—	Unitless scaling factor
$D_t K$	DtK	eV/cm	Deformation potential
ρ	density	g/cm ³	Mass density
$\hbar\omega_0$	hbarOmega	eV	Phonon energy

Polar-Optical Phonon Scattering

Polar-optical phonon (POP) scattering is an important scattering mechanism in materials with polar bonds such as III–Vs. POP scattering is treated as an intravalley, anisotropic scattering mechanism with a constant phonon-energy dispersion with energy $\hbar\omega$.

The unscreened matrix element is given by:

$$|\tilde{M}_{vk, v'k}|^2 = e \frac{\hbar\omega}{4q} \left(\frac{1}{\varepsilon_\infty} - \frac{1}{\varepsilon_0} \right) \left[N(\hbar\omega) + \frac{1}{2} \mp \frac{1}{2} \right] H_{vv'}(q) \quad (222)$$

where:

- $N(\hbar\omega) = 1/(e^{\hbar\omega/k_B T} - 1)$ is the phonon occupancy.
- q is the magnitude of the change in the wavevector between the initial and final states.
- ε_∞ and ε_0 are the high-frequency and static permittivities, respectively.

The upper and lower signs in [Equation 222](#) refer to absorption and emission processes, respectively. The POP form-factor, $H_{vv'}$, is given by:

$$H_{vv'}(q) = \iint \left(\overrightarrow{\Psi}_v(z; \mathbf{k}) \cdot \overrightarrow{\Psi}_{v'}^*(z; \mathbf{k}) \right) e^{-q|z-z'|} \left(\overrightarrow{\Psi}_v(z'; \mathbf{k}) \cdot \overrightarrow{\Psi}_{v'}^*(z'; \mathbf{k}) \right) dz dz' \quad (223)$$

By default, the wavefunctions are evaluated at the center of the \mathbf{k} -space grid. To take into account the initial \mathbf{k} -vector dependency of the wavefunction, use the `kdepFF` argument. The mechanism can be screened with either screening model.

Chapter 13: Subband and Mobility Calculations

Scattering Models

This model is specified with the name `POP`. [Table 46](#) lists the user-modifiable parameters for this model. Note that the scattering models for emission and absorption processes must be specified separately.

Table 46 Parameters for polar-optical phonon scattering

Equation parameter	Parameter name	Unit	Description
ϵ_∞	epsInfinity	ϵ_{vac}	High-frequency permittivity
ϵ_0	eps0	ϵ_{vac}	Static permittivity
$\hbar\omega$	hbarOmega	eV	Phonon energy

Remote Soft-Optical Phonon Scattering

Remote soft-optical (SO) phonon scattering is a mechanism that must be considered when utilizing high- κ dielectric materials in the gate stack of a transistor. It arises due to polarization induced in the high- κ layer, which leads to the creation of the SO phonon mode. This is referred to as a *remote mechanism* because there is an explicit dependence in the scattering model on the distance of the scattered carrier from the high- κ material that determines the strength of the phonon.

The unscreened matrix element is given by:

$$|\tilde{M}_{vk, v'k}|^2 = e^2 \frac{\hbar\omega_{\text{SO}}(q)}{4\pi q} \frac{1}{\hat{\epsilon}(q)} \left(N(\hbar\omega_{\text{SO}}(q)) + \frac{1}{2} \mp \frac{1}{2} \right) \times H_{vv'}(q) \quad (224)$$

where:

- $N(\hbar\omega) = 1/(e^{\hbar\omega/k_B T} - 1)$ is the phonon occupancy.
- q is the magnitude of the change in the wavevector between the initial and final states.
- $\hbar\omega_{\text{SO}}(q)$ is the q -dependent phonon energy described by the phonon dispersion.
- $1/\hat{\epsilon}(q) = (1/\epsilon_{\text{eff}}(\epsilon_{\text{HK, high}}) - 1/\epsilon_{\text{eff}}(\epsilon_{\text{HK, low}}))$, where ϵ_{eff} is the effective permittivity.

The upper and lower signs in [Equation 224](#) refer to absorption and emission processes, respectively.

The form factor, $H_{vv'}$, is given by:

$$H_{vv'}(q) = (\int \int \xi_n(z) \xi_{n'}^\dagger(z) \exp(-q|z + z'|) \xi_n^\dagger(z') \xi_{n'}(z') dz dz' \\ + \int \int \xi_n(z) \xi_{n'}^\dagger(z) \exp(-q(T_{\text{Semi}} - |z + z'|)) \xi_n^\dagger(z') \xi_{n'}(z') dz dz') \quad (225)$$

Chapter 13: Subband and Mobility Calculations

Scattering Models

The phonon dispersion is given by:

$$\omega_{SO}(q) = \omega_{TO} \left| \frac{\frac{1 - e^{-2qT_{HK}}}{1 + e^{-2qT_{HK}}} \frac{1 - \frac{\epsilon_{Ox} + \epsilon_{Semi}}{\epsilon_{Ox} - \epsilon_{Semi}} e^{2qT_{itl}}}{\frac{1 + \frac{\epsilon_{Ox} + \epsilon_{Semi}}{\epsilon_{Ox} - \epsilon_{Semi}} e^{2qT_{itl}}}{\frac{1 - e^{-2qT_{HK}}}{1 + e^{-2qT_{HK}}} \frac{1 - \frac{\epsilon_{Ox} + \epsilon_{Semi}}{\epsilon_{Ox} - \epsilon_{Semi}} e^{2qT_{itl}}}{\frac{1 + \frac{\epsilon_{Ox} + \epsilon_{Semi}}{\epsilon_{Ox} - \epsilon_{Semi}} e^{2qT_{itl}}}}}}{\epsilon_{Ox}} - \frac{\epsilon_{HK, low}}{\epsilon_{Ox}} \right. \\ \left. - \frac{\frac{1 - e^{-2qT_{HK}}}{1 + e^{-2qT_{HK}}} \frac{1 - \frac{\epsilon_{Ox} + \epsilon_{Semi}}{\epsilon_{Ox} - \epsilon_{Semi}} e^{2qT_{itl}}}{\frac{1 + \frac{\epsilon_{Ox} + \epsilon_{Semi}}{\epsilon_{Ox} - \epsilon_{Semi}} e^{2qT_{itl}}}{\frac{1 - e^{-2qT_{HK}}}{1 + e^{-2qT_{HK}}} \frac{1 - \frac{\epsilon_{Ox} + \epsilon_{Semi}}{\epsilon_{Ox} - \epsilon_{Semi}} e^{2qT_{itl}}}{\frac{1 + \frac{\epsilon_{Ox} + \epsilon_{Semi}}{\epsilon_{Ox} - \epsilon_{Semi}} e^{2qT_{itl}}}}}}{\epsilon_{Ox}} - \frac{\epsilon_{HK, high}}{\epsilon_{Ox}} \right| \quad (226)$$

The effective permittivity is given by:

$$\epsilon_{eff}(\epsilon_{HK}) = \epsilon_{HK} \left(\left(\frac{\epsilon_{Ox} - \epsilon_{Semi}}{2\epsilon_{Ox}} \right)^2 e^{-2qT_{itl}} + \left(\frac{\epsilon_{Ox} + \epsilon_{Semi}}{2\epsilon_{Ox}} \right)^2 e^{2qT_{itl}} + 2 \frac{\epsilon_{Ox}^2 - \epsilon_{Semi}^2}{(2\epsilon_{Ox})^2} \right) \frac{1 + e^{-2qT_{HK}}}{1 - e^{-2qT_{HK}}} \\ + \frac{(\epsilon_{Ox} + \epsilon_{Semi})^2}{8\epsilon_{Ox}} e^{2qT_{itl}} - \frac{(\epsilon_{Ox} - \epsilon_{Semi})^2}{8\epsilon_{Ox}} e^{-2qT_{itl}} + \frac{\epsilon_{Semi}}{2} \quad (227)$$

In Equation 224 to Equation 227, the thickness of the high- κ oxide (T_{HK}) and the thickness of the interfacial oxide (T_{itl}) are extracted automatically from the geometry of the device. If two interfaces are detected, then the semiconductor thickness (T_{Semi}) is also extracted automatically. If only one interface is detected, then the semiconductor thickness is assumed to be infinite, and the second integral in $H_{vv'}$ becomes zero.

The model is specified with the name `RemoteSOPhonon`. Table 47 lists the user-modifiable parameters for this model. Note that the scattering models for emission and absorption processes must be specified separately.

Note:

If you specify only an `hkMaterial` name, then only interfaces between the semiconductor and the `hkMaterial` are identified and applied to the scattering model. In this case, the interfacial oxide is neglected in the model by assuming that $T_{itl} \rightarrow 0$.

Be careful when specifying oxide material names for layers other than the gate stack. The names given to materials in the high- κ gate stack must be unique and must not be used in other layers, such as spacers, to avoid incorrect applications of the model.

Table 47 Parameters for remote soft-optical phonon scattering

Equation parameter	Parameter name	Unit	Description
–	interfacialMaterial	–	Name of the interfacial layer material.
–	hkMaterial	–	Name of the high- κ layer material.
ϵ_{Semi}	epsSemi	–	Static dielectric constant for the semiconductor.
ϵ_{Ox}	epsInterfacial	–	Static dielectric constant for the interfacial oxide layer.
$\epsilon_{\text{HK, low}}$	epsHkLow	–	Selected low-frequency dielectric constant for the high- κ oxide layer. It can be either static or intermediate, based on the chosen TO1 or TO2 phonon mode.
$\epsilon_{\text{HK, high}}$	epsHkHigh	–	Selected high-frequency dielectric constant for the high- κ oxide layer. It can be either intermediate or high-frequency, based on the chosen TO1 or TO2 phonon mode.
$\hbar\omega_{\text{TO}}$	hbarOmega	eV	Transverse optical phonon energy for the high- κ layer.

Alloy Disorder Scattering

A model for alloy disorder scattering for a heterostructure channel containing multiple semiconductor layers is available.

Alloy disorder scattering is an elastic isotropic process. For both electrons and holes, the matrix element for isotropic scattering is:

$$|M_{vk, v'k'}|^2 = \sum_i (1 - x_i) x_i \frac{a_i^3}{8} U^2 F_{i, v, v'} \quad (228)$$

where:

- a_i is the lattice constant of the SiGe alloy in layer i . The lattice constant is a model parameter that is typically interpolated as a function of mole fraction [20].
- U is the alloy scattering potential, which also is provided by users.

Because this mechanism is elastic and the matrix element is isotropic, the momentum relaxation factor is 1. Transitions due to alloy disorder are typically limited to either intra-subband or intravalley transitions.

This model is specified with the name `ElasticAlloy`.

Table 48 Parameters for alloy disorder scattering

Equation parameter	Parameter name	Unit	Description
U	<code>U</code>	eV	Alloy potential
a_l	<code>a</code>	cm	Lattice constant

Surface Roughness Scattering

Surface roughness scattering due to semiconductor–insulator interfaces is treated as an elastic but anisotropic process. All models for surface roughness scattering share some common modeling options and user-modifiable parameters, including which power-spectral density function to use, which momentum relaxation factor to use, which transition type to use, and whether to use carrier screening.

Power-Spectral Density Function

The surface roughness of a semiconductor–insulator interface can be characterized by a power-spectral density function, $S(q)$, which is a function of the magnitude of the difference in wavevectors between initial and final states: $q = |\mathbf{k}' - \mathbf{k}|$.

Different models for $S(q)$ are available, which are selected by using the `powerSpectrum` argument. The different forms of the power-spectral density function are:

- For the `powerSpectrum=Exponential` model:
$$S(q) = \frac{\pi\Delta^2\Lambda^2}{[1 + q^2\Lambda^2/2]^\beta}$$

This model provides the adjustable parameter `beta` for the exponent of the denominator. To obtain a true exponential spectrum, set `beta=1.5`.

- For the `powerSpectrum=Gaussian` model:
$$S(q) = \pi\Delta^2\Lambda^2 \exp(-q^2\Lambda^2/4)$$
- For the `powerSpectrum=Pirovano` model:
$$S(q) = \pi\Delta^2\Lambda^2 \exp(-q^4\Lambda^4/4)$$

All models share the common parameters listed in [Table 49](#).

Table 49 Common parameters for power-spectral density function models

Equation parameter	Parameter name	Unit	Description
Δ	delta	cm	Root mean square (RMS) amplitude
Λ	lambda	cm	Correlation length

Common Parameters

By default, `mrFactor=2` is used to compute the momentum transfer. For the transition type, transitions due to surface roughness are typically limited to either intra-subband or intravalley transitions. For screening, surface roughness scattering can remain unscreened or be treated using either of the screening models.

Surface Roughness Scattering From Parabolic Schrödinger for 1D

This formulation requires that the `Parabolic` Schrödinger solver be used. For an SOI device, the two semiconductor–insulator interfaces are assumed to be uncorrelated and, therefore, this model is applied to both interfaces using the same model parameters.

From the `Parabolic` Schrödinger solver, the unscreened matrix element for surface scattering is based on an integral formulation [21][22]. For bulk MOS capacitors, this is given by:

$$|M_{vk, v'k'}^{\text{unscr}}|^2 = \left| \int_{z_{\min}}^{z_{\max}} \Psi_v(z) \hat{\mathbf{e}}_z \cdot \mathbf{E} \Psi_{v'}(z) dz + (E_v^0 - E_{v'}^0) \int_{z_{\min}}^{z_{\max}} \left(\frac{d}{dz} \Psi_{v'}(z) \right) \Psi_v(z) dz \right|^2 \cdot S(q) \quad (229)$$

For SOI MOS capacitors, this is given by:

$$\begin{aligned} |M_{vk, v'k'}^{\text{unscr}}|^2 = & \left(-\frac{\hbar^2}{W} \int_{z_{\min}}^{z_{\max}} \Psi_v(z) \frac{d}{dz} \left(\frac{1}{m_z} \frac{d}{dz} (\Psi_{v'}(z)) \right) dz + \int_{z_{\min}}^{z_{\max}} \Psi_v \hat{\mathbf{e}}_z \cdot \mathbf{E} \left(1 - \frac{z}{W} \right) \Psi_{v'}(z) dz \right. \\ & \left. + (E_v^0 - E_{v'}^0) \int_{z_{\min}}^{z_{\max}} \left(\frac{d}{dz} \Psi_{v'}(z) \right) \left(1 - \frac{z}{W} \right) \Psi_v(z) dz \right)^2 \cdot S(q) \end{aligned} \quad (230)$$

where:

- z_{\min} and z_{\max} are the endpoints of the nonlocal line.
- $\hat{\mathbf{e}}_z \cdot \mathbf{E}$ is the vertical component of the electric field.
- E_v^0 is the minimum subband energy for subband v .
- Ψ_v is the wavefunction for subband v .

Specify `ScatteringModel=SRFromParabolic` to use this surface roughness scattering model. By default, `mrFactor=3` to use model 3.

Surface Roughness Scattering From 6kp Schrödinger for 1D

Based on the 6kp Schrödinger solver, the unscreened matrix element for surface scattering is given by [23]:

$$|M_{vk, v'k'}^{\text{unscr}}|^2 = |\Gamma_{vv'}|^2 \cdot S(q) \quad (231)$$

where:

$$|\Gamma_{vv'}|^2 = \left| \frac{\partial}{\partial z} \vec{\Psi}_{v'0}(0) \cdot H'_{\text{bulk}} \Big|_{\mathbf{k}' = \hat{\mathbf{e}}_z} \cdot \frac{\partial}{\partial z} \vec{\Psi}_{v0}(0) \right|^2 \quad (232)$$

and where:

- The wavefunction derivatives are evaluated at $\mathbf{k} = \mathbf{0}$ and at the semiconductor–insulator interface.
- H'_{bulk} is the bulk $\mathbf{k}\cdot\mathbf{p}$ Hamiltonian in device coordinates (see [Equation 150 on page 213](#)).

For an SOI device, the two semiconductor–insulator interfaces are assumed to be uncorrelated and, therefore, this model is applied to both interfaces using the same model parameters.

Specify `ScatteringModel=SRFrom6kp` or `ScatteringModel=SRFor1D` to use this surface roughness scattering model. By default, `mrFactor=2` to use model 2.

Isotropic Prange–Nee Surface Roughness Scattering From 6kp Schrödinger for 1D

Based on the 6kp Schrödinger solver and the Prange–Nee unscreened matrix element for surface scattering, the matrix element for an isotropic formulation of the surface roughness model is given by [24]:

$$|M_{vk, v'k'}^{\text{unscr}}|^2 = |\Gamma_{vv'}|^2 \cdot \frac{1}{(2\pi)^2} \int \int d\phi d\phi' S(q) [1 - \cos(\phi - \phi')] \quad (233)$$

where $|\Gamma_{vv'}|^2$ is given by [Equation 232](#).

For an SOI device, the two semiconductor–insulator interfaces are assumed to be uncorrelated and, therefore, this model is applied to both interfaces using the same model parameters.

The angular averaging in [Equation 233](#) over the initial and final states is performed on the isoenergy contour determined by the energy of the initial state.

Chapter 13: Subband and Mobility Calculations

Scattering Models

Specify `ScatteringModel=IsotropicPrangeNee` to use this surface roughness scattering model. Because this is an isotropic model, `mrFactor` is not used.

Usage for One Dimension

The names of the surface-roughness scattering models and their parameters are specified in the `Physics` command. For example, to specify an `SRFromParabolic` model, use a command of the form:

```
Physics material=Silicon ScatteringModel=SRFromParabolic \
    valleys=[list Delta1] transitionType=Intrasubband mrFactor=1 \
    screening=Lindhard powerSpectrum=Gaussian name=ElectronSR
```

In this example, an `SRFromParabolic` surface-roughness scattering model is created for intra-subband transitions within the valley named `Delta1`. The `mrFactor` argument selects model 1, and Lindhard screening is switched on. A Gaussian power spectrum is selected as well.

Surface Roughness Scattering for Two Dimensions

For 2D device structures, surface roughness scattering is implemented as a generalization of the Prange–Nee model as applied to semiconductor–insulator boundaries of arbitrary shape [25].

Specify `ScatteringModel=SRFor2D` to use this surface roughness scattering model.

Denoting the perimeter of the semiconductor–insulator boundary as C and the channel direction as z , the matrix element is decomposed using a Fourier series in the wavevector, q_s , around the perimeter:

$$|M_{vk, v'k'}^{\text{unscr}}|^2 = \sum_{\substack{q_s = -q_{\max} \\ q_s = q_{\max}}}^{q_{\max}} |\Gamma_{vv'}(q_s)|^2 \cdot S(\sqrt{q_s^2 + q_z^2}) \quad (234)$$

where $S(q)$ is the usual power-spectral density function. The Fourier decomposed matrix element at q_s is given by an integral around the perimeter of the semiconductor–insulator boundary:

$$\Gamma_{vv'}(q_s) = \oint_C ds \tilde{\Gamma}_{vv'}(s) e^{iq_s \cdot s} \quad (235)$$

where the matrix element at position s along the perimeter is given by:

$$\tilde{\Gamma}_{vv'}(s) = \Delta(s) \cdot [(\nabla_n \vec{\Psi}_{v0}(s)) \cdot H \cdot (\nabla_n \vec{\Psi}_{v'0}^*(s))] \quad (236)$$

where H' is the kinetic energy part of the bulk Hamiltonian. The gradients of the wavefunctions are taken along the local normal to the semiconductor–insulator interface at position s . The local amplitude of the surface roughness at position s is given by $\Delta(s)$.

Chapter 13: Subband and Mobility Calculations

Scattering Models

The specification of the local surface-roughness amplitude is performed using a list of pairs where each pair consists of a surface orientation and an amplitude value in cm. During the calculation of the matrix element around the perimeter, the surface normal of each local segment snaps to the closest user-specified surface orientation, and the corresponding amplitude value is used. For example, to specify different values of the surface roughness amplitude for (100) and (110) surfaces, the `delta` argument is specified as a Tcl list as follows:

```
delta=[list {100 0.2e-7} {110 0.5e-7}]
```

This specification would apply an amplitude value of 0.2 nm to surfaces that are closest to (100) and a value of 0.5 nm to surfaces closest to (110).

Conversely, you can specify a single value for the `delta` argument that will apply to all surfaces. For example, to apply a global amplitude value of 0.4 nm to all surfaces, specify the `delta` argument in Tcl as:

```
delta=0.4e-7
```

Usage for Two Dimensions

As an example, to specify an `SRFor2D` model for a silicon device, use a command of the form:

```
Physics material=Silicon ScatteringModel=SRFor2D name=ElectronSR \
    valleys=[list Delta1 Delta2 Delta3] transitionType=Intravalley \
    mrFactor=3 powerSpectrum=Exponential beta=1.5 lambda=1.5e-7 \
    delta=[list {100 0.2e-7} {110 0.5e-7}]
```

In this example, an `SRFor2D` surface roughness scattering model is created for intravalley transitions within the three silicon Δ -valleys. The `mrFactor` is set to select the model 3 and an exponential power spectrum is selected with an exponent of 1.5. The power spectrum `lambda` argument is set to 1.5 nm, while `delta` consists of a list of two pairs specifying an amplitude of 0.2 nm and 0.5 nm for surface orientations nearest to (100) and (110), respectively.

Coulomb Scattering

Coulomb scattering due to bulk impurities and interface charge is activated using the model name `Coulomb` and is treated as an elastic, intravalley scattering mechanism. The norm of the overall screened matrix element for Coulomb scattering is given by:

$$\left| M_{\mu\mu'}^{scr}(q) \right|^2 = s \cdot \left[\int \left| M_{\mu\mu'}^{0, scr}(q, \mathbf{x}_0) \right|^2 N_{BI}(\mathbf{x}_0) d\mathbf{x}_0 + \left| M_{\mu\mu'}^{0, scr}(q, \mathbf{x}_i) \right|^2 N_{SI}(\mathbf{x}_i) \right] \quad (237)$$

where:

- $M_{\mu\mu'}^{\text{scr}}(q)$ is the overall screened matrix element, evaluated as a function of the magnitude of the wavevector difference, $q = |\mathbf{k}' - \mathbf{k}|$, between the initial and final states.
- x_0 is the location of the bulk impurity charge.
- $N_{\text{BI}}(x_0)$ is the amount of bulk impurity charge at x_0 that contributes to scattering. This is given by the `TotalConcentration`.
- $N_{\text{SI}}(x_i)$ is a representative interface charge at location x_i . This is given by the absolute value of all the interface charges specified using the `TrapModel` keyword in the `Physics` command as described in [Interface Charge on page 195](#).
- s is a unitless factor that is used to scale the overall squared matrix element. This is specified using the `scaleFactor` parameter for Coulomb scattering. It is set to 1.0 by default.
- $M_{\mu\mu'}^{0, \text{scr}}(q, x_0)$ is the screened point-charge matrix element due to a point charge at x_0 .

The screened point-charge matrix element is computed from the unscreened point-charge matrix element, which is given by:

$$M_{\mu\mu'}^0(q, x_0) = \int \overrightarrow{\Psi}_{\mu}(x) \cdot \overrightarrow{\Psi}_{\mu'}^{\dagger}(x) G(q, x, x_0) dx \quad (238)$$

where:

- $\overrightarrow{\Psi}_{\mu}(x)$ is the wavefunction for subband μ .
- $G(q, x, x_0)$ is the Coulomb Green's function, described in [Coulomb Green's Function on page 229](#).

Common Parameters

By default, `mrFactor=2` is used to compute the momentum transfer. For the transition type, Coulomb scattering must always be treated as an intravalley mechanism (`transitionType=Intravalley`). For screening, Coulomb scattering must not remain unscreened. Either the scalar or tensor screening model can be used as described in [Screening on page 233](#).

Speeding Up the Matrix Element Calculation

The calculation of the unscreened matrix element, [Equation 238](#), can be time-consuming when bulk doping is treated in a large 2D cross section. An algorithm to speed up this calculation is activated by default. This algorithm uses additional memory to precompute some of the needed quantities. For typical applications, this additional memory consumption should not be large. However, if this additional memory consumption becomes an issue for

Chapter 13: Subband and Mobility Calculations

Scattering Models

large applications, you can deactivate this optimization by specifying `useCache=0` when activating Coulomb scattering.

Usage

Besides setting the required screening approach, the only adjustable argument for Coulomb scattering is `scaleFactor`. For each region in which Coulomb scattering is activated, the bulk `TotalConcentration` at all region nodes and all of the interface charges at the region interfaces are considered automatically in the Coulomb scattering calculation.

To ensure that all of the doping and interface charges in the device are considered in the Coulomb scattering calculation, it is recommended to always activate Coulomb scattering for all regions. For example, to activate Coulomb scattering for electrons in a silicon device with three Δ -valleys, it is recommended to use the following:

```
Physics material=all ScatteringModel=Coulomb name="eCoulomb"
    scaleFactor=1.0
    transitionType=Intravalley valleys=[list Delta1 Delta2 Delta3]
    screening=LindhardTDF
```

Default Scattering Models

Based on the default valley models created for silicon listed in [Table 31](#), a set of scattering models for silicon is specified by default. Parameters common to several of the scattering models are listed in [Table 50](#), the acoustic phonon scattering models are listed in [Table 51](#), the inelastic phonon scattering models are listed in [Table 52](#). For 1D devices, surface roughness scattering is also activated by default, and the default parameters are listed in [Table 53](#). The electron parameters here are suitable for (100) surfaces; while the hole parameters are suitable for all surface orientations.

Table 50 Default common scattering parameters for silicon

Parameter name	Default value
density	2.33 g/cm ³
ul	9.05e5 cm/s

Chapter 13: Subband and Mobility Calculations

Scattering Models

Table 51 Default acoustic phonon models and scattering parameters for silicon

Parameter name	ScatteringModel	Transition type	Valleys	Dac [eV]
ElectronAC	ElasticAcousticPhonon	Intravalley	Delta1, Delta2, Delta3	14.0
HoleAC	ElasticAcousticPhonon	Intravalley	Gamma	10.2

Table 52 Default inelastic phonon scattering models and parameters for Si

Parameter name	ScatteringModel	Transition type	Valleys	DtK [eV/cm]	$\hbar\omega$ [meV]
ElectronIVf1	InelasticPhonon	Intervalley	fValleys	0.3e8	18.96
ElectronIVf2	InelasticPhonon	Intervalley	fValleys	2.0e8	47.4
ElectronIVf3	InelasticPhonon	Intervalley	fValleys	2.0e8	59.03
ElectronIVg1	InelasticPhonon	gIntervalley	gValleys	0.5e8	12.06
ElectronIVg2	InelasticPhonon	gIntervalley	gValleys	0.8e8	18.53
ElectronIVg3	InelasticPhonon	gIntervalley	gValleys	11.0e8	62.04
HoleOP	InelasticPhonon	Intravalley	Gamma	15.0e8	62.0

Notes:

- Both emission and absorption versions of the models in [Table 52](#) are specified.
- fValleys refers to the valley pairs {Delta1 Delta2}, {Delta1 Delta3}, {Delta2 Delta1}, {Delta2 Delta3}, {Delta3 Delta1}, and {Delta3 Delta2}.
- gValleys refers to the valleys Delta1, Delta2, Delta3.

Chapter 13: Subband and Mobility Calculations

Material Database

Table 53 Default surface scattering models and parameters for silicon for 1D devices

Parameter name	ScatteringModel	Valleys	delta [cm]	lambda [cm]
ElectronSR	SRFromParabolic	Delta1, Delta2, Delta3	0.40e-7	1.2e-7
HoleSR	SRFrom6kp	Gamma	0.25e-7	4.0e-7

Note that both ElectronSR and HoleSR use:

- Intravalley transition type
- Exponential power spectrum with $\beta = 1.5$
- No dielectric screening

Material Database

Material properties and models are defined and stored within the material database (DB). The material DB is available to access, modify, and view material properties at any point within the Tcl interpreter of Sentaurus Band Structure.

You can utilize the material DB to define and store material definitions independently from the Sentaurus Band Structure command file.

The material definitions applied to device materials, or regions, or both are synchronized with the database and allow the material properties in use to be analyzed throughout the simulation.

Support for binary and ternary alloy material definitions is included, allowing for the application of arbitrary mole-fraction material instances.

[Table 54](#) lists the set of materials defined by default and available for use. If transport is available for a given carrier, then by default the phonon models are specified. For binary and ternary materials, the alloy scattering model is also defined.

These default materials are available to apply to any device structures loaded from TDR files.

Chapter 13: Subband and Mobility Calculations

Material Database

Table 54 Available materials in the material database

Material name	Default applied	Material type	Crystal form	Mole-fraction dependent	Electron transport available	Hole transport available
Silicon	Yes	Semiconductor	Cubic	No	Yes	Yes
PolySilicon	Yes	Semiconductor	Cubic	No	Yes	Yes
Germanium	No	Semiconductor	Cubic	No	Yes	Yes
Silicon _{1-x} Germanium _x	No	Semiconductor	Cubic	Yes	Yes	Yes
AlAs	No	Semiconductor	Cubic	No	Yes	No
GaAs	No	Semiconductor	Cubic	No	Yes	No
InAs	No	Semiconductor	Cubic	No	Yes	No
Al _x Ga _{1-x} As	No	Semiconductor	Cubic	Yes	Yes	No
In _x Ga _{1-x} As	No	Semiconductor	Cubic	Yes	Yes	No
In _{1-x} Al _x As	No	Semiconductor	Cubic	Yes	Yes	No
AlN	No	Semiconductor	Wurtzite	No	Yes	No
GaN	No	Semiconductor	Wurtzite	No	Yes	No
InN	No	Semiconductor	Wurtzite	No	Yes	No
Al _x In _{1-x} N	No	Semiconductor	Wurtzite	Yes	Yes	No
In _x Ga _{1-x} N	No	Semiconductor	Wurtzite	Yes	Yes	No
Al _x Ga _{1-x} N	No	Semiconductor	Wurtzite	Yes	Yes	No
4H-SiC	No	Semiconductor	Wurtzite	No	No	No
6H-SiC	No	Semiconductor	Wurtzite	No	No	No
Oxide	Yes	Insulator	—	No	No	No
LowK	No	Insulator	—	No	No	No

Chapter 13: Subband and Mobility Calculations

Material Database

Table 54 Available materials in the material database (Continued)

Material name	Default applied	Material type	Crystal form	Mole-fraction dependent	Electron transport available	Hole transport available
Ga ₂ O ₃	No	Insulator	—	No	No	No
Al ₂ O ₃	No	Insulator	—	No	No	No
Metal	No	Metal	—	No	No	No

Applying Materials to a Loaded Device

Only the Silicon, PolySilicon, and Oxide materials are applied by default to a LoadDevice command. These materials are applied to the loaded TDR device only when material name definitions match. All other material DB definitions must be manually applied to a loaded TDR device.

You can use the ApplyMaterial command to apply a material DB definition to a loaded TDR device (see [ApplyMaterial on page 351](#)). For example, to apply the DB Germanium material to the TDR device material Germanium, you would specify:

```
ApplyMaterial dbMaterial=Germanium deviceMaterial=Germanium
```

The ApplyMaterial command applies the material properties, as defined in the material DB, at the time of the command execution. Any subsequent modifications to the DB material, or the device material or region directly, are synchronized automatically between the definition in the material DB and that in the loaded device.

Note:

Using the ApplyMaterial command overwrites the existing material definition in the loaded device with the definition in the material DB.

In the previous example, there is an abstraction between the material name used in the material DB and the material name in the loaded TDR device. This abstraction allows you to apply any material DB definition to any material name tag in a loaded TDR device. An example command of applying the DB Germanium material to the TDR material tag Silicon is:

```
ApplyMaterial dbMaterial=Germanium deviceMaterial=Silicon
```

You can also apply a material definition to regions. For example, to apply the DB Germanium material to the region channel1, you would use the following command:

```
ApplyMaterial dbMaterial=Germanium deviceRegion=channel1
```

Chapter 13: Subband and Mobility Calculations

Material Database

For binary alloy material definitions, the mole fraction must be specified using the `fraction` argument. The following example applies the `SiliconGermanium` material with a mole fraction of $x=0.5$ to the region `channel1`:

```
ApplyMaterial dbMaterial=SiliconGermanium deviceRegion=channel1 \
    fraction=0.5
```

You can also specify stress or strain values by using the `ApplyMaterial` command.

Note:

When applying materials to loaded TDR devices, as a minimum, you should check that the material types match, for example, applying only insulator DB types to an insulator material tag in a TDR device.

Viewing the Mapping of Materials to a Loaded Device

The mapping of materials between the database and a device is recorded to allow synchronization and can be viewed at any time from within the Tcl interpreter.

To view the mapping of the currently loaded device, use the following command (see [ListMaterialMap on page 354](#)):

```
ListMaterialMap
```

The following example shows the output for a simple device:

```
List of mappings between material database to device:
- Silicon -> Silicon [regions: sil]
- Oxide -> Oxide [regions: ox1]
```

Database materials are always listed on the left-hand side of the assignment operator (`->`), with the device target listed on the right-hand side. The device target can be either a material or a region (see [Applying Materials to a Loaded Device on page 258](#)). Brackets on the right-hand side denote either the regions of a given mapped material, or the material of a given mapped region based on the specific device mapping target.

In this device example, there are two device materials, `Silicon` and `Oxide`, corresponding to two regions, `sil` and `ox1`. In the output, you can see that two mappings are stored:

- The first maps the database material `Silicon` to the device material `Silicon` that is associated with the device region `sil`.
- The second maps the database material `Oxide` to the device material `Oxide` that is associated with the device region `ox1`.

Chapter 13: Subband and Mobility Calculations

Material Database

If within this example the mapping was modified such that a specific material, `Silicon_sil` in this example, was assigned to the region `sil`, then the output would be the following:

```
List of mappings between material database to device:  
- Oxide -> Oxide [regions: ox1]  
- Silicon_sil -> sil [material: Silicon]
```

Viewing Material Parameters

The parameter values and models defined for a material in the material DB can be viewed at any time by using the Tcl interpreter.

For example, to view the parameters and models for the binary alloy `SiliconGermanium`, use the following command (see [PrintParameters on page 352](#)):

```
PrintParameters dbMaterial=SiliconGermanium
```

If you want to view only the parameters for a specific model, then you can specify a particular model type and name. The following command demonstrates how to view the `Delta1` valley of `SiliconGermanium`:

```
PrintParameters dbMaterial=SiliconGermanium valley=Delta1
```

Another example to view the parameter values for the `ElectronAC` scattering model of the material `SiliconGermanium` is:

```
PrintParameters dbMaterial=SiliconGermanium scatteringModel=ElectronAC
```

To view specific material parameters as mapped to a device material, use the following command:

```
PrintParameters deviceMaterial=Silicon
```

Alternatively, you can view the specific material parameters as mapped to a device region by using the following command:

```
PrintParameters deviceRegion=sil
```

These commands can also be used to output specific valley parameters, scattering model parameters, or both as previously described.

The output from any combination of the `PrintParameters` command can be directed to a file in the current directory. An example of this for the material associated with device region `sil` is:

```
PrintParameters deviceRegion=sil filename=region_sil_material.txt
```

Defining Binary Alloy Materials

The material DB supports binary alloy material definitions and provides a framework to define and modify them. This framework allows the material to be defined capturing the mole fraction dependence in a dynamic manner. That is, you can apply arbitrary and multiple fixed mole fractions to a loaded TDR device from a single material DB definition.

The SiGe binary alloy, $\text{Si}_{1-x}\text{Ge}_x$, is provided by default and the following sections describe the various functionality components used to construct the definition.

Defining Interpolated Parameters

Taking as an example a simplified $\text{Si}_{1-x}\text{Ge}_x$ binary alloy semiconductor, you can define a new semiconductor of name `SiGeTest` in the material DB, with linear interpolation, by using the following command:

```
CreateMaterial dbMaterial=SiGeTest \
    type=semiconductor \
    Permittivity=[list 11.7 16.0] \
    Affinity=[list 4.0727 3.92] \
    Bandgap=[list 1.1242 0.67] \
    Nc=[list 2.8213257e19 1.2571E+19] \
    Nv=[list 3.1046e19 6.2503E+18] \
    a0=[list 5.43 5.64] \
    s11=[list 0.762451 0.96] \
    s12=[list -0.213158 -0.26] \
    s44=[list 1.25 1.5]
```

With this command, each of the material parameters is defined as a simple linear interpolation from mole fraction $x=0$ to $x=1$.

You can use the `PrintParameters dbMaterial=SiGeTest` command to verify interpolation properties:

```
SiGeTest - Semiconductor Properties
    Permittivity= ((0.0...1.0), (11.7...16.0))
    Affinity= ((0.0...1.0), (4.0727...3.92)) [eV]
    Bandgap= ((0.0...1.0), (1.1242...0.67)) [eV]
    eDensity= eHybridDensity
    hDensity= hHybridDensity
    eBulkDensity= eMultiValleyDensity
    hBulkDensity= hMultiValleyDensity
    Nc= ((0.0...1.0), (2.8213257e19...1.2571E+19)) [cm^-3]
    Nv= ((0.0...1.0), (3.1046e19...6.2503E+18)) [cm^-3]

Crystal Properties:
    type= Cubic
    s11= ((0.0...1.0), (0.762451...0.96)) [%/GPa]
    s12= ((0.0...1.0), (-0.213158...-0.26)) [%/GPa]
    s44= ((0.0...1.0), (1.25...1.5)) [%/GPa]
```

Chapter 13: Subband and Mobility Calculations

Material Database

```
stressTensor= {{0 0 0} {0 0 0} {0 0 0}} [Pa]
a0= ((0.0...1.0), (5.43...5.64)) [Angstrom]
```

Taking the lattice constant a_0 as an example, you see the output shows that the value will vary from $a_0(x=0)=5.43$ to $a_0(x=1)=5.64$.

Not all material parameters can be interpolated, such as the carrier density model definition, the crystal orientations, and the stress tensor. If a parameter is of type `Integer`, `Boolean`, or `String`, then it cannot be interpolated. For details about parameter types, see [Material Database on page 342](#).

The interpolation of parameters is extended to the valley and scattering model definitions. Again, if a parameter has a type that is not of `Double` or describes an orientation, then it cannot be interpolated.

Parameter Interpolation Table

Certain parameters in binary alloy materials might require more accurate interpolation. This section describes interpolation using a piecewise linear model.

The piecewise linear interpolation model allows parameters to be defined at various fixed values between the mole fraction $x=0..1$. These values must be provided in an ASCII table format as shown here:

xFrac	Bandgap	Affinity	Nc	Nv	Permittivity	s11	s12	s44
0	1.12	4.05	2.9112E+19	2.0640E+19	11.7	0.77	-0.21	1.25
0.1	1.079	4.029	2.9293E+19	1.7144E+19	12.13	0.789	-0.215	1.275
0.2	1.042	4.021	2.9473E+19	1.4623E+19	12.56	0.808	-0.22	1.3
0.3	1.01	3.998	2.9649E+19	1.2745E+19	12.99	0.827	-0.225	1.325
0.4	0.981	3.989	2.9825E+19	1.1260E+19	13.42	0.846	-0.23	1.35
0.5	0.957	3.984	2.9999E+19	1.0270E+19	13.85	0.865	-0.235	1.375
0.6	0.915	3.938	3.0175E+19	9.3817E+18	14.28	0.884	-0.24	1.4
0.7	0.9	3.887	3.0352E+19	8.6116E+18	14.71	0.903	-0.245	1.425
0.8	0.89	3.832	3.0529E+19	7.8455E+18	15.14	0.922	-0.25	1.45
0.9	0.82	3.836	1.4504E+19	6.9983E+18	15.57	0.941	-0.255	1.475
1	0.67	3.92	1.2571E+19	6.2503E+18	16.0	0.96	-0.26	1.5

This table provides interpolation values over 11 points between $x=0..1$ for the material parameters of a $\text{Si}_{1-x}\text{Ge}_x$ binary alloy semiconductor. The mole fraction field must be named `xFrac` to ensure it is parsed properly. All other parameter fields must match their corresponding parameter name.

Using the $\text{Si}_{1-x}\text{Ge}_x$ material again as an example, you can define a new semiconductor with the name `SiGeTest` in the material DB with piecewise linear interpolation of the parameters by using the following command:

```
CreateMaterial dbMaterial=SiGeTest \
    type=semiconductor \
    table=SiGe_ElectrostaticParams.txt;
```

Chapter 13: Subband and Mobility Calculations

Material Database

Here, the file `SiGe_ElectrostaticParams.txt` contains the ASCII table previously described.

In certain circumstances, you might need to specify multiple sets of parameters within one ASCII table. To allow the parser to assign the parameters correctly, it is possible to prepend a name to the parameter field in the ASCII table. This prepended name can be either the name of the model to which it should be associated or an arbitrary name.

The following simplified table demonstrates two examples of prepended names for the `Permittivity` parameter:

xFrac	SiGeTest_Permittivity	MyInGaAs_Permittivity
0	11.7	12.9
0.1	12.13	13.04
0.2	12.56	13.18
0.3	12.99	13.32
0.4	13.42	13.46
0.5	13.85	13.6
0.6	14.28	13.74
0.7	14.71	13.88
0.8	15.14	14.02
0.9	15.57	14.16
1	16.0	14.3

This table is saved to an example file named `Permittivity_Test.txt`.

You could use this table to modify the permittivity of the present example material `SiGeTest`. The permittivity interpolation field `SiGeTest_Permittivity` can be assigned to `SiGeTest` using the following command (see [CreateMaterial and ModifyMaterial on page 342](#)):

```
ModifyMaterial dbMaterial=SiGeTest \
    table=Permittivity_Test.txt \
    useNamePrefix=1;
```

In addition, the permittivity interpolation field `MyInGaAs_Permittivity` can be assigned to the material definition `InGaAs` using the following command:

```
ModifyMaterial dbMaterial=InGaAs \
    table=Permittivity_Test.txt \
    useNamePrefix=MyInGaAs;
```

Interpolation of Material-Specific Scattering Models

There is a specific subset of scattering models that, when applied to a binary alloy material, are applied to each component of the binary alloy.

Using the $\text{Si}_{1-x}\text{Ge}_x$ semiconductor with acoustic phonon scattering as an example, this is typically applied using the component material parameters and using a linear interpolation of Dac^2 . This results in two acoustic phonon scattering models, each with the component material-specific density and sound velocity, but with an interpolated deformation potential.

Chapter 13: Subband and Mobility Calculations

Material Database

The scattering models that are applied in this way include:

- Acoustic elastic phonon
- Optical inelastic phonon
- Ohashi acoustic elastic phonon
- Tensor optical inelastic phonon
- Surface roughness

When specifying these scattering models for application in a binary alloy, an additional parameter is required to associate the definition with the binary alloy component. For the binary alloy material $A_{1-x}B_x$, consider that material *A* is the *major* component and material *B* is the *minor* component.

For example, to apply acoustic elastic phonon scattering to the SiGeTest material for the X-valleys, you define an acoustic phonon for each component using the following commands:

```
set massDensity_Si 2.33;          # [g/cm3] Mass density
set ul_Si 9.05e5;                # [cm/s] Speed of sound
set massDensity_Ge 5.323;         # [g/cm3] Mass density
set ul_Ge 5.41e5;                # [cm/s] Speed of sound

AddScatteringModel dbMaterial=SiGeTest \
                   ScatteringModel=ElasticAcousticPhonon \
                   name=ElectronAC_Si_X \
                   transitionType=Intravalley \
                   valleys=[list X1 X2 X3] \
                   Dac=14.0 \
                   density=$massDensity_Si \
                   ul=$ul_Si \
                   alloyComponent=major;

AddScatteringModel dbMaterial=SiGeTest \
                   ScatteringModel=ElasticAcousticPhonon \
                   name=ElectronAC_Ge_X \
                   transitionType=Intravalley \
                   valleys=[list X1 X2 X3] \
                   Dac=9.0 \
                   density=$massDensity_Ge \
                   ul=$ul_Ge \
                   alloyComponent=minor;
```

It is also possible to define both material components of a scattering model with one command. Using the acoustic elastic phonon example again, the command can specify both alloy components as follows:

```
AddScatteringModel dbMaterial=SiGeTest \
                   ScatteringModel=ElasticAcousticPhonon \
                   name=ElectronAC_X \
```

Chapter 13: Subband and Mobility Calculations

References

```
transitionType=Intravalley \
valleys=[list X1 X2 X3] \
Dac=[list 14.0 9.0] \
density=[list $massDensity_Si $massDensity_Ge] \
ul=[list $ul_Si $ul_Ge] \
alloyComponent=both;
```

References

- [1] B. A. Foreman, “Elimination of spurious solutions from eight-band $k \cdot p$ theory,” *Physical Review B*, vol. 56, no. 20, pp. R12748–R12751, 1997.
- [2] T. B. Bahder, “Eight-band $k \cdot p$ model of strained zinc-blende crystals,” *Physical Review B*, vol. 41, no. 17, pp. 11992–12001, 1990.
- [3] R. G. Veprek, S. Steiger, and B. Witzigmann, “Ellipticity and the spurious solution problem of $k \cdot p$ envelope equations,” *Physical Review B*, vol. 76, p. 165320, October 2007.
- [4] C. Y.-P. Chao and S. L. Chuang, “Spin-orbit-coupling effects on the valence-band structure of strained semiconductor quantum wells,” *Physical Review B*, vol. 46, no. 7, pp. 4110–4122, 1992.
- [5] S. Takagi *et al.*, “On the Universality of Inversion Layer Mobility in Si MOSFET’s: Part I—Effects of Substrate Impurity Concentration,” *IEEE Transactions on Electron Devices*, vol. 41, no. 12, pp. 2357–2362, 1994.
- [6] A.-T. Pham, B. Meinerzhagen, and C. Jungemann, “A fast $\vec{k} \cdot \vec{p}$ solver for hole inversion layers with an efficient 2D \vec{k} -space discretization,” *Journal of Computational Electronics*, vol. 7, no. 3, pp. 99–102, 2008.
- [7] C. Jungemann, A. Emunds, and W. L. Engl, “Simulation of Linear and Nonlinear Electron Transport in Homogeneous Silicon Inversion Layers,” *Solid-State Electronics*, vol. 36, no. 11, pp. 1529–1540, 1993.
- [8] M. V. Fischetti and S. E. Laux, “Monte Carlo study of electron transport in silicon inversion layers,” *Physical Review B*, vol. 48, no. 4, pp. 2244–2274, 1993.
- [9] S. Jin, M. V. Fischetti, and T.-W. Tang, “Modeling of electron mobility in gated silicon nanowires at room temperature: Surface roughness scattering, dielectric screening, and band nonparabolicity,” *Journal of Applied Physics*, vol. 102, p. 083715, October 2007.
- [10] F. Stern and W. E. Howard, “Properties of Semiconductor Surface Inversion Layers in the Electric Quantum Limit,” *Physical Review*, vol. 163, no. 3, pp. 816–835, 1967.
- [11] I. Souza, N. Marzari, and D. Vanderbilt, “Maximally localized Wannier functions for entangled energy bands,” *Physical Review B*, vol. 65, no. 3, p. 035109, 2001.

Chapter 13: Subband and Mobility Calculations

References

- [12] A. T. Pham, C. Jungemann, and B. Meinerzhagen, "Microscopic modeling of hole inversion layer mobility in unstrained and uniaxially stressed Si on arbitrarily oriented substrates," *Solid-State Electronics*, vol. 52, no. 9, pp. 1437–1442, 2008.
- [13] S. Rodríguez *et al.*, "Hole confinement and energy subbands in a silicon inversion layer using the effective mass theory," *Journal of Applied Physics*, vol. 86, no. 1, pp. 438–444, 1999.
- [14] S.-M. Hong, A.-T. Pham, and C. Jungemann, *Deterministic Solvers for the Boltzmann Transport Equation*, Springer: Vienna, 2011.
- [15] M. V. Fischetti *et al.*, "Six-band $k \cdot p$ calculation of the hole mobility in silicon inversion layers: Dependence on surface orientation, strain, and silicon thickness," *Journal of Applied Physics*, vol. 94, no. 2, pp. 1079–1095, 2003.
- [16] Y. Lee, *Modeling and Simulation of Nanowire MOSFETs*, PhD thesis, Tokyo Institute of Technology, Japan, March 2012.
- [17] D. Esseni and A. Abramo, "Modeling of Electron Mobility Degradation by Remote Coulomb Scattering in Ultrathin Oxide MOSFETs," *IEEE Transactions on Electron Devices*, vol. 50, no. 7, pp. 1665–1674, 2003.
- [18] T. Ohashi *et al.*, "Experimental Evidence of Increased Deformation Potential at MOS Interface and Its Impact on Characteristics of ETSOI FETs," in *IEDM Technical Meeting*, Washington, DC, USA, pp. 390–393, December 2011.
- [19] S. Koba *et al.*, "The Impact of Increased Deformation Potential at MOS Interface on Quasi-Ballistic Transport in Ultrathin Channel MOSFETs Scaled down to Sub-10 nm Channel Length," in *IEDM Technical Meeting*, Washington, DC, USA, pp. 312–315, December 2013.
- [20] M. M. Rieger and P. Vogl, "Electronic-band parameters in strained $\text{Si}_{1-x}\text{Ge}_x$ alloys on $\text{Si}_{1-y}\text{Ge}_y$ substrates," *Physical Review B*, vol. 48, no. 19, pp. 14276–14287, 1993.
- [21] S. Jin, M. V. Fischetti, and T.-W. Tang, "Modeling of Surface-Roughness Scattering in Ultrathin-Body SOI MOSFETs," *IEEE Transactions on Electron Devices*, vol. 54, no. 9, pp. 2191–2203, 2007.
- [22] D. Esseni, "On the Modeling of Surface Roughness Limited Mobility in SOI MOSFETs and Its Correlation to the Transistor Effective Field," *IEEE Transactions on Electron Devices*, vol. 51, no. 3, pp. 394–401, 2004.
- [23] A. T. Pham, C. Jungemann, and B. Meinerzhagen, "Modeling of hole inversion layer mobility in unstrained and uniaxially strained Si on arbitrarily oriented substrates," in *Proceedings of the 37th European Solid-State Device Research Conference (ESSDERC)*, Munich, Germany, pp. 390–393, September 2007.
- [24] A.-T. Pham, C. Jungemann, and B. Meinerzhagen, "Physics-Based Modeling of Hole Inversion-Layer Mobility in Strained-SiGe-on-Insulator," *IEEE Transactions on Electron Devices*, vol. 54, no. 9, pp. 2174–2182, 2007.

Chapter 13: Subband and Mobility Calculations

References

- [25] S. Jin *et al.*, “Coupled Drift-Diffusion (DD) and Multi-Subband Boltzmann Transport Equation (MSBTE) Solver for 3D Multi-Gate Transistors,” in *Proceedings of the International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, Glasgow, Scotland, pp. 348–351, September 2013.

14

Sentaurus Band Structure/Tcl Commands

This chapter presents information about the commands used by Sentaurus Band Structure.

Overview of Commands and Syntax

Sentaurus Band Structure extends the Tcl syntax by adding named argument parsing, type checking, support for real and complex 3D vector or matrix arithmetic, and application-specific commands for band-structure calculation. The type-naming scheme for arguments to Tcl commands is described in [get_type on page 339](#).

Table 55 summarizes the available types. Except where noted, commands use named arguments.

Table 55 Types recognized by Sentaurus Band Structure

Type	Description
Numeric types (implicit type propagation allowed from top to bottom)	
Integer	For example: 100
Double	For example: 3.1415927; 1.0
Complex	For example: (0.0,1.0) for $i = \sqrt{-1}$ (no spaces allowed)
Instantiable classes	
EPM::AtomicSpecies	Atomic species for use in an EPM::Crystal:
EPM::RealAtomicSpecies	• This subclass uses cubic-spline interpolation for V_{loc} .
EPM::RealAtomicSpecies_Friedel	• This subclass uses Friedel formula for V_{loc} .
EPM::RealAtomicSpecies_VlocProc	• This subclass uses a Tcl procedure for V_{loc} .
EPM::VirtualAtomicSpecies	• This subclass describes a statistical mixture of species for virtual crystal approximation (VCA).

Chapter 14: Sentaurus Band Structure/Tcl Commands

Overview of Commands and Syntax

Table 55 Types recognized by Sentaurus Band Structure (Continued)

Type	Description
EPM::Crystal	Empirical pseudopotential method (EPM) band-structure solver with crystal description
AnalyticBandSolver	Band-structure solver for analytic band models
Elasticity	Class for storing elasticity information:
CubicElasticity	<ul style="list-style-type: none">Subclass for crystals of cubic symmetry
bandstructure_t	Container class for band-energy results
groupVelocity_t	Container class for group-velocity results
inverseMass_t	Container class for (reciprocal) effective mass results
Tcl-related types (implicit type propagation might drop #n or /T)	
String	Any type can be propagated to a String
Array	The type for the name of a Tcl array data-structure
List	A Tcl list data-structure with an arbitrary number of elements of arbitrary types
List#n	A list with n elements of arbitrary type
List/ T	A list with an arbitrary number of elements of type T
List# n/T	A list with n elements of type T
Notation for frequently used types	
matrix3D	= List#3/vector3D (a 3×3 matrix (three row vectors))
RealMatrix3D	= List#3/RealVector3D (a real 3×3 matrix)
vector3D	= List#3/Complex (a 3D vector)
RealVector3D	= List#3/Double (a real 3D vector)

Notational Conventions of Syntax Description

Sentaurus Band Structure supports different ways of passing arguments to commands: positional arguments and named arguments. In both conventions:

- An argument enclosed by brackets is optional. For example: [optionalArg]
- An argument followed by an ellipsis (...) indicates zero or more arguments.
- The vertical bar | is an *exclusive or* operator. Of two or more arguments separated by |, exactly one must be specified.
- Braces are used solely for grouping.

Positional Arguments

This is the usual way of passing arguments in Tcl. Argument values are listed on the command line in the sequence in which they appear in the procedure definition.

Example:

```
vectorSubtract {1 1 1} {0 1 2}
```

Result:

```
1 0 -1
```

In this example, the command takes two arguments of type `vector3D` and subtracts the second from the first.

This is denoted by the following syntax description:

- This command subtracts **v** from **u** and returns the result **u – v**.

Tcl syntax (positional arguments):

- `vectorSubtract u v`
 - `vector3D u`
 - `vector3D v`

Named Arguments

In named argument parsing, the meaning of an argument is not determined by its position in the argument list. Instead, the receiving command parses its argument list for sequences of the form `name=value` or `-name value`, and sets up a map from argument names to argument values. For details about this mechanism, see [parse_args on page 337](#).

Chapter 14: Sentaurus Band Structure/Tcl Commands

Overview of Commands and Syntax

Example:

```
check_type obj=[list 1 2 2.5] type=List/Integer
```

Output:

```
"1 2 2.5" has type List#3/Double --- expected type: List/Integer.
```

This command checks whether the object passed as argument `obj` can be converted to the type specified in the `type` argument. In this example, this check fails because a list of three Doubles cannot be converted to a list of integers. The `obj` argument can accept objects of any type. Since any Tcl object can be converted to a string, the correct type for the `obj` argument is `String`.

This leads to the following syntax description for the `check_type` command:

```
check_type obj=String type=String
```

Description of arguments:

- `obj` specifies a Tcl object or the Tcl command associated with a C++ object.
- `type` specifies a type signature string (see [get_type on page 339](#)).

Object-Oriented Tcl Commands

In the object-oriented approach to Tcl commands, each object is represented by a command with the *same name* as the object. The first argument to this command specifies the method to perform on the object [1]. Many commands of Sentaurus Band Structure work this way.

For example, if there is an `EPM::Crystal` object called `bulkSi`, then there is also a `bulkSi` Tcl command. If you wanted to print the status of the `bulkSi` object, then you would use the `bulkSi status` method. In this documentation, this method is listed as `<EPM::Crystal> status` (see [<EPM::Crystal> status on page 290](#)).

Classes for the Band-Structure Calculation

Sentaurus Band Structure supports two band-structure calculation approaches: EPM and analytic band-structure formulas. For each approach, there is a band-structure calculator class named `EPM::Crystal` and `AnalyticBandSolver`.

An `AnalyticBandSolver` object stores parameters that relate directly to the shape of the band structure, for example, the energy separation `Delta` between the heavy-hole or light-hole bands and the split-off band at the Γ -point, or the electron effective mass in the conduction-band minimum.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Material Parameter Files

An `EPM::Crystal` object stores a geometric representation of the material under study. This description consists of the size and shape of the unit cell as well as the species and positions of ions inside each unit cell. These ions, in turn, are described by `EPM::AtomicSpecies` objects. You perform the band-structure calculation by calling the `computeBandstructure` method on a band-structure calculator object and passing it a list of **k**-vectors (see [<EPM::Crystal> computeBandstructure on page 285](#)).

Both `AnalyticBandSolver` and `EPM::Crystal` objects store their results in container objects as follows:

- Band energies (`bandstructure_t`)
- First derivatives with respect to **k** (`groupVelocity_t`)
- Second derivatives with respect to **k** (`inverseMass_t`)

The default names for these containers are `name.bandstructure`, `name.groupVelocity`, and `name.inverseMass` (only `EPM::Crystal` objects compute effective masses). You can overwrite these names by using the optional `bandstructure`, `groupVelocity`, and (for `EPM::Crystal`) `inverseMass` arguments to the [<EPM::Crystal> computeBandstructure](#) method.

In addition, if you set the global `automaticBandstructureFiles` argument, then the band-structure solver writes results for the highest three valence bands (four, for effective masses) and for the lowest four conduction bands to files with the default names of `energy.dat`, `velocity.dat`, `eInvMass.dat`, and `hInvMass.dat`. You can prefix or postfix these file names with a user-supplied string stored in the global `outputPrefix` and `outputPostfix` arguments.

Raw band-structure results in Sentaurus Band Structure will, in general, have nonzero values for both the conduction band minimum and the valence band maximum. Often, you want to shift either or both of these extrema to an energy of zero. This is supported through the global arguments `shiftConductionBands` and `shiftValenceBands`. Band-shifting applies to both the energies returned by the `<bandstructure_t> get` method and the values in `energy.dat`.

No shifting is applied to band energies when they are written manually into a `bandstructure_t` object using the `<bandstructure_t> set` method.

Material Parameter Files

In Sentaurus Band Structure, material parameter files are Tcl input files that can be `source'd` to make a given material available in a session. The installation directory of the default parameter files is `${STROOT_LIB} /sband`. File names have the format `name_param.tcl`, where `name` identifies the material.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Global Settings

To change the directory for autoloading of parameter files to *dir*, specify the command:

```
sBandSet dbPath=dir
```

Global Settings

These commands retrieve or set global Sentaurus Band Structure arguments.

sBandGet and sBandSet

The command `sBandGet` retrieves the global Sentaurus Band Structure arguments.

The command `sBandSet` sets the global Sentaurus Band Structure arguments. This command accepts any number of *name=value* pairs, where:

- *name* is the name of one of the arguments listed in the *Syntax* section.
- *value* is a valid value for that argument.

Syntax

```
sBandGet automaticBandstructureFiles | cutOffEnergy | dbPath |
    degeneratePerturbationTheory | deltaK | derivOrder | fileFormat |
    nThreads | outputPostfix | outputPrefix | Qcutoff | Q2cutoff |
    shearStrainInKVecTransform | shiftConductionBands |
    shiftValenceBands | showProgressBars |
    verbose | withLS | withNL
```

```
sBandSet [automaticBandstructureFiles=Boolean] |
    [cutOffEnergy=Double] [dbPath=String] |
    [degeneratePerturbationTheory=Boolean] [deltaK=Double] |
    [derivOrder=Integer] [fileFormat=String] [nThreads=Integer] |
    [outputPostfix=String] [outputPrefix=String] [Qcutoff=Double] |
    [Q2cutoff=Double] [shearStrainInKVecTransform=Boolean] |
    [shiftConductionBands=Boolean] [shiftValenceBands=Boolean] |
    [showProgressBars=Boolean] [verbose=Boolean] |
    [withLS=Boolean] [withNL=Boolean]
```

Argument	Description
automaticBandstructureFiles	Specifies how to control automatic writing of energy, group velocity, and effective mass files from the <i><EPM::Crystal></i> <code>computeBandstructure</code> method. Options are: <ul style="list-style-type: none">• 0: Do not write automatic files.• 1: Automatically generate files (default).

Chapter 14: Sentaurus Band Structure/Tcl Commands

Global Settings

Argument	Description
cutOffEnergy	Sets the maximum kinetic energy (in Rydberg) of plane waves to be included in the basis for the Bloch wavefunctions. Default: 12.0
dbPath	Sets the directory containing the default material parameter files, the <code>sband_util.tcl</code> file, and so on. Default: (installation dependent)
degeneratePerturbationTheory	Specifies how to resolve degeneracies during the calculation of group velocities or effective masses. Options are: <ul style="list-style-type: none">• 0: Apply a small strain perturbation to split the degenerate states (default).• 1: Diagonalize the perturbing Hamiltonian on the degenerate eigenspace, and apply degenerate perturbation theory.
deltaK	Sets the step size (in $1/r_{\text{Bohr}}$) used for taking numeric \mathbf{k} -derivatives of the band structure. Default: 1e-4
derivOrder	Sets the maximum order up to which \mathbf{k} -derivatives of the band energies are computed. Options are: <ul style="list-style-type: none">• 0: Compute energies only.• 1: Compute energies and group velocities (default).• 2: Compute energies, group velocities, and reciprocal effective masses.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Global Settings

Argument	Description
fileFormat	Selects one or more output formats for <code>automaticBandstructureFiles</code> . Each format is associated with a predefined constant Tcl variable (see File Formats for Storing Band Data on page 136). Supported formats are: <ul style="list-style-type: none">• <code>\$SHORT_FORMAT</code> (default): Outputs only the three highest conduction bands and the four lowest valence bands (see File Format \$SHORT_FORMAT on page 136).• <code>\$LONG_FORMAT</code>: Outputs all bands (no sign flips) (see File Format \$LONG_FORMAT on page 139).• <code>\$TDR3DTENSOR_FORMAT</code>: Stores band data as a 3D TDR tensor-grid file (requires the <code>k</code>-vectors to form a 3D tensor grid).• <code>\$MONTECARLO_FORMAT</code>: Stores data in the format for single-particle device Monte Carlo simulations (see Creating Band Data for Sentaurus Device Monte Carlo on page 139). Multiple formats can be selected by combining their values using the binary OR operation. For example, to select both <code>\$SHORT_FORMAT</code> and <code>\$LONG_FORMAT</code> , specify <code>fileFormat=[expr \$SHORT_FORMAT \$LONG_FORMAT]</code> .
nThreads	Sets the number of threads to use for parallel tasks. Default: 1
outputPostfix, outputPrefix	If <code>automaticBandstructureFiles</code> is set, then <code>outputPrefix</code> and <code>outputPostfix</code> are prefixes and postfixes, respectively, to the file names of the automatic energy, group velocity, and effective mass file names. Default: (empty string)
Qcutoff, Q2cutoff	Sets the maximum $\ q\ $ (in $2\pi/a$) or $\ q\ ^2$ (in $(2\pi/a)^2$) for evaluating the local part of the atomic form factor. In strained crystals, the norm is computed for the corresponding unstrained wavevector. Default: -1 (that is, inactive; use the per-species value; compare to AtomicSpecies on page 292) If both the per-species and the global <code>Qcutoff</code> or <code>Q2cutoff</code> arguments are inactive (that is, they have negative values), then an error condition will occur during the band-structure calculation.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Global Settings

Argument	Description
shearStrainInKVecTransform	Specifies how to control the \mathbf{k} -vector coordinate system in the presence of shear strain. Options are: <ul style="list-style-type: none">• 0: Include only diagonal strain components in the basis (default).• 1: Include shear strain components in the basis.
shiftConductionBands	Specifies how to control <i>normalization</i> of the conduction-band energies. If both <code>shiftConductionBands</code> and <code>shiftValenceBands</code> are set, then <code>shiftConductionBands</code> is applied after <code>shiftValenceBands</code> . Options are: <ul style="list-style-type: none">• 0: Leave band structure unmodified (default).• 1: Shift the conduction bands in order to move the minimum of the lowest conduction band to zero energy.
shiftValenceBands	Specifies how to control <i>normalization</i> of the valence-band energies. Selecting this argument silently includes the Γ -point in the list of \mathbf{k} -vectors. Options are: <ul style="list-style-type: none">• 0: Leave band structure unmodified.• 1: Shift the entire band structure in order to move the maximum of the top valence band to zero energy (default).
showProgressBars	Specifies whether to display progress bars from the confined $\mathbf{k} \cdot \mathbf{p}$ solver. Options are: <ul style="list-style-type: none">• 0: Do not display progress bars (default).• 1: Display progress bars.
verbose	Specifies the level of output of additional status messages. Options are: <ul style="list-style-type: none">• 0: Minimal console output (default).• 1: Verbose output.
withLS	Specifies whether to use spin-orbit coupling. Options are: <ul style="list-style-type: none">• 0: Do not use spin-orbit coupling.• 1: Use spin-orbit coupling (default).
withNL	Specifies whether to use nonlocal potential terms. Options are: <ul style="list-style-type: none">• 0: Do not use nonlocal potential terms.• 1: Use nonlocal potential terms (default).

Examples

Retrieve the computed energies and their first derivatives (group velocities):

```
sBandGet derivOrder
```

Specify to compute the second derivatives of the band energies (reciprocal effective masses):

```
sBandSet derivOrder=2
```

EPM::Crystal Objects

This section discusses EPM::Crystal objects.

Creating Standard and Custom EPM::Crystal Objects

These commands create standard crystals, for example, bulk silicon, bulk germanium, and silicon germanium (use the `Crystal` command to define a custom crystal):

- [compute_SiGe_a0](#)
- [Crystal](#)
- [DiamondCrystal](#)
- [GermaniumCrystal](#)
- [SiGeCrystal](#)
- [SiliconCrystal](#)
- [StrainedCrystal](#)

compute_SiGe_a0

This command returns the relaxed lattice constant (in Å) of a SiGe crystal with germanium mole fraction x_{Ge} according to the formula [2]:

$$a_{\text{SiGe}} = a_{\text{Si}} + 0.200326 \text{ \AA} \cdot x_{\text{Ge}}(1 - x_{\text{Ge}}) + (a_{\text{Ge}} - a_{\text{Si}})x_{\text{Ge}}^2 \quad (239)$$

Syntax

```
compute_SiGe_a0 xGe
```

Argument	Description
xGe	Positional argument of type Double. Returns the germanium mole fraction of a SiGe crystal.

Examples

Return the relaxed lattice constant of $\text{Si}_{0.5}\text{Ge}_{0.5}$:

```
compute_SiGe_a0 0.5
```

Returns: 5.5350815

The relaxed lattice constant is 5.535 Å.

Crystal

This low-level command creates new custom EPM::Crystal objects. All the standard commands in this section rely on Crystal for the actual creation of new EPM::Crystal objects.

Syntax

```
Crystal name=String a0=Double  
[atoms=List/List#2] [primitiveBasis=List#3/RealVector3D]  
[strain=List#3/RealVector3D] [zeta=Double]
```

Argument	Description
a0	Sets the lattice constant for the new crystal (in Å).
atoms	Specifies a list of the species and positions of atoms within the unit cell of the crystal. Each list entry must have the form species=EPM::Crystal pos=List/RealVector3D where: <ul style="list-style-type: none">• species is the type of atoms to be inserted into the unit cell of the crystal.• pos is a list of 3D position vectors for the placement of the atoms (in units of a0).

Argument	Description
name	Sets both the command name and the handle string associated with the new EPM::Crystal object.
primitiveBasis	Each one of the three entries of this <i>List</i> must convert to a (real) <i>vector3D</i> . Together, these three vectors span the unit cell of the new crystal (in units of <i>a0</i>).
strain	If the atomic positions and the primitiveBasis supplied during construction of a Crystal object correspond to nonzero strain, then the corresponding strain tensor (3 × 3 matrix) should be supplied to the Crystal command for inclusion in the basis vectors of the reciprocal space. Default: zero strain
zeta	Sets the internal strain parameter.

Examples

Construct a bulk silicon crystal called bulk:

```
requireAtomicSpecies Si
Crystal name=bulk a0=[Si get latticeConstant] \
    primitiveBasis=[list {0 0.5 0.5} {0.5 0 0.5} {0.5 0.5 0}] \
    atoms=[list species=Si pos=[list {0.125 0.125 0.125} \
        {-0.125 -0.125 -0.125}]]
```

This is not entirely equivalent to the SiliconCrystal name=bulk command because the latter command also sets the internal strain parameter and the elasticity of the new EPM::Crystal object.

DiamondCrystal

This command creates a diamond structure crystal for the EPM band-structure calculation.

If no EPM::AtomicSpecies of name \$species exists when the DiamondCrystal command is called, then the atomic species is created automatically using the default parameters from the file \${species}_param.tcl.

Syntax

```
DiamondCrystal name=String species=EPM::AtomicSpecies
    [a0=Double]
```

Argument	Description
a0	Sets the lattice constant of the crystal (in Å). If a0 is not specified, then the reference lattice spacing of species is used.
name	Sets the name of the new EPM::Crystal object.
species	Specifies the atomic species to use for the new EPM::Crystal object.

Examples

```
DiamondCrystal name=bulk species=Si
```

This is equivalent to the SiliconCrystal name=bulk command.

GermaniumCrystal

This command calls DiamondCrystal to create a germanium crystal for the EPM band-structure calculation (see [DiamondCrystal on page 279](#)).

If no EPM::AtomicSpecies of name Ge exists when the GermaniumCrystal command is called, then the atomic species is created automatically using the default parameters from the file Ge_param.tcl.

Syntax

```
GermaniumCrystal name=String
```

Argument	Description
name	Sets the name of the new EPM::Crystal object.

Examples

Create a bulk Ge crystal called bulk using the default EPM parameters:

```
GermaniumCrystal name=bulk
```

SiGeCrystal

This command calls DiamondCrystal to create a $\text{Si}_{1-x}\text{Ge}_x$ crystal for the EPM band-structure calculation in the VCA, that is, each atom is treated as a statistical mixture of a silicon and a germanium atom (see [DiamondCrystal on page 279](#) and [Arguments for Building a Virtual Atomic Species on page 294](#)).

Chapter 14: Sentaurus Band Structure/Tcl Commands

EPM::Crystal Objects

The lattice constant is interpolated using the `compute_SiGe_a0` command (see [compute_SiGe_a0 on page 278](#)). The internal strain coefficient and the compliance tensor are interpolated linearly.

Syntax

```
SiGeCrystal name=String xGe=Double
```

Argument	Description
name	Sets the name of the new EPM::Crystal object.
xGe	Sets the germanium mole fraction.

Examples

Create a $\text{Si}_{0.7}\text{Ge}_{0.3}$ crystal with the name `SiGe30`:

```
SiGeCrystal name=SiGe30 xGe=0.3
```

SiliconCrystal

This command calls `DiamondCrystal` to create a silicon crystal for the EPM band-structure calculation (see [DiamondCrystal on page 279](#)).

If no `EPM::AtomicSpecies` of name `Si` exists when the `SiliconCrystal` command is called, then the atomic species is created automatically using the default parameters from the file `Si_param.tcl`.

Syntax

```
SiliconCrystal name=String
```

Argument	Description
name	Sets the name of the new EPM::Crystal object.

Examples

Create a bulk silicon crystal called `bulk` using the default EPM parameters:

```
SiliconCrystal name=bulk
```

StrainedCrystal

This command builds an `EPM::Crystal` object from an unstrained crystal, the strain tensor, and (optionally) the internal strain parameter.

The `EPM::Crystal` object is associated with the Tcl command `name` as described in the comment section of the `Crystal` command.

Typically, calling `StrainedCrystal` expressly is necessary only if multiple `EPM::Crystal` objects with different strain conditions need to coexist. In most other cases, it is expected that using the `<EPM::Crystal> set strain` method will result in simpler scripts (see [<EPM::Crystal> set on page 289](#)).

Syntax

```
StrainedCrystal name=String unstrained=EPM::Crystal
    [strain=RealMatrix3D] [zeta=Double]
```

Argument	Description
<code>name</code>	Sets the name of the new <code>EPM::Crystal</code> object.
<code>strain</code>	Sets the strain tensor used to deform <code>unstrained</code> into the final crystal. If <code>strain</code> is omitted, then it defaults to zero. Default: zero strain
<code>unstrained</code>	Sets the name of the underlying unstrained <code>EPM::Crystal</code> object from which the strained crystal will be built.
<code>zeta</code>	Sets the internal strain parameter of the crystal. If the object specified by <code>unstrained</code> already contains a valid value for the internal strain parameter, then <code>zeta</code> is optional; otherwise, it is required. If <code>zeta</code> is not specified, then its value is inherited from the specified unstrained crystal.

Examples

Create an unstrained silicon crystal called `unstrainedSi`, and then create a new crystal `strainedSi` by deforming the `unstrainedSi` crystal by applying 1% of biaxial (100) strain:

```
SiliconCrystal name=unstrainedSi
set strain [[unstrained get elasticity] biaxialStrain \
    dir=[list 0 1 1] inPlaneStrain=0.01
StrainedCrystal name=strainedSi unstrained=unstrainedSi strain=$strain
```

Using EPM::Crystal Objects

EPM::Crystal objects are accessed by using their names as Tcl command names. This is denoted by *<EPM::Crystal>*.

Using the object name as a Tcl command without arguments causes Sentaurus Band Structure to output a message that identifies the associated object. Alternatively, the first argument must specify a *method* that will be executed on the object.

Note:

Named argument parsing is used for the arguments to these methods.

The available methods are:

- [*<EPM::Crystal> addAtom*](#)
- [*<EPM::Crystal> apply*](#)
- [*<EPM::Crystal> computeBandstructure*](#)
- [*<EPM::Crystal> computeDOS*](#)
- [*<EPM::Crystal> destroy*](#)
- [*<EPM::Crystal> get*](#)
- [*<EPM::Crystal> scaleKvector*](#)
- [*<EPM::Crystal> set*](#)
- [*<EPM::Crystal> status*](#)
- [*<EPM::Crystal> unscaleKvector*](#)

<EPM::Crystal> addAtom

This method adds an atom to the unit cell of the crystal.

Syntax

<EPM::Crystal> addAtom species=EPM::AtomicSpecies pos=RealVector3D

Argument	Description
pos	Sets the position of the new atom. Unit: lattice spacing a_0
species	Sets the type of atom to be added to the unit cell.

Examples

```
source [sBandGet dbPath]/Si param.tcl      ;# load Si parameters
Crystal name=myCrystal                      ;# create an empty crystal
myCrystal set a0=5.43                         ;# lattice spacing is 5.43 Å
myCrystal addAtom species=Si pos=[list 0.125 0.125 0.125] ;# insert first atom
myCrystal addAtom species=Si pos=[list -0.125 -0.125 -0.125] ;# insert second atom
myCrystal status                             ;# show the crystal
```

Result:

```
myCrystal is an EPM::Crystal object.
lattice constant = 5.43 [AAngstroems]
cell spanning vectors: not yet defined.
atoms in the unit cell:
    species: "Si"
    {0.125 0.125 0.125}
    {-0.125 -0.125 -0.125}
```

<EPM::Crystal> apply

This method uses the elasticity of the material described by the `EPM::Crystal` object to apply certain strain conditions.

Syntax

```
<EPM::Crystal> apply method arg ...
```

Argument	Description
<code>method</code>	Specifies the name of a method of elasticity that returns a strain tensor, for example, <code>biaxialStrain</code> or <code>uniaxialStrain</code> . This must be the first argument.
<code>arg</code>	Specifies a list of arguments.

Description

The call:

```
<EPM::Crystal> apply method arg ...
```

is equivalent to:

```
<EPM::Crystal> set strain=[<EPM::Crystal> get elasticity]
method arg ...]
```

Special Case

```
<EPM::Crystal> apply biaxialStrain dir=vector3D | String  
substrate=EPM::Crystal | AnalyticBandSolver
```

This call computes the in-plane strain from the lattice mismatch between the EPM::Crystal object and the substrate material, and then replaces the substrate argument with inPlaneStrain=result before calling the *Elasticity* biaxialStrain method (see [Biaxial Strain on page 133](#) and [Elasticity biaxialStrain on page 310](#)).

For example, the following commands compute the strain in a pseudomorphic silicon film grown on an infinitely thick (100) Si_{0.7}Ge_{0.3} substrate:

```
SiliconCrystal name=silicon; SiGeCrystal name=SiGe30 xGe=0.3  
silicon apply biaxialStrain substrate=SiGe30 dir=[list 0 0 1]  
puts [silicon get strain]
```

Result:

```
{0.0113938 0 0} {0 0.0113938 0} {0 0 -0.00884294925374}
```

Examples

Apply 1 GPa of tensile stress along $\mathbf{d} = (1, 1, 0)/\sqrt{2}$ to a silicon crystal:

```
SiliconCrystal name=bulk  
bulk apply uniaxialStrain dir=[list 1 1 0] stress=1.0e9  
puts [bulk get strain]
```

Output:

```
{0.00274646443943 0.003125 0.0}  
{0.003125 0.00274646443943 0.0}  
{0.0 0.0 -0.00213158434105}
```

<EPM::Crystal> computeBandstructure

This method computes band-structure data.

Syntax

```
<EPM::Crystal> computeBandstructure  
{|kFile=String | kVectors=List/vector3D}  
[bandstructure=bandstructure_t]  
[groupVelocity=groupVelocity_t]  
[inverseMass=inverseMass_t] [nBands=Integer]  
[remainingOnly] [symmetry=List/List#3/Integer]
```

Argument	Description
bandstructure	Specifies the <code>bandstructure_t</code> container object for storing band-energy results. If this argument is omitted, then it defaults to <code>name.bandstructure</code> , where <code>name</code> is the name of the crystal object. If <code>name.bandstructure</code> does not exist, then it is created automatically.
groupVelocity	Specifies the <code>groupVelocity_t</code> container object for storing group-velocity results. If this argument is omitted, then it defaults to <code>name.groupVelocity</code> , where <code>name</code> is the name of the crystal object. If <code>name.groupVelocity</code> does not exist, then it is created automatically.
inverseMass	Specifies the <code>inverseMass_t</code> container object for storing reciprocal effective mass results. If this argument is omitted, then it defaults to <code>name.inverseMass</code> , where <code>name</code> is the name of the <code>EPM::Crystal</code> object. If <code>name.inverseMass</code> does not exist, then it is created automatically.
kFile	Sets the name of a file containing one k -vector per line as a space-separated list of three Doubles. Unit: $2\pi/a_{xyz}$
kVectors	Sets the k -vectors used for the band-structure calculation as a List of <code>vector3D</code> . If only a single k -vector is provided, then it does not need to be wrapped in a one-element list. Unit: $2\pi/a_{xyz}$
nBands	Sets the number of bands to compute. If spin-orbit coupling is active (see <code>sBandSet</code> in sBandGet and sBandSet on page 273), then the number of bands is multiplied internally by 2.
remainingOnly	If you specify this option, then the eigen-problem is solved only at k -vectors for which the containers of the results do not yet contain data.
symmetry	Specifies a list of symmetry operations (see determineSymmetry on page 314 for the notation). These symmetries can be used to reduce the computational effort of the band-structure calculation.

Examples

Set up a bulk silicon crystal and compute its band energies (in eV) at the zone center and near the conduction band minimum:

```
SiCrystal name=bulk
set Gamma {0 0 0} ; set Delta {0.85 0 0}
bulk computeBandstructure kVectors=[list $Gamma $Delta]
puts "Gamma: [bulk.bandstructure get kVector=$Gamma]"
puts "Delta: [bulk.bandstructure get kVector=$Delta]"
```

Output:

```
writing "energy.dat"
writing "velocity.dat"
Gamma: -12.4325779396 -0.0434084085113 0.0 0.0
3.34924141875 3.37860208567 3.37860208567 4.28964364273
Delta: -9.3238120997 -6.92864680226 -2.94277507525 -2.94061556233
1.06286616589 1.59472285825 10.8782585993 10.8892234861
```

<EPM::Crystal> computeDOS

This method computes density-of-states (DOS) data and returns it as a list of energy/DOS(energy) pairs and, optionally, writes it to a file. Band energies are computed automatically unless they are already present in the `bandstructure_t` container object. Units are for energy [eV] and for DOS [$\text{eV}^{-1}\text{cm}^{-3}$].

Note:

The factor 2 for spin degeneracy is not yet included.

Syntax

```
<EPM::Crystal> computeDOS nk= Integer carrierType= String
[algorithm= String] [band= Integer]
[bandstructure= bandstructure_t]
[DOSfile= String] [maxE= Double] [maxK= Double] [nE= Integer]
```

Argument	Description
algorithm	Selects the numeric algorithm used for the DOS calculation. Options are: <ul style="list-style-type: none">• <code>sccm</code>: State-counting method (default)• <code>tet</code>: Tetrahedral integration method, which is less susceptible to numeric noise than the state-counting method, especially at small energy intervals For backward compatibility, the default is the state-counting method.
band	Restricts the DOS computation to the band with this band index. Default: Use all bands
bandstructure	Specifies the <code>bandstructure_t</code> container object for storing and retrieving band-energy results. If this argument is omitted, then it defaults to <code>name.bandstructure</code> , where <code>name</code> is the name of the crystal object. If <code>name.bandstructure</code> does not exist, then it is created automatically.
carrierType	Sets the carrier type. Options are: <ul style="list-style-type: none">• <code>e</code> or <code>electron</code> for electron DOS• <code>h</code> or <code>hole</code> for hole DOS
DOSfile	Sets the file name for storing DOS data. Default: No file

Argument	Description
maxE	Sets the maximum energy for the DOS calculation. Default: 10 eV
maxK	Includes only \mathbf{k} -vectors with $ \mathbf{k} < \text{maxK}$ in the DOS calculation (unit: $2\pi/a_{xyz}$). Default: 2
nE	Sets the number of energy intervals for the DOS calculation. Default: 100
nk	Controls the density of the \mathbf{k} -vector grid used during the DOS calculation. It specifies the number of \mathbf{k} -vector intervals along the Δ -line (that is, between Γ and X).

Examples

Compute the hole DOS for unstrained bulk silicon and write it to the file dos_h.dat:

```
SiCrystal name=bulk  
bulk computeDOS nk=100 carrierType=h DOSfile=dos_h.dat
```

Because of the large number of \mathbf{k} -vectors, this example has a considerable runtime. A similar but faster example is shown in [<AnalyticBandSolver> computeDOS on page 304](#).

<EPM::Crystal> destroy

This method destroys the EPM::Crystal object. It has no arguments.

Syntax

```
<EPM::Crystal> destroy
```

<EPM::Crystal> get

This method provides read access to the parameters of the specified EPM::Crystal object.

Syntax

```
<EPM::Crystal> get a0 | cellVolume | elasticity | primitiveBasis |  
strain | zeta
```

Argument	Description
a0	Returns the lattice constant a_0 in Å.
cellVolume	Returns the volume of the unit cell in Å ³ .

Argument	Description
elasticity	Returns the name of the object describing the elasticity of the material.
primitiveBasis	Returns the spanning vectors of the unit cell as List#3/Vector3D.
strain	Returns the strain tensor as a 3×3 matrix.
zeta	Returns the internal strain parameter (typically, $0 < zeta < 1$).

Examples

Output the lattice constant of bulk germanium:

```
GermaniumCrystal name=bulkGe  
bulkGe get a0
```

<EPM::Crystal> scaleKvector

This method returns \mathbf{k} converted to units of $1/r_{\text{Bohr}}$. See [<EPM::Crystal> unscaleKvector on page 291](#).

Syntax

```
<EPM::Crystal> scaleKvector K=vector3D
```

Argument	Description
K	Sets a \mathbf{k} -vector in units of $2\pi/a_{x/y/z}$.

<EPM::Crystal> set

This method allows you to modify the parameters of the specified EPM::Crystal object.

Note:

You must specify exactly *one* of the supported arguments.

Syntax

```
<EPM::Crystal> set a0=Double | cellVolume=Double |  
elasticity=String | primitiveBasis=List#3/RealVector3D |  
strain=RealMatrix3D | zeta=Double
```

Argument	Description
a0	Sets the lattice constant a_0 in Å.
cellVolume	Sets the volume of the unit cell in Å ³ .
elasticity	Sets the name of an object describing the elasticity of the material.
primitiveBasis	Sets three vectors that span the unit cell (unit is lattice constant).
strain	Sets the strain tensor from a 3 × 3 matrix.
zeta	Sets the internal strain parameter (typically, 0 < zeta < 1).

Examples

This example completes the manual construction of a silicon crystal started in the example for [<EPM::Crystal> addAtom on page 283](#):

```
myCrystal set primitiveBasis=[list {0.0 0.5 0.5} {0.5 0.0 0.5}
{0.5 0.5 0.0}]
myCrystal status
```

Result:

```
myCrystal is an EPM::Crystal object.
lattice constant = 5.43 [AAngstroems]
internal strain parameter: 0.53
unit cell spanned by (unit: lattice constant):
{0 0.5 0.5}
{0.5 0 0.5}
{0.5 0.5 0}
atoms in the unit cell:
species: "Si"
{0.125 0.125 0.125}
{-0.125 -0.125 -0.125}
```

<EPM::Crystal> status

This method returns status information about the specified `EPM::Crystal` object. It has no arguments.

Syntax

```
<EPM::Crystal> status
```

Examples

```
SiliconCrystal name=bulkSi  
bulkSi status
```

Result:

```
"bulkSi" is an EPM::Crystal object.  
lattice constant = 5.43 [AAngstroems]  
internal strain parameter: 0.53  
unit cell spanned by (unit: lattice constant):  
{0 0.5 0.5}  
{0.5 0 0.5}  
{0.5 0.5 0}  
atoms in the unit cell:  
species: "Si"  
{0.125 0.125 0.125}  
{-0.125 -0.125 -0.125}
```

<EPM::Crystal> unscaleKvector

This method returns \mathbf{k} converted to units of $2\pi/a_{x/y/z}$. See [<EPM::Crystal> scaleKvector on page 289](#).

Syntax

```
<EPM::Crystal> unscaleKvector K=vector3D
```

Argument	Description
----------	-------------

K	Sets a \mathbf{k} -vector in units of $1/r_{\text{Bohr}}$.
---	---

EPM::AtomicSpecies Objects

This section discusses EPM::AtomicSpecies objects.

Creating and Loading EPM::AtomicSpecies Objects

These commands create and load EPM::AtomicSpecies objects.

AtomicSpecies

This command creates a new `EPM::AtomicSpecies` object. It has two variants for building:

- A *real* atomic species
- A *virtual* atomic species (that is, a statistical mixture of two atomic species)

Syntax

```
AtomicSpecies name=String
  { latticeConstant=Double muLS=Double nLS=Integer
    nValence=Integer zetaLS=Double A0=Double B0=Double
    R0=Double A2=Double R2=Double
    [Qcutoff=Double | Q2cutoff=Double] Vloc=Array |
    {a1=Double a2=Double a3=Double a4=Double
      a5=Double a6=Double [Vloc=Array]} |
    {VlocProc=String [Vloc=Array]} }
  |
  { moleFraction=Double species1=EPM::AtomicSpecies
    species2=EPM::AtomicSpecies }
  |
  { species1=EPM::AtomicSpecies preFactor1=Double
    species2=EPM::AtomicSpecies preFactor2=Double }
```

Arguments for Building a Real Atomic Species

This variant of the command builds an `EPM::AtomicSpecies` object of dynamic type `EPM::RealAtomicSpecies`.

Argument	Description
<code>A0</code>	Sets the energy prefactor (in Rydberg) for the nonlocal pseudopotential at $L = 0$.
<code>a1, a2, a3, a4, a5, a6</code>	If these optional arguments are present, then Sentaurus Band Structure uses the Friedel interpolation formula (see Friedel Interpolation Formula for Local Pseudopotential on page 160) instead of a cubic spline for interpolation of the local pseudopotential. The dynamic type of the <code>EPM::AtomicSpecies</code> object constructed in the presence of <code>a1</code> to <code>a6</code> is <code>EPM::RealAtomicSpecies_Friedel</code> . Units: Hartree atomic units, that is, a_1 in Hartree; a_2, a_4, a_5, a_6 in $1/r_{\text{Bohr}}^2$; a_3 in r_{Bohr}^2
<code>A2</code>	Sets the energy prefactor (in Rydberg) for the nonlocal pseudopotential at $L = 2$.
<code>B0</code>	Sets the energy-dependent correction (unit: 1) to the prefactor <code>A0</code> for the nonlocal $L = 0$ pseudopotential: $\Delta A_0(\mathbf{K}_1, \mathbf{K}_2) = \frac{\hbar^2}{2m_0} (\ \mathbf{K}_1\ \cdot \ \mathbf{K}_2\ - \ \mathbf{k}_F\ ^2) B_0$

Chapter 14: Sentaurus Band Structure/Tcl Commands

EPM::AtomicSpecies Objects

Argument	Description
latticeConstant	Sets the lattice constant a_0 of the reference crystal for the new EPM::RealAtomicSpecies object.
muLS	Sets the energy parameter for spin-orbit splitting (in Rydberg).
name	Sets the name of the new EPM::RealAtomicSpecies object.
nLS	Sets the principal quantum number of the core shell that dominates the spin-orbit interaction (usually, one less than the principal quantum number of the species).
nValence	Sets the number of valence electrons.
Qcutoff, Q2cutoff	Sets the maximum $\ q\ $ (in $2\pi/a$) or $\ q\ ^2$ (in $(2\pi/a)^2$) for evaluating the local part of the atomic form factor. In strained crystals, the norm is computed for the corresponding unstrained wavevector. Default: -1 (that is, inactive; use the global cut-off value; compare to sBandGet and sBandSet on page 273) If both the per-species and the global Qcutoff or Q2cutoff arguments are inactive (that is, they have negative values), then an error condition will occur during the band-structure calculation.
R0	Sets the well radius (in Å) for the nonlocal pseudopotential at $L = 0$.
R2	Sets the well radius (in Å) for the nonlocal pseudopotential at $L = 2$.
Vloc (optional if either a1 to a6, or VlocProc are supplied)	Sets the name of a Tcl array. The key of this field is the modulus squared of a reciprocal lattice vector (unit: $(2\pi/a_0)^2$). The value type is the local pseudopotential at this wavevector in Rydberg.
VlocProc (optional parameter for local potential from Tcl procedure)	Specifies the name of a Tcl procedure used to evaluate the local pseudopotential form factor. This procedure must take a single argument, the magnitude of the crystal-momentum transfer vector $\ q\ $ (in $1/r_{\text{Bohr}}$). Before returning the value, the local pseudopotential must be multiplied by the unit cell volume per ion Ω_a . The return value of the procedure is, therefore, in Hartree. An example of VlocProc is given in Arguments for Combining Pseudopotentials of Two Species .
zetaLS	Sets the length scale parameter for spin-orbit splitting in $1/r_{\text{Bohr}}$.

Arguments for Building a Virtual Atomic Species

This variant of the command builds an `EPM::AtomicSpecies` object of dynamic type `EPM::VirtualAtomicSpecies`. Such an object represents a statistical mixture of two atomic species in the spirit of the VCA, that is, the atomic form factor is interpolated linearly between the two species involved.

Argument	Description
<code>moleFraction</code>	Sets the mole fraction of <code>species2</code> in the virtual species.
<code>name</code>	Sets the name of the new <code>EPM::VirtualAtomicSpecies</code> object.
<code>species1</code>	Sets the first constituent species of the virtual species.
<code>species2</code>	Sets the second constituent species of the virtual species.

Arguments for Combining Pseudopotentials of Two Species

This variant of the command builds an `EPM::AtomicSpecies` object of dynamic type `EPM::VirtualAtomicSpecies`. Such an object represents a statistical mixture of two atomic species in the spirit of the VCA, that is, the atomic form factors of two species are combined linearly with two weighting factors.

Argument	Description
<code>name</code>	Sets the name of the new <code>EPM::VirtualAtomicSpecies</code> object.
<code>preFactor1</code>	Sets the weighting factor associated with <code>species1</code> .
<code>preFactor2</code>	Sets the weighting factor associated with <code>species2</code> .
<code>species1</code>	Sets the first constituent species of the virtual species.
<code>species2</code>	Sets the second constituent species of the virtual species.

Examples

Build a virtual atomic species for $\text{Si}_{0.7}\text{Ge}_{0.3}$:

```
requireAtomicSpecies Si
requireAtomicSpecies Ge
AtomicSpecies name=SiGe_30 species1=Si species2=Ge moleFraction=0.3
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

EPM::AtomicSpecies Objects

Combine local and nonlocal pseudopotential form factors:

```
# Local pseudopotential form factor is created first.  
AtomicSpecies name=Loc VlocProc=myVlocProc Q2cutoff=1000  
  
# Nonlocal pseudopotential form factor is created next.  
AtomicSpecies name=nonLoc Q2cutoff=0.0 Vloc=emptyVloc \  
A0=$A0 B0=$B0 R0=$R0 A2=$A2 R2=$R2 zetaLS=$zetaLS muLS=$muLS  
  
# Combine local and nonlocal pseudopotential form factors.  
AtomicSpecies name=Loc_nonLoc species1=Loc preFactor1=1.0 \  
species2=nonLoc preFactor2=1.0
```

Here, the local and nonlocal pseudopotential form factors are multiplied by the weighting factors `preFactor1` and `preFactor2`, respectively (all 1 in this example).

This combination is useful when you want to build an atomic species with a customized local pseudopotential form factor. In the previous example, you can use your own form factor for the local pseudopotential as a function of $\|q\|$, which is specified by the procedure `myVlocProc`. You should provide this procedure. An example of the procedure `myVlocProc` is:

```
#####  
# Note:  
# - This procedure must have only 1 argument q  
# - Unit of q is 1/rBohr  
#####  
proc myVlocProc {q} {  
    # Declare global variables used in this procedure.  
    global rBohr; global PI; global a0  
  
    # Convert unit of q from 1/rBohr to 2pi/a0.  
    set unscale_q [expr $q / ($PI * 2.0 * $rBohr / $a0)]  
  
    # Calculate local pseudopotential for a given unscale_q.  
    set Vloc [getVloc $unscale_q]  
  
    # Calculate unit cell volume per ion.  
    set Vion [expr 0.125 * pow($a0 / $rBohr,3)]  
  
    # Multiply Vloc and Vion, and return.  
    return [expr $Vloc*$Vion]  
}
```

requireAtomicSpecies

This command provides an automatic loading mechanism for the default atomic species from parameter files.

If an EPM::AtomicSpecies object called *name* is not yet present, then Sentaurus Band Structure searches its database directory (compare to dbPath in [sBandGet and sBandSet on page 273](#)) for a file called *name_param.tcl*.

If such a file exists, then it is source'd into the Sentaurus Band Structure/Tcl interpreter. Afterwards, Sentaurus Band Structure rechecks for the existence of the atomic species *name*. If the species still does not exist, then an error message is generated.

Syntax

```
requireAtomicSpecies name
```

Argument	Description
name	Positional argument of type String.

Examples

Return the Si parameters from the file *your-Sentaurus-Band-Structure-database-path/Si_param.tcl*:

```
requireAtomicSpecies Si
```

Using EPM::AtomicSpecies Objects

EPM::AtomicSpecies objects are accessed by using their names as Tcl command names. This is denoted by <EPM::AtomicSpecies>.

Using the object name as a Tcl command without arguments causes Sentaurus Band Structure to output a message that identifies the associated object. Alternatively, the first argument must specify a *method* that will be executed on the object.

Note:

Named argument parsing is used for the arguments to these methods.

The available methods are:

- <EPM::AtomicSpecies> destroy
- <EPM::AtomicSpecies> dynamicType
- <EPM::AtomicSpecies> get

- <EPM::AtomicSpecies> rename
- <EPM::AtomicSpecies> set
- <EPM::AtomicSpecies> status

<EPM::AtomicSpecies> destroy

This method destroys the EPM::AtomicSpecies object. It has no arguments.

Syntax

```
<EPM::AtomicSpecies> destroy
```

<EPM::AtomicSpecies> dynamicType

This method returns the dynamic type of the associated EPM::AtomicSpecies object, that is, one of the subclasses:

- EPM::RealAtomicSpecies
- EPM::RealAtomicSpecies_Friedel
- EPM::RealAtomicSpecies_VlocProc
- EPM::VirtualAtomicSpecies

It has no arguments.

Syntax

```
<EPM::AtomicSpecies> dynamicType
```

Examples

```
source [sBandGet dbPath]/Si_param.tcl
puts "Si has dynamic type: [Si dynamicType]"
source [sBandGet dbPath]/Ge_param.tcl
AtomicSpecies name=SiGe species1=Si species2=Ge moleFraction=0.3
puts "SiGe has dynamic type: [SiGe dynamicType]"
```

Result:

```
Si has dynamic type: EPM::RealAtomicSpecies
SiGe has dynamic type: EPM::VirtualAtomicSpecies
```

<EPM::AtomicSpecies> get

This method returns the value of one of the arguments of an EPM::AtomicSpecies object (all arguments of the AtomicSpecies command except name, species1, species2, preFactor1, and preFactor2). See [AtomicSpecies on page 292](#).

Chapter 14: Sentaurus Band Structure/Tcl Commands

EPM::AtomicSpecies Objects

If an argument is not accessible because of the incorrect dynamic type of the associated object, then the command fails.

Syntax

```
<EPM::AtomicSpecies> get A0 | a1 | a2 | a3 | a4 | a5 | a6 | A2 |
    B0 | latticeConstant | moleFraction | muLS | nLS |
    nValence | Qcutoff | Q2cutoff | R0 | R2 |
    Vloc | VlocProc | zetaLS
```

Argument	Description
A0	Returns the energy prefactor (in Rydberg) for the nonlocal pseudopotential at $L = 0$.
a1, a2, a3, a4, a5, a6	Returns the value of these arguments as used in the Friedel interpolation formula (see Friedel Interpolation Formula for Local Pseudopotential on page 160).
A2	Returns the energy prefactor (in Rydberg) for the nonlocal pseudopotential at $L = 2$.
B0	Returns the energy-dependent correction (unit: 1) to the prefactor A0 for the nonlocal $L = 0$ pseudopotential.
latticeConstant	Returns the lattice constant a_0 of the reference crystal for the EPM::RealAtomicSpecies object.
moleFraction	Returns the mole fraction of species2 in the virtual species.
muLS	Returns the energy parameter for spin-orbit splitting (in Rydberg).
nLS	Returns the principal quantum number of the core shell that dominates the spin-orbit interaction.
nValence	Returns the number of valence electrons.
Qcutoff, Q2cutoff	Returns the maximum $\ q\ $ (in $2\pi/a$) or $\ q\ ^2$ (in $(2\pi/a)^2$) for evaluating the local part of the atomic form factor.
R0	Returns the well radius (in Å) for the nonlocal pseudopotential at $L = 0$.
R2	Returns the well radius (in Å) for the nonlocal pseudopotential at $L = 2$.

Argument	Description
vloc	Returns the name of a Tcl array. Note: Because of the special treatment of arrays in Tcl, the <code><EPM::AtomicSpecies> get vloc</code> method cannot directly return a Tcl array (in Tcl, arrays are not variables). Instead, it returns a List/List#2/Double suitable for setting an array using the Tcl command <code>array set</code> . <code>vloc</code> accepts an optional argument <code>q2</code> . If <code>q2</code> is present, then the command returns the interpolated local pseudopotential $V_{\text{loc}}(\ \mathbf{q}\ ^2 = q_2)$.
vlocProc	Returns the name of a Tcl procedure used to evaluate the local pseudopotential form factor.
zetaLS	Returns the length scale parameter for spin-orbit splitting in $1/r_{\text{Bohr}}$.

<EPM::AtomicSpecies> rename

This method changes the Tcl command name that is the handle associated with the EPM::AtomicSpecies object and the `name` argument of the EPM::AtomicSpecies object to `newName`.

Syntax

`<EPM::AtomicSpecies> rename newName=String`

Argument	Description
newName	Sets the name of the new Tcl command and handle string.

<EPM::AtomicSpecies> set

This method allows you to modify one or more arguments of the associated EPM::AtomicSpecies object (all arguments of the `AtomicSpecies` command except `name`, `species1`, `species2`, `preFactor1`, and `preFactor2`). See [AtomicSpecies on page 292](#).

To change the name of an EPM::AtomicSpecies object, see [`<EPM::AtomicSpecies> rename`](#).

If an argument is not accessible because of an incorrect dynamic type of the associated object, then the command fails.

Syntax

```
<EPM::AtomicSpecies> set  
  [A0=Double] [a1=Double] [a2=Double] [a3=Double]  
  [a4=Double] [a5=Double] [a6=Double]  
  [A2=Double] [B0=Double]  
  [latticeConstant=Double] [muLS=Double]  
  [nLS=Integer] [nValence=Integer]  
  { [moleFraction=Double] | {[Qcutoff=Double] | Q2cutoff=Double} } }  
  [R0=Double] [R2=Double] [Vloc=Array] [VlocProc=String]  
  [zetaLS=Double]
```

For information about the arguments, see [AtomicSpecies on page 292](#).

<EPM::AtomicSpecies> status

This method outputs status information about the EPM::AtomicSpecies object. It has no arguments.

Syntax

```
<EPM::AtomicSpecies> status
```

Examples

```
requireAtomicSpecies Ge      ;# load germanium parameters  
Ge status
```

Output:

```
species name = "Ge"  
dynamic type: EPM::RealAtomicSpecies  
nValenceElectrons: 4  
Vloc entries assume a lattice constant of 5.65 AAngstroems.  
Vloc(3) = -0.221 [Ry]  
Vloc(8) = 0.019 [Ry]  
Vloc(11) = 0.056 [Ry]  
Qcutoff = 3.52704 (Q2cutoff = 12.44) ;# not overridden by global Qcutoff  
volume = 22.5453 [AAngstroem^3]  
muLS = 0.000965 [Ry]  
zetaLS = 5.34 [1/rBohr]  
principal quantum number: 3  
nonLocalWell = Gaussian  
A0 = 0 [Ry]  
B0 = 0 [1]  
R0 = 0 [AAngstroem]  
A2 = 0.275 [Ry]  
R2 = 1.22 [AAngstroem]
```

AnalyticBandSolver Objects

This section discusses AnalyticBandSolver objects.

Creating AnalyticBandSolver Objects

This section discusses the commands to create AnalyticBandSolver objects.

AnalyticBandSolver

This command creates a new AnalyticBandSolver object.

The `name` argument becomes the name of a new Tcl command for accessing the new AnalyticBandSolver object (see [Using AnalyticBandSolver Objects on page 302](#)).

Syntax

```
AnalyticBandSolver name=String
```

Argument	Description
<code>name</code>	Sets the name of the new AnalyticBandSolver object.

AnalyticBandSolverFromSpecies

This command creates an AnalyticBandSolver object for a given atomic species by using information in the parameter files of the atomic species distributed with Sentaurus Band Structure, which contains EPM parameters, elasticity data, and templates for AnalyticBandSolver objects that approximate the EPM band structures.

Syntax

```
AnalyticBandSolverFromSpecies name=String species=EPM::AtomicSpecies
```

Argument	Description
<code>name</code>	Sets the name of the new AnalyticBandSolver object.
<code>species</code>	Specifies the species from which to use parameters in its parameter file.

Chapter 14: Sentaurus Band Structure/Tcl Commands

AnalyticBandSolver Objects

Examples

Build an AnalyticBandSolver object suitable for silicon:

```
AnalyticBandSolverFromSpecies name=analytic species=Si
```

Using AnalyticBandSolver Objects

AnalyticBandSolver objects are accessed by using their names as Tcl command names. This is denoted by `<AnalyticBandSolver>`.

Using the object name as a Tcl command without arguments causes Sentaurus Band Structure to output a message that identifies the associated object. Alternatively, the first argument must specify a *method* that will be executed on the object.

Note:

Named argument parsing is used for the arguments to the methods.

The available methods are:

- `<AnalyticBandSolver> apply`
- `<AnalyticBandSolver> computeBandstructure`
- `<AnalyticBandSolver> computeDOS`
- `<AnalyticBandSolver> copy`
- `<AnalyticBandSolver> destroy`
- `<AnalyticBandSolver> get`
- `<AnalyticBandSolver> set`
- `<AnalyticBandSolver> status`

<AnalyticBandSolver> apply

This method uses the elasticity of the material described by the `AnalyticBandSolver` object to apply certain strain conditions.

The call:

```
<AnalyticBandSolver> apply method arg ...
```

is equivalent to:

```
<AnalyticBandSolver> set \
    strain=[[<AnalyticBandSolver> get elasticity] method arg ...
```

Syntax

```
<AnalyticBandSolver> apply method arg ...
```

Argument	Description
<code>method</code>	Specifies the name of a method of elasticity that returns a strain tensor, for example, <code>biaxialStrain</code> or <code>uniaxialStrain</code> . This must be the first argument.
<code>arg</code>	Specifies a list of arguments.

Examples

Apply 1 GPa of tensile stress along $\mathbf{d} = (1, 1, 0)/\sqrt{2}$ to an `AnalyticBandSolver` object for silicon:

```
AnalyticBandSolverFromSpecies name=kp species=Si
kp apply uniaxialStrain dir=[list 1 1 0] stress=1.0e9
puts [kp get strain]
```

Output:

```
{0.00274646443943 0.003125 0.0}
{0.003125 0.00274646443943 0.0}
{0.0 0.0 -0.00213158434105}
```

<AnalyticBandSolver> computeBandstructure

This command computes band-structure data.

Syntax

```
<AnalyticBandSolver> computeBandstructure
    [kFile=String | kVectors=List/Vector3D]
    [bandstructure=bandstructure_t] [groupVelocity=groupVelocity_t]
    [remainingOnly] [symmetry=List/List#3/Integer]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

AnalyticBandSolver Objects

Argument	Description
bandstructure	Specifies the <code>bandstructure_t</code> container object for storing the band-structure result. If this argument is omitted, then it defaults to <code>name.bandstructure</code> , where <code>name</code> is the name of the crystal object. If <code>name.bandstructure</code> does not exist, then it is created automatically.
groupVelocity	Specifies the <code>groupVelocity_t</code> container object for storing the group-velocity results. If this argument is omitted, then it defaults to <code>name.groupVelocity</code> , where <code>name</code> is the name of the crystal object. If <code>name.groupVelocity</code> does not exist, then it is created automatically.
kFile	Sets the name of a file that contains one k -vector per line as a space-separated list of three Doubles. Unit: $2\pi/a_{x/y/z}$
kVectors	Specifies the k -vectors to be used for the band-structure calculation as a list of real 3D vectors. Unit: $2\pi/a_{x/y/z}$
remainingOnly, symmetry	These arguments are supported only to keep the interface compatible with the <code><EPM::Crystal> computeBandstructure</code> method (see <EPM::Crystal> computeBandstructure on page 285). When used with <code><AnalyticBandSolver> computeBandstructure</code> , these arguments have no effect.

<AnalyticBandSolver> computeDOS

This command computes DOS data and returns it as a list of energy/DOS(energy) pairs and, optionally, writes it to a file. Band energies are computed automatically unless they are already present in the specified `bandstructure_t` container object. Units are for energy [eV] and for DOS [$\text{eV}^{-1}\text{cm}^{-3}$].

Note:

The factor 2 for spin degeneracy is not yet included.

Syntax

```
<AnalyticBandSolver> computeDOS nk=Integer carrierType=String  
[algorithm=String] [band=Integer]  
[bandstructure=bandstructure_t] [DOSfile=String]  
[maxE=Double] [maxK=Double] [nE=Integer]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

AnalyticBandSolver Objects

Argument	Description
algorithm	Selects the numeric algorithm used for the DOS calculation. Options are: <ul style="list-style-type: none">• scm: State-counting method (default)• tet: Tetrahedral integration method, which is less susceptible to numeric noise than the state-counting method, especially at small energy intervals For backward compatibility, the default is the state-counting method.
band	Restricts the DOS computation to the band with this band index. Default: Use all bands
bandstructure	Specifies the <code>bandstructure_t</code> container object for storing and retrieving band-energy results. If this argument is omitted, then it defaults to <code>name.bandstructure</code> , where <code>name</code> is the name of the crystal object. If <code>name.bandstructure</code> does not exist, then it is created automatically.
carrierType	Sets the carrier type. Options are: <ul style="list-style-type: none">• e or electron for electron DOS• h or hole for hole DOS
DOSfile	Sets the file name for storing DOS data. Default: No file
maxE	Sets the maximum energy for the DOS calculation. Default: 10 eV
maxK	Includes only k -vectors with $ k < \text{maxK}$ in the DOS calculation (unit: $2\pi/a_{xyz}$). Default: 2
nE	Sets the number of energy intervals for the DOS calculation. Default: 100
nk	Controls the density of the k -grid used during the DOS calculation. It specifies the number of k -intervals along the Δ -line (that is, between Γ and X).

Examples

Compute the hole DOS for unstrained bulk silicon using six-band, restrict the **k**-vector range to vectors shorter than 0.2 (in the $2\pi/a_{xyz}$ metric), and write it to the file `dos_h.dat`:

```
AnalyticBandSolverFromSpecies name=kpSi species=Si  
kpSi computeDOS nk=100 carrierType=h DOSfile=dos_h.dat maxK=0.2
```

<AnalyticBandSolver> copy

This method duplicates an `AnalyticBandSolver` object.

Syntax

```
<AnalyticBandSolver> copy newName=String
```

Argument	Description
newName	Sets the name of the copy of the <code>AnalyticBandSolver</code> object.

<AnalyticBandSolver> destroy

This method destroys the `AnalyticBandSolver` object. It has no arguments.

Syntax

```
<AnalyticBandSolver> destroy
```

<AnalyticBandSolver> get

This method returns the value of one of the arguments of the analytic band-structure solver.

Syntax

```
<AnalyticBandSolver> get a_v | a0 | b | conductionBands |  
d | dbs | Delta | el_alpha | elasticity |  
gamma1 | gamma2 | gamma3 | K0 | kp_alpha | M | method |  
ml | mt | strain | valenceBands | Xi_d | Xi_s | Xi_u
```

For information about the arguments, see [<AnalyticBandSolver> set on page 306](#).

<AnalyticBandSolver> set

This method sets one or more arguments of the analytic band-structure solver.

Syntax

```
<AnalyticBandSolver> set  
[a_v=Double] [a0=Double] [b=Double] [conductionBands=Boolean]  
[d=Double] [dbs=Double] [Delta=Double]  
[el_alpha=List#2/Double] [elasticity=Elasticity]  
[gamma1=Double] [gamma2=Double] [gamma3=Double]  
[K0=RealVector3D] [kp_alpha=List#2/Double] [M=Double]  
[method=Integer] [ml=Double] [mt=Double]  
[strain=RealMatrix3D] [valenceBands=Boolean]  
[Xi_d=Double] [Xi_s=Double] [Xi_u=Double]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

AnalyticBandSolver Objects

Argument	Description
General	
a0	Sets the relaxed lattice spacing in Å.
conductionBands	Specifies whether to compute the conduction bands. Options are: <ul style="list-style-type: none">• 0: Do not compute conduction bands (default).• 1: Compute conduction bands.
elasticity	Sets the name of an <code>Elasticity</code> object in which to store elastic properties.
strain	Sets the strain tensor as a 3×3 matrix represented by a list of 3D row vectors.
valenceBands	Specifies whether to compute the valence bands. Options values are: <ul style="list-style-type: none">• 0: Do not compute valence bands.• 1: Compute valence bands (default).
Specific to conduction band	
dbs	Sets the energy difference between first and second conduction bands at the band minimum.
el_alpha	Sets the nonparabolicity [eV^{-1}] for the ellipsoidal model. The first list entry applies to the lowest conduction band; the second list entry applies to the second lowest conduction band.
k0	Sets the position of band minima (unit: $2\pi/a_{x/y/z}$). This is <i>not</i> a vector. The three entries of <code>k0</code> are the k -space distances between the X -points along the $x/y/z$ directions and the corresponding Δ -valley minimum.
kp_alpha	Sets the nonparabolicity [eV^{-1}] for the two-band $\mathbf{k}\cdot\mathbf{p}$ model. The first list entry applies to the lowest conduction band; the second list entry applies to the second lowest conduction band.
M	Sets the Sverdlov $\mathbf{k}\cdot\mathbf{p}$ parameter [3].
method	Sets the method to use. Options are: <ul style="list-style-type: none">• 0: Use two-band $\mathbf{k}\cdot\mathbf{p}$ (default).• 1: Use ellipsoidal bands.
m1	Sets the longitudinal electron mass (unit: m_0).

Argument	Description
mt	Sets the transverse electron mass (unit: m_0).
xi_d, xi_s, xi_u	Sets the conduction-band deformation potentials in eV.
Specific to valence band	
a_v, b, d	Sets the Bir–Pikus deformation potentials in eV.
Delta	Sets the spin-orbit splitting energy in eV.
gamma1, gamma2, gamma3	Sets the Luttinger band parameters in atomic units.

<AnalyticBandSolver> status

This method outputs status information about the `AnalyticBandSolver` object. It has no arguments.

Syntax

```
<AnalyticBandSolver> status
```

Elasticity Objects

Sentaurus Band Structure uses `Elasticity` objects to describe the elastic properties of materials. The information stored in an `Elasticity` object consists of a symmetry class and a sequence of elasticities s_{ij} (in %/GPa) that uniquely defines the compliance tensor S .

Creating Elasticity Objects

`Elasticity` objects are created by the `Elasticity` command.

Elasticity

This command creates a new `Elasticity` object. The `name` argument becomes the name of a new Tcl command for accessing the new `Elasticity` object (see [Using Elasticity Objects on page 309](#)).

Syntax

```
Elasticity name=String  
{ type=Cubic  
{ c11=Double c12=Double c44=Double } |
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Elasticity Objects

```
{ s11=Double s12=Double s44=Double }
}
{
  type=Wurtzite
  {
    c11=Double c12=Double c13=Double c33=Double c44=Double } |
  {
    s11=Double s12=Double s13=Double s33=Double s44=Double }
}
{
  e1=Elasticity e2=Elasticity x2=Double }
```

Argument	Description
c11, c12, c44	Sets the components of the compliance tensor (unit: GPa/% = Mbar). Only Cubic type.
c11, c12, c13, c33, c44	Sets the components of the compliance tensor (unit: GPa/% = Mbar). Only Wurtzite type.
e1, e2, x2	A new <code>Elasticity</code> object can be constructed by linear interpolation between two existing <code>Elasticity</code> objects, <code>e1</code> and <code>e2</code> . $x2 \in [0,1]$ is the statistical weight of <code>e2</code> in the interpolated <code>Elasticity</code> object. Note: Both <code>Elasticity</code> objects must be of the same <code>type</code> .
name	Sets the name of the new <code>Elasticity</code> object.
s11, s12, s44	Sets the components of the stiffness tensor (unit: %/GPa = 1/Mbar). Only Cubic type.
s11, s12, s13, s33, s44	Sets the components of the stiffness tensor (unit: %/GPa = 1/Mbar). Only Wurtzite type.
type	Selects the symmetry class of the crystal. Options are <code>Cubic</code> or <code>Wurtzite</code> .

Using Elasticity Objects

Elasticity objects are accessed by using their names as Tcl command names. This is denoted by `<Elasticity>`.

Using the object name as a Tcl command without arguments causes Sentaurus Band Structure to output a message that identifies the associated object. Alternatively, the first argument must specify a *method* that will be executed on the object. The available methods are:

- `<Elasticity> biaxialStrain`
- `<Elasticity> biaxialStrainRatio`
- `<Elasticity> copy`

Chapter 14: Sentaurus Band Structure/Tcl Commands

Elasticity Objects

- <Elasticity> destroy
- <Elasticity> dynamicType
- <Elasticity> status
- <Elasticity> strainFromStress
- <Elasticity> uniaxialStrain

<Elasticity> biaxialStrain

This method returns the biaxial strain tensor for a pseudomorphic material layer epitaxially grown on an infinitely thick substrate.

Note:

Only available for cubic elasticity.

Syntax

```
<Elasticity> biaxialStrain dir=vector3D | String  
inPlaneStrain=Double
```

Argument	Description
dir	Sets the normal direction of the interface between substrate and material as vector3D. For backward compatibility, the fixed strings 100 and 111 are recognized as synonyms for the normal vectors {0 0 1} and {1 1 1}, respectively.
inPlaneStrain	Selects the strain parallel to the material boundary between the pseudomorphic layer and substrate. Options are: <ul style="list-style-type: none">Positive values: Tensile strainNegative values: Compressive strain

Examples

```
Elasticity name=E type=Cubic c11=1.675 c12=0.65 c44=0.8  
E biaxialStrain dir=100 inPlaneStrain=0.01
```

Result:

```
{0.01 0 0} {0 0.01 0} {0 0 -0.00776119402986}
```

<Elasticity> biaxialStrainRatio

This method returns the ratio between out-of-plane and in-plane strain for a pseudomorphic material layer epitaxially grown on an infinitely thick substrate.

Note:

Only available for cubic elasticity.

Syntax

```
<Elasticity> biaxialStrainRatio dir=String
```

Argument	Description
-----------------	--------------------

dir	Sets the direction of the interface between <code>substrate</code> and <code>material</code> . Options are <code>100</code> and <code>111</code> .
-----	--

Examples

```
Elasticity name=E type=Cubic s11=0.762 s12=-0.213 s44=1.25  
E biaxialStrainRatio dir=111
```

The result is `-0.445643793369`: 1% tensile in-plane strain leads to 0.446% compressive strain normal to the `(110)` interface.

<Elasticity> copy

This method duplicates an `Elasticity` object.

Syntax

```
<Elasticity> copy newName=String
```

Argument	Description
-----------------	--------------------

newName	Sets the name of the copy of the <code>Elasticity</code> object.
---------	--

<Elasticity> destroy

This method destroys the specified object. It has no arguments.

Syntax

```
<Elasticity> destroy
```

<Elasticity> dynamicType

This method returns the dynamic type of an `Elasticity` object. It has no arguments.

Syntax

```
<Elasticity> dynamicType
```

<Elasticity> status

This method outputs status information about the `Elasticity` object. It has no arguments.

Syntax

```
<Elasticity> status
```

<Elasticity> strainFromStress

This method returns the strain tensor that corresponds to the stress tensor specified in the `stress` argument.

Note:

Both the stress tensor and the resulting strain tensor use the principal-axis coordinate system of the crystal, which typically differs from the process or device coordinate systems of Sentaurus Process or Sentaurus Device.

Syntax

```
<Elasticity> strainFromStress stress=RealMatrix3D
```

Argument	Description
<code>stress</code>	Specifies the stress tensor represented as a list of 3D row vectors (unit: Pa).

Examples

For a cubic crystal:

```
Elasticity name=E type=Cubic c11=1.675 c12=0.650 c44=0.8
set stress {{1.0e9 0.0 0.0} {0.0 0.0 0.0} {0.0 0.0 0.0}}
E strainFromStress stress=$stress
```

Result:

```
{0.00762451321992 0.0 0.0}
{0.0 -0.00213158434106 0.0}
{0.0 0.0 -0.00213158434106}
```

For a wurtzite crystal:

```
Elasticity name=E type=Wurtzite \
c11=3.90 c12=1.45 c13=1.06 c33=3.98 c44=1.05
set stress {{1.0e9 0.0 0.0} {0.0 0.0 0.0} {0.0 0.0 0.0}}
E strainFromStress stress=$stress
```

Result:

```
{0.003085666109684904 0.0 0.0}
{0.0 -0.0009959665433763208 0.0}
{0.0 0.0 -0.000556553150825904}
```

<Elasticity> uniaxialStrain

This method returns the strain tensor resulting from uniaxial `stress` along direction `dir`.

Note:

Only available for cubic elasticity.

Syntax

```
<Elasticity> uniaxialStrain dir=RealVector3D stress=Double
```

Argument	Description
dir	Sets the direction of uniaxial stress as a three-component vector. This vector does not need to be normalized.
stress	Specifies the stress in direction <code>dir</code> in Pa. Options are: <ul style="list-style-type: none">Positive values: Tensile stressNegative values: Compressive stress

Examples

```
SiliconCrystal name=Silicon
[ Silicon get elasticity] uniaxialStrain dir=[list 1 1 1] stress=1.0e9
```

Result:

```
{0.00112044817927 0.00208333333333 0.00208333333333}
{0.00208333333333 0.00112044817927 0.00208333333333}
{0.00208333333333 0.00208333333333 0.00112044817927}
```

Band-Structure Operations

This section presents operations on band structures.

determineSymmetry

This command returns a list of symmetries for use by [<EPM::Crystal> computeBandstructure on page 285](#).

These symmetries are generators of the symmetry group that results from deforming a crystal of class `crystalClass` by applying `strain`. Each symmetry is represented by a list of three integers that describe permutations and sign flips of the coordinates of 3D \mathbf{k} -space.

The symmetry $\{l\ m\ n\}$ maps the \mathbf{k} -vector $(k_1\ k_2\ k_3)$ to $(\text{sgn}(l)k_{|l|}\ \text{sgn}(m)k_{|m|}\ \text{sgn}(n)k_{|n|})$.

Syntax

```
determineSymmetry crystalClass=String strain=Matrix3D
```

Argument	Description
<code>crystalClass</code>	Selects the name of the symmetry class of the crystal. Options are: <ul style="list-style-type: none">"cubic": The relaxed crystal has cubic symmetry."generic": Use if the crystal class is not known. Only Kramer's symmetry ($\{-1\ -2\ -3\}$) is assumed. Default: "cubic"
<code>strain</code>	Sets the strain tensor that reduces the symmetry relative to the full symmetry of the crystal class.

Examples

See [Creating Band Data for Sentaurus Device Monte Carlo on page 139](#).

Creating Input Files

This section presents the commands for creating input files.

createGarandMCFiles

This command determines the symmetry of the strained `crystal` (assuming cubic symmetry unless specified otherwise), performs band-structure calculation including

Chapter 14: Sentaurus Band Structure/Tcl Commands

Creating Input Files

symmetry effects, and writes band-structure data files for Garand MC to the current working directory.

You can define a prefix or postfix to the band-structure output files by using the `outputPrefix` or `outputPostfix` argument of the `sBandSet` command (see [sBandGet and sBandSet on page 273](#)).

To use these files for a Garand MC simulation, see the *Garand User Guide*.

Syntax

```
createGarandMCFfiles crystal=EPM::Crystal | AnalyticBandSolver  
[crystalClass=String] [Kmax=Double] [numK=Integer]
```

Argument	Description
crystal	Calculates band-structure data using this band-structure calculator object. <code>crystal</code> can be either an <code>EPM::Crystal</code> object or an <code>AnalyticBandSolver</code> object.
crystalClass	Selects the name of the symmetry class of the crystal. Options are: <ul style="list-style-type: none">"cubic": The relaxed crystal has cubic symmetry."generic": Use if the crystal class is not known. Only Kramer's symmetry ($\{-1 -2 -3\}$) is assumed. Default: "cubic"
Kmax	Sets the maximum value to use for the k -grid for the calculation of the dispersion (unit: $2\pi/a_0$). Default: 1.0
numK	Sets the number of uniformly spaced points to use in the k -grid for the calculation of the dispersion. Default: 96

Examples

See [Creating Band Data for Garand MC on page 140](#).

createMonteCarloFiles

This command determines the symmetry of the strained `crystal` (assuming cubic symmetry unless specified otherwise), performs band-structure calculation including symmetry effects, writes band-structure data files for Sentaurus Device Monte Carlo, creates an `ana.dat` file with the correct strained lattice constants (the other values in `ana.dat` are parameters for impurity scattering rates, which are retained at relaxed silicon defaults), and copies additional auxiliary files (`${STROOT_LIB}/sparta/* .dat`) from the TCAD Sentaurus library directory to the specified output directory.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Finding Conduction Band Valley Positions

To use these files for a Sentaurus Device Monte Carlo simulation in Sentaurus Device, set `MonteCarloPath` in the `File` section of your Sentaurus Device command file to the name of the output directory of `createMonteCarloFiles` (see [Arbitrary Stress on page 86](#) for details).

Syntax

```
createMonteCarloFiles crystal=EPM::Crystal | AnalyticBandSolver  
[crystalClass=String] [directory=String]
```

Argument	Description
crystal	Calculates band-structure data using this band-structure calculator object. <code>crystal</code> can be either an <code>EPM::Crystal</code> object or an <code>AnalyticBandSolver</code> object.
crystalClass	Selects the name of the symmetry class of the crystal. Options are: <ul style="list-style-type: none">• "cubic": The relaxed crystal has cubic symmetry.• "generic": Use if the crystal class is not known. Only Kramer's symmetry ($\{-1 -2 -3\}$) is assumed. Default: "cubic"
directory	Sets the output directory path. To be treated correctly by the Monte Carlo simulator, the directory name must contain <code>SPARTA</code> (all uppercase).

Examples

See [Creating Band Data for Sentaurus Device Monte Carlo on page 139](#).

Finding Conduction Band Valley Positions

These commands are used to find the positions of conduction band valleys.

findBandMinimum

This command uses a 3D Newton search to find the minimum of a specified band of a specified crystal. The search starts at `k`-vector `kStart`. The `k`-vector at which either $\|\mathbf{dk}\|$ or $\|\mathbf{v}\|$ is smaller than the corresponding tolerance is returned as a Tcl list with three components (unit: $2\pi/a_{x/y/z}$).

Chapter 14: Sentaurus Band Structure/Tcl Commands

Finding Conduction Band Valley Positions

Syntax

```
findBandMinimum crystal=:EPM::Crystal kStart=RealVector3D  
[band=Integer] [bandstructure=String]  
[damping=Double] [dK_max=Double]  
[groupVelocity=String] [inverseMass=String] [v_max=Double]
```

Argument	Description
band	Sets the index of a band for which to find its minimum. Default: 4
bandstructure, groupVelocity, inverseMass	Set these arguments to specify the names of the container objects used by this command to store the band structure, group velocity, and reciprocal effective mass results computed during the Newton search. Upon successful completion of the <code>findBandMinimum</code> command, these container objects contain valid data at the converged minimum. If necessary, the <code>findBandMinimum</code> command constructs <code>bandstructure_t</code> , <code>groupVelocity_t</code> , and <code>inverseMass_t</code> objects of the requested name.
crystal	Sets the name of the crystal whose band structure will be searched.
damping	Sets the value by which to multiply the update vector <code>dk</code> . Default: 1
dK_max	Convergence is reached if the length of the (undamped) update vector <code>dk</code> is less than <code>dK_max</code> (unit: $2\pi/a_{x/y/z}$). Default: 1e-7
kStart	Sets the starting point of the Newton search at the specified k -vector (unit: $2\pi/a_{x/y/z}$). The k -vector at which either $\ dk\ $ or $\ v\ $ is smaller than the corresponding tolerance is returned as a Tcl list with three components.
v_max	Convergence is reached if the modulus of the gradient of the band structure (or the group velocity $\ v\ $) at the current k -vector is smaller than <code>v_max</code> (unit: $\frac{eV}{h}a_{x/y/z}$). Default: 1e-7

Examples

```
SiliconCrystal name=bulkSi  
findBandMinimum crystal=bulkSi kStart=[list 1 0 0]
```

Output:

```
K = 0.855746 0.000000 -0.000000; |dk|=1.443e-01; |v| = 1.049e-01  
K = 0.849893 -0.000000 0.000000; |dk|=5.853e-03; |v| = 3.936e-03  
K = 0.849880 -0.000000 0.000000; |dk|=1.332e-05; |v| = 8.924e-06  
K = 0.849880 -0.000000 0.000000; |v| = 4.590e-09; done.
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Finding Conduction Band Valley Positions

Result:

```
0.849879596213 -4.3193223934e-13 4.1853096846e-13
```

findBandMinima

This command uses 3D Newton searches to find the minima of a specified band of a specified crystal near each of the **k**-vectors contained in the list `kStart`. All searches are performed in parallel.

The **k**-vectors of the minima found by this command are returned as an ordered list of 3D vectors. The order of the minimum vectors corresponds to that of the starting vectors in `kStart`. Convergence criteria are the same as in [findBandMinimum on page 316](#).

Syntax

```
findBandMinima crystal=EPM::Crystal kStart=List/RealVector3D  
[band=Integer] [bandstructure=String] [damping=Double]  
[[dK_max=Double groupVelocity=String] [inverseMass=String]  
[v_max=Double] [variables=List/String]
```

Argument	Description
band	Sets the index of a band for which to find its minima. Default: 4
bandstructure, groupVelocity, inverseMass	Set these arguments to specify the names of the container objects used by this command to store the band structure, group velocity, and reciprocal effective mass results computed during the Newton search. Upon successful completion of the <code>findBandMinima</code> command, these container objects contain valid data at all minima that were found. If necessary, the <code>findBandMinima</code> command constructs <code>bandstructure_t</code> , <code>groupVelocity_t</code> , and <code>inverseMass_t</code> objects of the requested name.
crystal	Sets the name of the crystal whose band structure will be searched.
damping	Sets the value by which to multiply the update vector <code>dk</code> . Default: 1
dK_max	Convergence is reached if the length of the (undamped) update vector <code>dk</code> is less than <code>dK_max</code> (unit: $2\pi/a_{x/y/z}$). Default: 1e-7
kStart	Sets a list of starting points (unit: $2\pi/a_{x/y/z}$).
v_max	Convergence is reached if the modulus of the gradient of the band structure (or the group velocity $\ \mathbf{v}\ $) at the current k -vector is smaller than <code>v_max</code> (unit: $\frac{eV}{h}a_{x/y/z}$). Default: 1e-7

Chapter 14: Sentaurus Band Structure/Tcl Commands

Container Objects for Band-Structure Results

Argument	Description
variables	The length of this list must match the length of kStart. If variables is supplied, then the <code>findBandMinima</code> command creates or sets a variable in the scope of the calling function for each vector in kStart. The names of these variables are the entries of variables, and their values upon successful completion of the <code>findBandMinima</code> command are the positions of the band minima that were found.

Examples

Perform a parallel search for the conduction-band minima in the x- and z-direction, and store the **k**-vectors of the minima in the Tcl variables `DeltaX` and `DeltaZ`, respectively:

```
SiemensCrystal name=silicon
silicon apply uniaxialStrain dir=[list 1 1 0] stress=3.0e9
findBandMinima crystal=silicon kStart=[list {0.85 0 0} {0 0 0.85} \
variables=[list DeltaX DeltaZ]
puts "DeltaX: \$DeltaX"
puts "DeltaZ: \$DeltaZ"
```

Output:

```
0: K = 0.851077 -0.008973 -0.000000; |dK|=9.037e-03; |v| = 2.777e-02
1: K = -0.000000 -0.000000 0.865961; |dK|=1.596e-02; |v| = 8.967e-03
0: K = 0.851145 -0.008965 0.000000; |dK|=6.910e-05; |v| = 5.026e-05
1: K = -0.000000 0.000000 0.866406; |dK|=4.451e-04; |v| = 2.362e-04
0: K = 0.851145 -0.008965 0.000000; |v| = 2.302e-08; done.
1: K = 0.000000 -0.000000 0.866407; |dK|=7.345e-07; |v| = 3.891e-07
1: K = 0.000000 -0.000000 0.866407; |v| = 2.550e-10; done.
0.851145300153 -0.00896520620977 2.80874278617e-13 4.56550566776e-12
-6.99094408167e-13 0.866407041341
DeltaX: { 0.851145300153 -0.00896520620977 2.80874278617e-13 }
DeltaZ: { 4.56550566776e-12 -6.99094408167e-13 0.866407041341 }
```

Container Objects for Band-Structure Results

This section presents the commands for creating container objects for band-structure results.

bandstructure_t

This command creates a new `bandstructure_t` object. The `name` argument becomes the name of a new Tcl command for accessing the `bandstructure_t` object.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Container Objects for Band-Structure Results

Syntax

```
bandstructure_t name=String
```

Argument	Description
name	Sets the name of the new <code>bandstructure_t</code> object.

Using <bandstructure_t> Objects

`bandstructure_t` objects are accessed by using their names as Tcl command names. This is denoted by `<bandstructure_t>`.

Using the object name as a Tcl command without arguments causes Sentaurus Band Structure to output a message that identifies the associated object. Alternatively, the first argument must specify a *method* that will be executed on the object. The available methods listed here.

<bandstructure_t> destroy

This method destroys the specified object. It has no arguments.

```
<bandstructure_t> destroy
```

<bandstructure_t> get

This method returns a list containing the band energies at **k**-vector `kVector` or (if the optional argument `band` is given) the energy of the specified `band` at **k**-vector `kVector`.

```
<bandstructure_t> get kVector=vector3D [band=Integer]
```

<bandstructure_t> get { conductionMin | valenceMax }

This method returns the conduction-band minimum or valence-band maximum (in eV).

```
<bandstructure_t> get { conductionMin | valenceMax }
```

<bandstructure_t> get kVectors

This method returns a list of all **k**-vectors for which band energies are stored. It has no arguments.

```
<bandstructure_t> get kVectors
```

<bandstructure_t> get topValenceBand

This method queries the index of the highest valence band stored in the `bandstructure_t` object.

```
<bandstructure_t> get topValenceBand
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Container Objects for Band-Structure Results

It returns -1 if no valence bands are stored.

<bandstructure_t> set

This method sets the band energies at `kVector` to the vector `energies`. No shifting is applied (compared with the `shiftConductionBands` or `shiftValenceBands` arguments in [sBandGet and sBandSet on page 273](#)).

```
<bandstructure_t> set kVector=vector3D energies=List/Double
```

<bandstructure_t> set topValenceBand

For band-shifting purposes, this method treats `topValenceBand` as the index of the highest valence band.

```
<bandstructure_t> set topValenceBand=Integer
```

<bandstructure_t> size

This method returns the number of *k*-points in the dataset. It has no arguments.

```
<bandstructure_t> size
```

<bandstructure_t> status

This method outputs status information about the `bandstructure_t` object. It has no arguments.

```
<bandstructure_t> status
```

groupVelocity_t

This command creates a new `groupVelocity_t` object. The `name` argument becomes the name of a new Tcl command for accessing the `groupVelocity_t` object.

Syntax

```
groupVelocity_t name=String
```

Argument	Description
name	Sets the name of the new <code>groupVelocity_t</code> object.

Using <groupVelocity_t> Objects

`groupVelocity_t` objects are accessed by using their names as Tcl command names. This is denoted by `<groupVelocity_t>`.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Container Objects for Band-Structure Results

Using the object name as a Tcl command without arguments causes Sentaurus Band Structure to output a message that identifies the associated object. Alternatively, the first argument must specify a *method* that will be executed on the object. The available methods listed here.

<groupVelocity_t> destroy

This method destroys the specified object. It has no arguments.

```
<groupVelocity_t> destroy
```

<groupVelocity_t> get

This method returns a list of the group velocity vectors at **k**-vector **kVector** or (if the optional argument **band** is given) the group velocity vector of the specified **band** at **k**-vector **kVector**. Velocities are given in units of $\frac{eV}{h}a_{x/y/z}$ where $a_{x/y/z}$ denotes the lattice constant along the x-, y-, and z-axis, resulting from deformation under the diagonal part of the strain tensor.

```
<groupVelocity_t> get kVector=vector3D [band=Integer]
```

<groupVelocity_t> size

This method returns the number of **k**-points in the dataset. It has no arguments.

```
<groupVelocity_t> size
```

<groupVelocity_t> status

This method outputs status information about the **groupVelocity_t** object. It has no arguments.

```
<groupVelocity_t> status
```

inverseMass_t

This command creates a new **inverseMass_t** object. The **name** argument becomes the name of a new Tcl command for accessing the **inverseMass_t** object.

Syntax

```
inverseMass_t name=String
```

Argument	Description
name	Sets the name of the new inverseMass_t object.

Using <inverseMass_t> Objects

<inverseMass_t> objects are accessed by using their names as Tcl command names. This is denoted by <inverseMass_t>.

Using the object name as a Tcl command without arguments causes Sentaurus Band Structure to output a message that identifies the associated object. Alternatively, the first argument must specify a *method* that will be executed on the object. The available methods listed here.

<inverseMass_t> destroy

This method destroys the specified object. It has no arguments.

```
<inverseMass_t> destroy
```

<inverseMass_t> get

This method returns a list of the reciprocal effective-mass tensors at **k**-vector **kVector** or (if the optional argument **band** is given) the reciprocal effective-mass tensor of the specified band at **k**-vector **kVector**.

```
<inverseMass_t> get kVector=vector3D [band=Integer]
```

<inverseMass_t> size

This method returns the number of **k**-points in the dataset. It has no arguments.

```
<inverseMass_t> size
```

<inverseMass_t> status

This method outputs status information about the **inverseMass_t** object. It has no arguments.

```
<inverseMass_t> status
```

3D Vector/Matrix Auxiliary Commands

This section presents auxiliary commands related to 3D vectors and matrices.

crossProduct

This command returns the cross product **u** × **v**.

Tcl syntax (positional arguments):

```
crossProduct u v
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

3D Vector/Matrix Auxiliary Commands

- `vector3D u`
- `vector3D v`

Examples

```
crossProduct {1 0 0} {0 1 0}
```

Result: {0.0 0.0 1.0}

invertMatrix

This command returns the inverse M^{-1} of a 3×3 matrix M.

Tcl syntax (positional arguments):

```
invertMatrix M
```

- `matrix3D M`

Examples

```
invertMatrix {{0.5 -0.5 0.0} {0.5 0.5 0.0} {0.0 0.0 2.0}}
```

Result: {1.0 1.0 0.0} {-1.0 1.0 0.0} {0.0 0.0 0.5}

```
invertMatrix {{1 0 0} {0 1 0} {0 0 0}}
```

Result:

Caught an exception of type std::domain error.

Description: Cannot invert a singular matrix!

```
invertMatrix {{0 (0,1) 0} {(0,-1) 0 0} {0 0 1}}
```

Result: {0 (-0,1) 0} {(0,-1) 0 0} {0 0 1}

This complex matrix is its own inverse.

matrixProduct

This command returns the matrix product AB.

Tcl syntax (positional arguments):

```
matrixProduct A B
```

- `matrix3D A`
- `matrix3D B`

Examples

```
matrixProduct {{0 1 0} {1 0 0} {0 0 2}} {{0 1 0} {1 0 0} {0 0 0.5}}
```

Result: {1.0 0.0 0.0} {0.0 1.0 0.0} {0.0 0.0 1.0}

matrixVectorProduct

This command returns the vector $\mathbf{y} := \mathbf{M}\mathbf{x}$.

Tcl syntax (positional arguments):

```
matrixVectorProduct M x
```

- `matrix3D M`
- `vector3D x`

Examples

```
matrixVectorProduct {{0.5 0.5 0} {{0.5 -0.5 0} {0 0 2}} {1 0 1}}
```

Result: 0.5 0.5 2.0

scalarProduct

This command returns the scalar product $\mathbf{u} \cdot \mathbf{v}$.

Tcl syntax (positional arguments):

```
scalarProduct u v
```

- `vector3D u`
- `vector3D v`

Examples

```
scalarProduct {1 0 0} {0 1 0}
```

Result: 0.0

The vectors are orthogonal.

solveLinearEquation

This command solves the linear equation $Mx = b$ and returns the solution vector x .

Note:

This command uses named argument parsing.

Syntax

```
solveLinearEquation M=matrix3D b=vector3D
```

Argument	Description
b	Specifies a three-dimensional vector b .
M	Specifies a 3×3 matrix M.

Examples

```
set A {{0.5 -0.5 0.0}
        {0.5 0.5 0.5}
        {0.0 0.0 2.0}}
solveLinearEquation M=$A b=[list 1 0 0]
```

Result: 1.0 -1.0 0.0

Note:

The value for the **b** vector is wrapped into a `list` call to work around a limitation of Tcl (the Tcl parser does not permit argument lists of the form `b={1 0 0}`).

Alternatively, a list argument can be set by expanding a variable as shown for argument **M**.

transposeMatrix

This command returns the transposed 3×3 matrix M^T .

Tcl syntax (positional arguments):

```
transposeMatrix M
```

- `matrix3D M`

Examples

```
transposeMatrix {{1 2 3} {0 4 5} {0 0 6}}
```

Result: {{1.0 0.0 0.0} {2.0 4.0 0.0} {3.0 5.0 6.0}}

unitVec

This command returns a unit vector along the direction of **v**.

Tcl syntax (positional arguments):

unitVec v

- **vector3D v**

Examples

unitVec {1 1 1}

Result: 0.57735026919 0.57735026919 0.57735026919

unitVec {0 0 0}

Result:

Caught an exception of type std::domain error.

Description: Cannot construct unit vector from zero vector.

vectorAdd

This command adds vectors and returns the result **u + v**.

Tcl syntax (positional arguments):

vectorAdd u v

- **vector3D u**
- **vector3D v**

Examples

vectorAdd {1 0 0} {1 1 0}

Result: 2.0 1.0 0.0

vectorAdd {1 0 1} {(0,1) 1 1}

Result: (1,1) 1.0 2.0

vectorDivide

This command divides a vector **v** by a scalar *x* and returns the result **v/x**.

Tcl syntax (positional arguments):

```
vectorDivide v x
```

- **vector3D v**
- **Double x**

Examples

```
vectorDivide {2 1 0} 2
```

Result: 1.0 0.5 0.0

The divisor is promoted automatically to a **Double** to avoid unexpected round-off effects as in `expr 1/2` (**result: 0**) versus `expr 1/2.0` (**result: 0.5**).

vectorLength

This command returns the length of vector **v**.

Tcl syntax (positional arguments):

```
vectorLength v
```

- **vector3D v**

Examples

```
vectorLength {1 1 0}
```

Result: 1.41421356237

```
vectorLength {(1,1) 0 0}
```

Result: 1.41421356237

vectorMultiply

This command multiplies a vector by a scalar and returns the result xv .

Note:

This command uses named argument parsing.

Syntax

```
vectorMultiply v=vector3D x=Double
```

Argument	Description
v	Specifies a 3D vector.
x	Specifies a scalar.

Examples

```
vectorMultiply v=[list 1 2 3] x=1.5
```

Result: 1.5 3.0 4.5

vectorSubtract

This command takes two vector arguments and returns the vector difference $u - v$.

Tcl syntax (positional arguments):

```
vectorSubtract u v
• vector3D u
• vector3D v
```

Examples

```
vectorSubtract {3.0 2.0 1.0} {1.0 2.0 3.0}
```

Result: 2.0 0.0 -2.0

Tcl Support

This section presents commands that provide Tcl support for various operations.

Support for Complex Numbers

These commands provide Tcl support for complex numbers.

imaginaryPart

This command returns the imaginary part of a complex number.

Tcl syntax (positional arguments):

```
imaginaryPart z
```

- Complex *z*

Examples

```
imaginaryPart (1.5,-2.0)
```

Result: -2.0

realPart

This command returns the real part of a complex number.

Tcl syntax (positional arguments):

```
realPart z
```

- Complex *z*

Examples

```
realPart (1.5,-2.0)
```

Result: 1.5

Support for Named Argument Parsing

These commands provide Tcl support for named argument parsing:

- [check_args](#)
- [get_arg](#)
- [get_known_args](#)
- [get_unknown_args](#)
- [has_arg](#)
- [init_arg](#)
- [parse_args](#)

check_args

This command returns an error if the named argument array `$arrayName` contains unknown arguments in the sense of `get_unknown_args` (see [get_unknown_args](#) on page 334).

Syntax

```
check_args arrayName
```

Argument	Description
<code>arrayName</code>	Positional argument of type <code>Array</code> , which is usually a named argument array from <code>parse_args</code> .

Examples

```
proc myProcXY args { parse args para $args
    set X [get arg para "X"]
    set Y "<undefined>"
    if { [has arg para "Y"] } {
        set Y [get arg para "Y"]
    }
    check args para
    puts "argument list is valid: X=$X; Y=$Y."
}
myProcXY X=1
myProcXY X=2 Y=yString
myProcXY X=3 Y=yStringA Z=something
```

This test defines a Tcl procedure `myProcXY` with a mandatory argument `x` and an optional argument `y`. This procedure is then called with three different argument lists.

Output:

```
argument list is valid: X=1; Y=<undefined>.  
argument list is valid: X=1; Y=yString.  
Error: unknown argument { Z } detected. The following arguments are  
known: X Y.
```

get_arg

This command obtains a named argument *name* from a named argument array *arrayName* (typically created by a call to `parse_args` (see [parse_args on page 337](#))). If array *arrayName* does not have an entry at key *name*, then an error condition occurs.

Syntax

```
get_arg arrayName name [type]
```

Argument	Description
<i>arrayName</i>	Positional argument of type <code>Array</code> , which is usually a named argument array from <code>parse_args</code> .
<i>name</i>	Positional argument. Name of the argument.
<i>type</i>	Positional argument. Optional type signature string (compare with get_arg on page 332). This optional argument can be supplied to impose a type restriction on the argument value <code>\$arrayName(\$name)</code> . If the result of argument lookup cannot be converted to type <i>type</i> , then a type mismatch error condition will occur. As a side effect, <i>name</i> is added to the list of <i>known</i> arguments.

Examples

```
parse args para { x=1.5 }
puts "test1: [get arg para "x"]"
```

Result: test1: 1.5

```
puts "test2: [get arg para "x" Double]"
```

Result: test2: 1.5: Successful type check.

```
puts "test3: [get arg para "x" Integer]"
```

Result:

```
Type mismatch error in argument "x":  
"1.5" has type Double --- expected type: Integer.
```

```
puts "test4: [get arg para "y" ]"
```

Result:

```
Argument "y" not found. The following named arguments were supplied:  
"x".
```

get_known_args

This command extracts a list of *known* argument names from the named argument array `$arrayName` in the scope of the calling procedure. An argument name becomes *known* by being passed as the `name` argument to `has_arg` or `get_arg` (see [has_arg on page 334](#) and [get_arg on page 332](#)).

This command is intended for use on arrays created by `parse_args` (see [parse_args on page 337](#)).

Syntax

```
get_known_args arrayName
```

Argument	Description
<code>arrayName</code>	Positional argument of type <code>Array</code> , which is usually a named argument array from <code>parse_args</code> .

Examples

```
parse_args para { x=3.0 y=test }  
get_known_args para
```

Result: <none>: Initially, all arguments are unknown.

```
get_arg para "x"  
has_arg para "z"  
get_known_args para
```

Result: The `get_arg` command returns `3.0` (the value of the `x` argument). The call to `has_arg` returns `0` (no argument `z` is contained in the array `para`).

The call to `get_known_args` returns `x z`, which are the names of the arguments mentioned in the two preceding calls.

get_unknown_args

This command extracts a list of *unknown* argument names from the named argument array `$arrayName` in the scope of the calling procedure. An argument name becomes *known* by being passed as the `name` argument to `has_arg` or `get_arg` (see [has_arg on page 334](#) and [get_arg on page 332](#)). All other argument names are unknown.

This command is intended for use on arrays created by `parse_args` (see [parse_args on page 337](#)).

Syntax

```
get_unknown_args arrayName
```

Argument	Description
<code>arrayName</code>	Positional argument of type <code>Array</code> , which is usually a named argument array from <code>parse_args</code> .

Examples

```
parse_args para { x=3.0 y=test }
get_unknown_args para
```

Result: `x y`: Initially, all arguments in `para` are “unknown”.

```
get_arg para "x"
has_arg para "z"
get_known_args para
```

Result: The `get_arg` command returns `3.0` (the value of the `x` argument). The call to `has_arg` returns `0` (no argument `z` is contained in the array `para`).

The call to `get_unknown_args` returns `y`, which is the name of the argument that was not mentioned in the two preceding calls.

has_arg

This command queries whether the array `$arrayName` in the scope of the calling procedure contains an entry for `$name` (return value: 1) or not (return value: 0). As a side effect, `name` is added to the list of *known* arguments.

This command is intended for use on arrays created by `parse_args` (see [parse_args on page 337](#)).

Syntax

```
has_arg arrayName name
```

Argument	Description
arrayName	Positional argument of type <code>Array</code> , which is usually a named argument array from <code>parse_args</code> .
name	Positional argument. Check for the presence of the argument <code>name</code> .

Examples

```
parse args para { x=0 }
puts "looking for x: [has arg para x]"
puts "looking for y: [has arg para y]"
```

Result:

```
looking for x: 1 Found in named argument array para.
looking for y: 0 Not found in named argument array para.
```

init_arg

This command queries whether the array `$arrayName` in the scope of the calling procedure contains a value at key `$name`. If successful, this value is returned; otherwise, the procedure returns `$defaultValue`.

This command is intended for use on arrays created by `parse_args` (see [parse_args on page 337](#)).

By default, `init_arg` accepts only argument values that can be promoted to the type of `defaultValue`. The optional argument `type` can be supplied to specify a target type different from the type of the default value. This can be useful, for example, if the default value of an argument is real, but complex values are also acceptable. As a side effect, `name` is added to the list of `known` arguments.

Syntax

```
init_arg arrayName name defaultValue [type]
```

Argument	Description
arrayName	Positional argument of type <code>Array</code> , which is usually a named argument array from <code>parse_args</code> .
defaultValue	Positional argument. Default value if argument <code>name</code> does not exist.
name	Positional argument. Return value of parameter <code>name</code> .
type	Positional argument. Optional target type signature. If <code>type</code> is not given, then the value of argument <code>name</code> must be converted to the type of <code>defaultValue</code> . Otherwise, it must be converted to <code>type</code> .

Examples

```
parse_args para1 { x=5.0 }
set x [init_arg para1 "x" 1.0]
set y [init_arg para1 "y" 2.0]
puts "x: $x; y: $y"
```

Result: x: 5.0; y: 2.0 – x takes the value `$para1{"x"}=5.0`; y is not listed in para and defaults to 2.0.

```
parse_args para2 { x=(0,1) }
puts "x: [init_arg para2 "x" 1.0]"
```

Result:

```
Type mismatch error in argument "x":
"(0,1)" has type Complex --- expected type: Double.
```

Here, `init_arg` fails because the type of the actual argument value (`Complex`) differs from the type of the default argument (`Double`).

In situations where this is required (it is acceptable to have a complex number, but the default value is real), you can provide an explicit target type:

```
puts "x: [init_arg para2 "x" 1.0 Complex]"
```

Result: "x: (0,1)"

parse_args

This command extracts named-argument declarations from the `List $argv` and stores the results of this search in an array of name `$arrayName` in the scope of the calling procedure. The command `parse_args` supports two syntactic forms for declaring an argument `name`:

- `name=value`
- `-name value`

The `value` of each extracted argument is stored as an array entry `$arrayName($name)`.

The difference between the `name=value` and the `-name value` notations is that the former is slightly more readable and the latter is more efficient, because the argument name and argument value do not need to be combined into a single string object. This allows the value object to be passed by reference and preserves its internal type representation. In addition, the parsing rules of Tcl do not allow list literals to be passed using the `name=value` syntax.

Syntax

```
parse_args arrayName argv
```

Argument	Description
argv	A List of command-line arguments.
arrayName	Name of output Array in scope of the calling function.

Examples

```
parse_args A {x=5 y="string with spaces" z=word -myList {a b c}}
puts "x: $A(x); y: $A(y); z: $A(z); myList: $A(myList)."
```

The command `parse_args` extracts four named arguments `x`, `y`, `z`, and `myList` from `argv` and stores them in Array `A`.

Result: `x: 5; y: string with spaces; z: word; myList: a b c.`

```
proc myProc args {
    parse_args para $args
    puts "value of argument x: [get_arg para "x"]
}
myProc x="This is a string."
```

This example shows how `parse_args` is used to parse the argument list of a Tcl procedure. The first command defines a procedure `myProc`; `myProc` calls `parse_args` to parse its argument list `args` and to store the result in Array `para`.

`get_arg` is used to report the value of the `x` argument passed to `myProc`. The second command is an example call to `myProc`.

Result: value of argument `x`: This is a string.

Support for Type Checking

These commands provide Tcl support for type checking:

- [check_type](#)
- [common_type](#)
- [get_type](#)

check_type

This command checks whether the specified object can be converted to the specified type. If so, then the command returns; otherwise, an exception is thrown.

Note:

This command uses named argument parsing for its argument list.

Syntax

`check_type obj=String type=String`

Argument	Description
<code>obj</code>	Specifies a Tcl object or the Tcl command associated with a C++ object.
<code>type</code>	Specifies a type signature string (see get_type on page 339).

Examples

`check_type obj=1.5 type=Double`

Result: success (no message)

`check_type obj=1.5 type=Integer`

Result: "1.5" has type Double --- expected type: Integer.

```
check_type obj=1.0 type=Integer
```

Result: "1.0" has type Double --- expected type: Integer.

common_type

This command takes two type signature strings and returns the most specialized type to which objects of both types can be propagated.

Syntax

```
common_type type1 type2
```

Argument	Description
type1, type2	Positional arguments: type signature strings.

Examples

```
common_type Double Complex
```

Returns Complex because a real number can be promoted to a complex number but not vice versa.

```
common_type List#3/List#2/Double List#3/List/Integer
```

Returns List#3/List/Double because an Integer can be propagated to a Double, and a List of specified length can be propagated to a list of arbitrary length.

get_type

This command returns the type of the specified object. This command recognizes the following types:

- Numeric types (implicit type propagation allowed from top to bottom)
- Instantiable classes
- Tcl-related types

For details, see [Table 55 on page 268](#). The type of a List is analyzed recursively. For example, get_type {{1 2} {1 3} {1 4}} returns a type signature of List#3/List#2/Integer for a List with three elements, each of which is a List with two elements of type Integer.

Syntax

```
get_type obj
```

Argument	Description
<i>obj</i>	Positional argument. Find the type of this object.

Examples

```
get_type {{1 2} {1 2 3} {1 2}}
```

Returns List#3/List/Integer, that is, a list of three lists (arbitrary length) of integers.

```
get_type {1 1.5 "a"}
```

Returns List#3 (no common element type other than String).

General Utilities

These commands provide general utilities.

import_array

This command brings the array *\$arrayName* from the scope two levels up to the scope of the calling function under the name *localName*. If *localName* is not given, then it defaults to *theArray*.

Syntax

```
import_array arrayName [localName]
```

Argument	Description
<i>arrayName</i>	Positional argument. An Array in the scope of the caller of import_array.
<i>localName</i>	Positional argument. New name for the same array in the scope of the caller of import_array.

Examples

```
proc myProc {} {
    puts "A before import: [array exists A]"
    puts "X before import: [array exists X]"
```

```
import array X A
puts "A after import: [array exists A]"
puts "A(1) = $A(1)"
}
set X(1) 5
myProc
```

The call to `import_array` makes Array `X` visible inside `myProc`; the name for accessing array `X` from within `myProc` is `A`.

Output of myProc:

```
A before import: 0
X before import: 0
A after import: 1
A(1) = 5
```

tee

This command outputs a message to both the screen and the specified file.

Syntax

```
tee file message
```

Argument	Description
<code>file</code>	Positional argument. A Tcl file descriptor.
<code>message</code>	Positional argument. Output this message.

Examples

```
set f [open "tmp.file" "w"]
tee $f "This is a test."
close $f
exec cat tmp.file
```

This test opens a file `tmp.file` for writing ("`w`"), writes the message "This is a test." both to the screen and the file, closes the file, and then displays the contents of `tmp.file` using the UNIX command `cat`.

Output:

```
This is a test.
This is a test.
```

Material Database

This section describes the commands available in the material DB from within the Sentaurus Band Structure Tcl interpreter (see [Material Database on page 256](#)):

- [CreateMaterial and ModifyMaterial](#)
 - [CloneMaterial](#)
 - [ListMaterials](#)
 - [RemoveMaterial](#)
 - [AddValley and ModifyValley](#)
 - [RemoveValley](#)
 - [AddScatteringModel and ModifyScatteringModel](#)
 - [RemoveScatteringModel](#)
 - [ApplyMaterial](#)
 - [PrintParameters](#)
 - [ListMaterialMap](#)
-

CreateMaterial and ModifyMaterial

The `CreateMaterial` command creates material definitions. The `ModifyMaterial` command modifies existing material definitions.

Syntax

```
CreateMaterial dbMaterial=String type=String
    Affinity=Double Bandgap=Double Permittivity=Double
    [a0=Double]
    [eBulkDensity=Double | String] [hBulkDensity=Double | String]
    [eDensity=Double | String] [hDensity=Double | String]
    [Nc=Double] [Nv=Double]
    [s11=Double] [s12=Double] [s44=Double]
    [strainTensor=RealMatrix3D | stressTensor=RealMatrix3D]
    [table=String] [useNamePrefix=Boolean | String]

ModifyMaterial dbMaterial=String
    [a0=Double] [Affinity=Double] [Bandgap=Double]
    [eBulkDensity=Double | String] [hBulkDensity=Double | String]
    [eDensity=Double | String] [hDensity=Double | String]
    [Nc=Double] [Nv=Double] [Permittivity=Double]
    [s11=Double] [s12=Double] [s44=Double]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Material Database

```
[strainTensor=RealMatrix3D | stressTensor=RealMatrix3D]

```

Argument	Description
a0	Sets the relaxed lattice spacing in Å. This argument is valid only for semiconductor material types.
Affinity	Sets the relaxed electron affinity in eV. Default: 4.0727
Bandgap	Sets the relaxed band gap in eV. Default: 1.242
dbMaterial	Sets the name of the material.
eBulkDensity	Sets the electrostatic model definition for eBulkDensity (see Carrier Density on page 190). If you specify a Double value, then this model is set to this constant value. This argument is valid only for semiconductor and metal material types. Default: [eMultiValleyDensity]
eDensity	Sets the electrostatic model definition for eDensity (see Carrier Density on page 190). If you specify a Double value, then this model is set to this constant value. This argument is valid only for semiconductor and metal material types. Default: [eHybridDensity]
hBulkDensity	Sets the electrostatic model definition for hBulkDensity (see Carrier Density on page 190). If you specify a Double value, then this model is set to this constant value. This argument is valid only for semiconductor and metal material types. Default: [hMultiValleyDensity]
hDensity	Sets the electrostatic model definition for hDensity (see Carrier Density on page 190). If you specify a Double value, then this model is set to this constant value. This argument is valid only for semiconductor and metal material types. Default: [hHybridDensity]
Nc , Nv	Set the effective conduction band and valence band DOS at 300 K in cm ⁻³ . These arguments are valid only for semiconductor and metal material types.
Permittivity	Sets the dielectric permittivity relative to the vacuum permittivity. Default: 11.7
s11, s12, s44	Set the components of the compliance tensor used to define the elastic properties for strain calculations (unit: %/GPa = 1/Mbar). These arguments are valid only for semiconductor material types.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Material Database

Argument	Description
strainTensor	<p>Specifies the strain tensor, which is represented as a list of 3D row vectors in the crystal axis frame (unit: Pa):</p> <ul style="list-style-type: none">Positive values: Tensile stressNegative values: Compressive stress <p>This argument is valid only for semiconductor material types.</p>
stressTensor	<p>Specifies the stress tensor, which is represented as a list of 3D row vectors in the device axis frame (unit: Pa):</p> <ul style="list-style-type: none">Positive values: Tensile stressNegative values: Compressive stress <p>This argument is valid only for semiconductor material types.</p>
table	Sets the file name for importing parameter values from an ASCII text file.
type	Sets the type of the material to be created or modified. Options are: <ul style="list-style-type: none">insulatormetalsemiconductor
useNamePrefix	<p>Specifies whether to activate a prefix for parameters when importing the table. Options are:</p> <ul style="list-style-type: none">0: Do not use prefix.1: Use material name as prefix. <p>Alternatively, if you specify a <code>String</code> value, then the specified value is used as a prefix.</p>

Examples

Create a new semiconductor called `Silicon` in the material DB:

```
CreateMaterial dbMaterial=Silicon type=semiconductor \
  Permittivity=11.7 Affinity=4.0727 Bandgap=1.1242 \
  Nc=2.8213257e19 Nv=3.1046e19 \
  a0=5.43 \
  s11=0.762451 s12=-0.213158 s44=1.25
```

Modify the existing DB `Silicon` material to specify a tensile strain component of 1 GPa in the x-device direction:

```
ModifyMaterial dbMaterial=Silicon \
  stressTensor=[list {1.e9 0 0} {0 0 0} {0 0 0}]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Material Database

Create a new insulator material called `Oxide` in the material DB:

```
CreateMaterial dbMaterial=Oxide type=insulator Permittivity=3.9 \
Affinity=0.9 Bandgap=9.0
```

Modify the existing DB insulator `Oxide` to have a relative permittivity of 2.5:

```
ModifyMaterial dbMaterial=Oxide Permittivity=2.5
```

Create a new metal material called `Metal` in the material DB:

```
CreateMaterial dbMaterial=Metal type=metal Permittivity=11.7 \
Affinity=4.0727 Bandgap=0.0
```

Modify the existing DB metal `Metal` to have an affinity of 4.25:

```
ModifyMaterial dbMaterial=Metal Affinity=4.25
```

CloneMaterial

This command clones an existing material to a new name.

Syntax

```
CloneMaterial srcName=String dstName=String
```

Argument	Description
<code>dstName</code>	Sets the name of the destination cloned material.
<code>srcName</code>	Sets the name of the source material to clone.

Examples

Add a new material with the name `chnSilicon` that is a direct clone of the `Silicon` material:

```
CloneMaterial srcName=Silicon dstName=chnSilicon
```

You can modify the models and parameters for the `chnSilicon` material without affecting the `Silicon` material.

ListMaterials

This command lists all materials, and associated types, available in the material DB to the screen. It has no arguments.

Syntax

```
ListMaterials
```

Examples

```
ListMaterials
```

RemoveMaterial

This command removes a material definition from the material DB.

Syntax

```
RemoveMaterial dbMaterial=String
```

Argument	Description
dbMaterial	Sets the name of the material to be removed from the material DB.

Examples

Remove the Germanium semiconductor from the material DB:

```
RemoveMaterial dbMaterial=Germanium
```

AddValley and ModifyValley

The `AddValley` command adds a valley model definition to a specific material.

The `ModifyValley` command modifies a valley model definition for a specific material.

Syntax

```
AddValley dbMaterial=String ValleyModel=String name=String  
degeneracy=Integer  
[useForEBulkDensity=Boolean | useForHBulkDensity=Boolean]  
[table=String] [useNamePrefix=Boolean | String]]  
[virtualBandEdgeShift=Double]  
<paraName1>=<value1> ...
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Material Database

```
ModifyValley dbMaterial=String ValleyModel=String
  [degeneracy=Integer] [name=String]
  [useForEBulkDensity=Boolean | useForHBulkDensity=Boolean]
  [table=String] [useNamePrefix=Boolean | String]
  [virtualBandEdgeShift=Double]
  <paraName1>=<value1> ...
```

Argument	Description
dbMaterial	Sets the name of the material to which a valley model definition is added.
degeneracy	Sets the valley degeneracy.
name	Sets the name of the valley model definition to be added or modified.
< <i>paraName1</i> >	This is the name of any valid parameter for the selected valley model and is used to set the value of that parameter.
table	Sets the file name for importing parameter values from an ASCII text file.
useForEBulkDensity	Specifies whether to create a bulk electron density model automatically. Options are: <ul style="list-style-type: none">• 0: Do not create bulk model (default).• 1: Create bulk model.
useForHBulkDensity	Specifies whether to create a bulk hole density model automatically. Options are: <ul style="list-style-type: none">• 0: Do not create bulk model (default).• 1: Create bulk model.
useNamePrefix	Specifies whether to activate a prefix for parameters when importing the table. Options are: <ul style="list-style-type: none">• 0: Do not use prefix.• 1: Use material name as prefix. Alternatively, if you specify a <i>String</i> value, then the specified value is used as a prefix.
ValleyModel	Sets the type of the valley model. All types of valley model are available and include: <ul style="list-style-type: none">• ConstantEllipsoid, 2kpEllipsoid• 2kpValley, 3kpValley, 6kpValley, 8kpValley

Chapter 14: Sentaurus Band Structure/Tcl Commands

Material Database

Argument	Description
virtualBandEdgeShift	Sets the virtual conduction band or valence band edge shift to pull the band edge below the Fermi level in the physical metal contact. This argument applies only to metal material types.

Examples

Create a valley model with the name `Delta1` and based on the `2kpEllipsoid` model:

```
AddValley dbMaterial=Silicon ValleyModel=2kpEllipsoid name=Delta1 \
degeneracy=2 longAxis=100 useForEBulkDensity=1 ml=0.92
```

In this command, the valley degeneracy is specified as 2, and the longitudinal axis (`longAxis`) and `ml` parameters of the model are set to 100 and 0.92, respectively. Specifying `useForEBulkDensity=1` automatically creates a model to compute the bulk electron density for this valley.

RemoveValley

This command removes a valley model definition from a specific material in the material DB.

Syntax

```
RemoveValley dbMaterial=String name=String
```

Argument	Description
dbMaterial	Sets the name of the DB material from which to remove a valley model definition.
name	Sets the name of the valley model definition to remove. To remove all valley model definitions, specify the alias <code>all</code> .

Examples

Remove the valley `Gamma` from the DB material `Silicon`:

```
RemoveValley dbMaterial=Silicon name=Gamma
```

AddScatteringModel and ModifyScatteringModel

The `AddScatteringModel` creates a scattering model definition within a specific material.

The `ModifyScatteringModel` modifies a scattering model definition within a specific material.

Syntax

```
AddScatteringModel dbMaterial=String ScatteringModel=String
    name=String transitionType=String valleys=List
    [alloyComponent=String]
    [kdepFF=Integer] [mrFactor=Integer] [table=String]]
    [useNamePrefix=Boolean | String]
    <paraName1>=<value1> ...
```



```
ModifyScatteringModel dbMaterial=String ScatteringModel=String
    name=String
    [alloyComponent=String]
    [kdepFF=Integer] [mrFactor=Integer] [table=String]]
    [transitionType=String] [useNamePrefix=Boolean | String]
    [valleys=List]
    <paraName1>=<value1> ...
```

Argument	Description
alloyComponent	Specifies the component of a binary alloy with which a scattering model is associated. Using the alloy definition $A_{1-x}B_x$, material A is the major component and material B is the minor component (see Interpolation of Material-Specific Scattering Models on page 263). Options are: <ul style="list-style-type: none">• none – not associated with a component (default)• major• minor• both
dbMaterial	Sets the name of the material in which to add a scattering model definition.
kdepFF	Sets the wavefunction overlap form-factor.
mrFactor	Sets the momentum relaxation factor.
name	Sets the name of the scattering model definition to add or to modify.
<paraName1>	This is the name of any valid parameter for the selected scattering model and is used to set the value of that parameter.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Material Database

Argument	Description
ScatteringModel	Sets the type of the scattering model. All types of scattering model are available, including: <ul style="list-style-type: none">• ElasticAcousticPhonon, OhashiAcousticPhonon, InelasticPhonon, TensorElasticAcousticPhonon, TensorInelasticPhonon, POP, RemoteSOPhonon• ElasticAlloy• IsotropicPrangeNee, SRFor1D, SRFor2D, SRFfrom6kp, SRFfromParabolic• Coulomb
table	Sets the file name for importing parameter values from an ASCII text file.
transitionType	Specifies the type of valley transitions allowed. Options are: <ul style="list-style-type: none">• gIntervalley• Intervalley• Intrasubband• Intravalley
useNamePrefix	Specifies whether to activate a prefix for parameters when importing the table. Options are: <ul style="list-style-type: none">• 0: Do not use prefix.• 1: Use material name as prefix. Alternatively, if you specify a <code>String</code> value, then the specified value is used as a prefix.
valleys	Specifies a list of valley pairs denoting the allowed transitions.

Examples

Add an `SRFor2D` scattering model definition to the `Silicon` material:

```
AddScatteringModel dbMaterial=Silicon ScatteringModel=SRFor2D \
    name=ElectronSR transitionType=Intravalley \
    valleys=[list Delta1 Delta2 Delta3] \
    delta=[list {100 0.3e-7} {110 0.3e-7}] \
    lambda=1.5e-7 powerSpectrum=Exponential beta=1.5
```

Here, the model is defined as `Intravalley` and, therefore, the valley list is simply the individual valleys to which the model is applied.

For details about the parameters that apply to each scattering model, see [Scattering Models on page 231](#).

RemoveScatteringModel

This command removes a scattering model definition from a specific material in the material DB.

Syntax

```
RemoveScatteringModel dbMaterial=String name=String
```

Argument	Description
dbMaterial	Sets the name of the DB material from which to remove a scattering model definition.
name	Sets the name of the scattering model definition to remove. To remove all scattering model definitions, specify the alias <code>all</code> .

Examples

Remove the scattering model `ElectronSR` from the DB material `Silicon`:

```
RemoveScatteringModel dbMaterial=Silicon name=ElectronSR
```

ApplyMaterial

This command applies the specified material to the currently loaded device (see [Applying Materials to a Loaded Device on page 258](#)).

Syntax

```
ApplyMaterial dbMaterial=String
  {deviceMaterial=String | deviceRegion=String}
  [fraction=Double] [stressTensor=RealMatrix3D] [verbose=Boolean]
```

Argument	Description
dbMaterial	Sets the name of the material in the material DB to apply to the loaded device.
deviceMaterial	Sets the name of the target material in the loaded device. Convenient aliases are provided: <ul style="list-style-type: none">• <code>all</code> for all regions• <code>insulator</code> for all insulator regions• <code>semiconductor</code> for all semiconductor regions
deviceRegion	Sets the name of the target region in the loaded device.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Material Database

Argument	Description
fraction	Specifies the mole fraction to apply. This argument applies only to alloy materials.
stressTensor	Specifies the stress tensor, which is represented as a list of 3D row vectors (unit: Pa): <ul style="list-style-type: none">• Positive values: Tensile stress• Negative values: Compressive stress This argument is valid only for semiconductor material types.
verbose	Specifies whether to output to screen the explicit Physics commands applied. Options are: <ul style="list-style-type: none">• 0: Do not output the commands (default).• 1: Output the commands.

Examples

Apply the DB material Silicon to the region channel1 in the loaded device, and print the complete set of Physics commands used:

```
ApplyMaterial dbMaterial=Silicon deviceRegion=channel1 verbose=1
```

Apply the alloy DB material SiliconGermanium with a mole fraction x=0.5 to the material tag SiliconGermanium in the loaded device:

```
ApplyMaterial dbMaterial=SiliconGermanium \
    deviceMaterial=SiliconGermanium fraction=0.5
```

PrintParameters

This command prints to screen the requested parameters of a material.

Syntax

```
PrintParameters
  [dbMaterial=String | deviceMaterial=String | deviceRegion=String]
  [filename=String] [fraction=Double]
  [scatteringModel=String | valley=String]
  [strainTensor=RealMatrix3D | stressTensor=RealMatrix3D]
```

Argument	Description
dbMaterial	Sets the name of the database material whose parameters should be output.
deviceMaterial	Sets the name of the device material whose parameters should be output.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Material Database

Argument	Description
deviceRegion	Sets the name of the device region whose parameters should be output.
filename	Specifies the name of the file where the chosen output is written. The file is written to the current working directory. If the file exists, then it is overwritten. Otherwise, a new file is created.
fraction	Sets a specific mole-fraction value of a material to output to screen. Note: This argument affects only the values output.
scatteringModel	Sets the name of the scattering model whose parameters should be output.
valley	Sets the name of the valley model whose parameters should be output.
strainTensor	Specifies the strain tensor, which is represented as a list of 3D row vectors in the crystal axis frame (unit: Pa): <ul style="list-style-type: none">• Positive values: Tensile stress• Negative values: Compressive stress This argument is valid only for semiconductor material types. Note: This argument affects only the values output. It can be used only after a device has been loaded.
stressTensor	Specifies the stress tensor, which is represented as a list of 3D row vectors in the device axis frame (unit: Pa): <ul style="list-style-type: none">• Positive values: Tensile stress• Negative values: Compressive stress This argument is valid only for semiconductor material types. Note: This argument affects only the values output.

Examples

Print the material parameters for Silicon to screen:

```
PrintParameters dbMaterial=Silicon
```

Print the parameters for the Silicon valley Delta1 to screen:

```
PrintParameters dbMaterial=Silicon valley=Delta1
```

Print the parameters for SiliconGermanium at a mole fraction x=0.5 to screen:

```
PrintParameters dbMaterial=SiliconGermanium fraction=0.5
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Print the parameters for the device region `channel1` to screen:

```
PrintParameters deviceRegion=channel1
```

ListMaterialMap

This command lists the mappings between the database material definitions and the loaded device materials and regions to screen.

Syntax

```
ListMaterialMap
```

Examples

```
ListMaterialMap
```

Subband and Mobility Calculations

This section describes the commands used for subband and mobility calculations:

- [AddToFile](#)
- [ComputeMass](#)
- [ComputeMobility](#)
- [ComputeVinj](#)
- [Dopant](#)
- [Extract](#)
- [GetLast](#)
- [LoadDevice](#)
- [Material](#)
- [Math](#)
- [Physics](#)
- [Save](#)
- [SaveDitProfile](#)
- [SaveK](#)
- [Solve](#)

AddToLogFile

This command adds a scalar value to the bias log file at the current bias.

Syntax

```
AddLogFile name=String value=Double
```

Argument	Description
name	Sets the name of the new field to appear in the bias log file.
value	Sets the scalar value to add to the bias log file.

Examples

Add the field called `Ec1` to the current bias log file with the value given by the Tcl variable `Ec`:

```
AddLogFile name=Ec1 value=$Ec
```

ComputeMass

This command computes a particular component of the inverse transport mass tensor, expressed as an effective mass in units of the free electron mass based on the last Schrödinger solution.

For 2D devices, the transport direction is known and need not be specified. The calculation can be limited to a particular set of subbands using the `subbands` argument. If all subbands are included in the calculation, then the total mass and the masses per valley are computed. All computed results are stored in the bias log file. The total mass is returned from the calculation. The calculation typically requires a finer k -space grid than the dispersion. This finer grid can be specified using the `Nk` and `Nphi` arguments.

Syntax

```
ComputeMass {xx | xy | yy} direction=RealVector3D  
[Nk=Integer] [nonlocal=String] [Nphi=Integer] [subbands=List]
```

Argument	Description
direction	Sets the direction, relative to the in-plane device axes, for which the directed mass tensor component should be computed. For 1D devices only.
Nk	Sets the number of radial k -space grid points to use for the mass calculation. Default: 128

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
nonlocal	Sets the name of the nonlocal line for which to compute the mass. If not specified, then the first found nonlocal line is used.
Nphi	Sets the number of angular k -space grid points to use for the mass calculation. For 1D devices only. Default: 128
subbands	Specifies a Tcl list of subbands to be included in the calculation. Default: Include all subbands
xx	Specifies that the xx tensor component of the mass is computed. Used by default. For 1D devices only.
xy	Specifies that the xy tensor component of the mass is computed. For 1D devices only.
yy	Specifies that the yy tensor component of the mass is computed. For 1D devices only.

Examples

Compute the yy transport mass tensor component in a 1D device for only the Delta3_0 subband:

```
set mass [ComputeMass yy subbands=[list Delta3_0]]
```

The computed results are added automatically to the bias log file, and the total mass is returned to the Tcl variable `mass`.

ComputeMobility

This command computes one component of the mobility tensor for a particular nonlocal line or nonlocal area based on the last Schrödinger solution. One tensor component in the device coordinate system can be specified, or the mobility component along one particular direction can be computed. The mobility is computed in units of cm²/Vs.

Syntax

```
ComputeMobility {xx | yy | xy | direction=RealVector3D}  
[nonlocal=String] [writeMobilityPerValley]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
direction	Sets the direction, relative to the in-plane device axes, for which the directed mobility tensor component should be computed. For 1D devices only.
nonlocal	Sets the name of the nonlocal line for which to compute the mobility. If not specified, then the first found nonlocal line is used.
xx	Specifies that the xx component of the mobility tensor should be computed. For 1D devices only.
xy	Specifies that the xy component of the mobility tensor should be computed. For 1D devices only.
yy	Specifies that the yy component of the mobility tensor should be computed. For 1D devices only.
writeMobilityPerValley	If this option is specified, then the mobility in each valley is written to the bias log file.

Examples

Compute the mobility along the [110] direction for a 1D device structure:

```
set mobility [ComputeMobility direction=[list 1 1 0]]
```

The result is added automatically to the bias log file and is returned to the Tcl variable mobility.

ComputeVinj

This command computes the thermal injection velocity in units of cm/s along the user-specified direction based on the last Schrödinger solution.

For 2D devices, the transport direction is known and need not be specified. The calculation can be limited to a particular set of subbands using the subbands argument. If all subbands are included in the calculation, then the total velocity and the velocities per valley are computed. All computed results are stored in the bias log file. The total velocity is returned from the calculation. The calculation typically requires a finer k -space grid than the dispersion. This finer grid can be specified using the Nk and Nphi arguments.

Syntax

```
ComputeVinj {x | y | direction=RealVector3D}  
[Nk=Integer] [nonlocal=String] [Nphi=Integer] [subbands=List]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
direction	Sets the direction, relative to the in-plane device axes, for which the velocity is computed. For 1D only.
Nk	Sets the number of radial k -space grid points to use for the velocity calculation. Default: 128
nonlocal	Sets the name of the nonlocal line for which to compute the velocity. If not specified, then the first found nonlocal line is used.
Nphi	Sets the number of angular k -space grid points to use for the velocity calculation. For 1D only. Default: 128
subbands	Specifies a Tcl list of subbands to be included in the calculation. Default: Include all subbands
x	Specifies that the x-component of the velocity is computed. Used by default. For 1D only.
y	Specifies that the y-component of the velocity is computed. For 1D only.

Examples

Compute the thermal injection velocity along the [110] direction relative to the device axes in a 1D device:

```
set velocity [ComputeVinj direction=[list 1 1 0]]
```

The computed results are added automatically to the bias log file, and the total velocity is returned to the Tcl variable `velocity`.

Dopant

This command defines a new dopant. After a dopant is defined, the active concentration of the dopant will be read from the TDR file during the next `LoadDevice` command.

Syntax

```
Dopant material=String name=String symbol=String type=String  
[print] [remove] [removeAll]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
material	Sets the material for which the dopant is defined. You also can use all, insulator, or semiconductor.
name	Sets the name of the dopant, for example, Boron.
print	Specify this option to print a list of the defined dopants.
remove	Specify this option to remove a defined dopant by name for a specific material.
removeAll	Specify this option to remove all defined dopants.
symbol	Sets the element symbol of the dopant, for example, B.
type	Sets the dopant type, either donor or acceptor.

Examples

Define boron as an acceptor dopant in silicon with element symbol B:

```
Dopant material=Silicon name=Boron symbol=B type=acceptor
```

Extract

This command extracts a scalar value from the named model. It can either extract a model value at a point or integrate the model over a region or a nonlocal line or nonlocal area.

Syntax

```
Extract model=String { z=Double | {x=Double y=Double} |
                      { {nonlocal=String | region=String} integral }
                    }
```

Argument	Description
integral	Specify this option to integrate the model over the specified region.
model	Specifies a valid model keyword.
nonlocal	Sets the name of the nonlocal line or nonlocal area over which to integrate the model.
region	Sets the name of the region over which to integrate the model.
x	Specifies the x-coordinate where the model value should be extracted, in μm . For 2D only.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
y	Specifies the y-coordinate where the model value should be extracted, in μm . For 2D only.
z	Specifies the z-coordinate where the model value should be extracted, in μm . For 1D only.

Examples

Extract the value of the conduction band energy at the z-coordinate of 0.0 μm and place the result into the Tcl variable named `Ec`:

```
set Ec [Extract model=ConductionBandEnergy z=0.0]
```

GetLast

This command obtains the latest value of the named field from the bias log file.

Syntax

```
GetLast name=String
```

Argument	Description
name	Sets the name of the field from which to retrieve its latest value from the bias log file.

Examples

Extract the latest value for the field named `V(Gate)` from the bias log file and return it to the Tcl variable `Vgate`:

```
set Vgate [GetLast name="V(Gate)"]
```

LoadDevice

This command loads a 1D or 2D device structure from a TDR file. See [Loading a 1D Device Structure on page 142](#).

Syntax

```
LoadDevice tdrFile=String [ignoreDopants=Integer]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
ignoreDopants	When set to 1, individual active dopants in the TDR file are ignored. Default: 0 (that is, known dopants are read from the TDR file)
tdrFile	Sets the name of a TDR file to load.

Examples

Load the TDR file `moscap.tdr` including any doping-related fields and contacts:

```
LoadDevice tdrFile=moscap.tdr
```

Material

This command adds new materials to Sentaurus Band Structure or outputs information about existing materials.

Syntax

```
Material getList | getNames | {getType name=String} |  
{name=String type=String}
```

Argument	Description
getList	Specify this option to output a Tcl list of known materials and their material type.
getNames	Specify this option to output a Tcl list of known materials by name.
getType	Specify this option to output the type of the named material.
name	Sets the name of a material.
type	Specifies the type of the material. Options are: <ul style="list-style-type: none">• insulator• semiconductor

Examples

Add a new material named `Germanium`, specified as a semiconductor:

```
Material name=Germanium type=semiconductor
```

Models and parameters for this material can then be specified.

Math

This command is used to create the geometry of a nonlocal line or nonlocal area to set parameters related to the solution of the Poisson and Schrödinger equations, and the mobility calculations. For the specification of a nonlocal line or nonlocal area, only mesh elements fully contained within the specified geometry parameters are made part of the nonlocal line or nonlocal area.

Note:

Convergence of the Poisson equation requires that both the update and residual tolerances be met, that is, `potentialUpdateTolerance` and `residualTolerance`.

Syntax

```
Math { nonlocal name=String [regions=List]
      { [minZ=Double] [maxZ=Double] } |
      { [minX=Double] [maxX=Double] [minY=Double] [maxY=Double] }
    }
  {
    [confinedEigensolver=Integer] [damping=Boolean]
    [doOnFailure=Integer] [dynamicSubbandThreshold=Double]
    [iterations=Integer]
    [maxInnerIter=Integer] [potentialUpdateClamp=Double]
    [potentialUpdateTolerance=Double] [residualTolerance=Double]
    [useDonettiOverlapFix=Boolean]
  }
```

Argument	Description
<code>confinedEigensolver</code>	Specifies which eigensolver to use for the confined $k \cdot p$ Schrödinger equations. Options are: <ul style="list-style-type: none">• 0: Arpack• 1: Dense Lapack• 2: Banded Lapack (default for 1D)• 3: Banded Lapack with parallelized ‘inverse iteration’ algorithm (see second example for further information)• 4: Arpack shift-and-invert (default for 2D)
<code>damping</code>	Specifies whether to use a damping algorithm during the solution of the Poisson equation to improve convergence. Options are: <ul style="list-style-type: none">• 0: Do not use damping (default).• 1: Use damping.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
doOnFailure	Specifies the action to take when the Poisson equation does not converge. Options are: <ul style="list-style-type: none">• 0: Continue the simulation.• 1: Issue a Tcl error that can be treated with the <code>catch</code> command (default).
dynamicSubbandThreshold	Sets the threshold for deactivating the dynamic subband evaluation (unit: V). Default: 5e-3
iterations	Sets the maximum number of allowed Newton iterations. Default: 50
maxInnerIter	Sets the maximum number of allowed internal iterations for the nonlinear Poisson problem. The default value of 1 results in a Corrector step. A value greater than 1 allows for additional Predictor steps before the Corrector step. Default: 1
maxX	Sets the maximum x-coordinate of the nonlocal area, in μm . For 2D only. Default: ∞
maxY	Sets the maximum y-coordinate of the nonlocal area, in μm . For 2D only. Default: ∞
maxZ	Sets the maximum z-coordinate of the nonlocal line, in μm . For 1D only. Default: ∞
minX	Sets the minimum x-coordinate of the nonlocal area, in μm . For 2D only. Default: $-\infty$
minY	Sets the minimum y-coordinate of the nonlocal area, in μm . For 2D only. Default: $-\infty$
minZ	Sets the minimum z-coordinate of the nonlocal line, in μm . For 1D only. Default: $-\infty$
name	Sets the name of the nonlocal line or nonlocal area.
nonlocal	Specifies that a nonlocal line or nonlocal area should be defined with the specified name.
potentialUpdateClamp	Sets the absolute value of the maximum-allowed update applied to the potential during one Newton iteration in volts. Default: 1
potentialUpdateTolerance	Sets the convergence tolerance applied to the potential update in units of the thermal voltage: $k_B T/q$. Default: 1.0e-5

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
regions	Specifies a list of regions to be included in the nonlocal line or nonlocal area. Default: All regions
residualTolerance	Sets the convergence tolerance applied to the residual of the Poisson equation in units of C/cm ² . Default: 1.0
useDonettiOverlapFix	Specifies whether to compute the wavefunction overlap form-factor between degenerate subbands from the intra-subband form-factor. Options are: <ul style="list-style-type: none">• 0: Do not use this modification.• 1: Use this modification (default).

Examples

Create a nonlocal line with the name `NL1` consisting of mesh elements from the region named `sil` and made of points that fall within $z=0$ μm to $z=0.02$ μm:

```
Math nonlocal name=NL1 minZ=0 maxZ=20.0e-3 regions=[List sil]
```

Switch from the default banded Lapack eigensolver (for calculating both confined $\mathbf{k} \cdot \mathbf{p}$ eigenenergies and wavefunctions) to an alternative scheme that uses the standard Lapack eigensolver only for calculating eigenvalues:

```
Math confinedEigensolver=3
```

Eigenvectors are then computed from the eigenvalues by a parallelized ‘inverse iteration’ scheme. In scenarios that do not use \mathbf{k} -dependent wavefunctions, this alternative algorithm can provide improved scalability on machines with a large number of CPU cores.

Progress bars (see `showProgressBars` in [sBandGet and sBandSet on page 273](#)) offer some guidance on the choice of algorithm:

- The time taken for the wavefunction calculation at the subband minimum is framed by < and >.
- Progress of the calculation of the subband dispersion on the polar grid is indicated by a number of dots (.)
- If the completion marker (>) of the wavefunction calculation is very close to the end of the progress bar, then this typically indicates a scaling bottleneck, which might be overcome by switching from eigensolver mode 2 to mode 3.

Physics

The `Physics` command specifies a range of physical models and parameters. The types of model and parameter that can be specified are separated into different groups:

- [Physics for Top-Level Parameters](#)
- [Physics for Contacts](#)
- [Physics for Electrostatic Models](#)
- [Physics for Valley Models](#)
- [Physics for Scattering Models](#)
- [Physics for Trap Models](#)
- [Physics for Interface Potential Spike Models](#)
- [Physics of Nonlocal Lines or Nonlocal Areas](#)

Physics for Top-Level Parameters

This command sets the top-level global device parameters that define the device axes and temperature. It also allows printing of all defined models.

Syntax

```
Physics {[surfaceOrientation=RealVector3D] [xDirection=RealVector3D]} |  
  {[xDirection=RealVector3D] [yDirection=RealVector3D]} |  
  {[wurtziteXOrientation=RealVector4D]  
   [wurtziteYOrientation=RealVector4D]  
   [wurtziteZOrientation=RealVector4D]} |  
  {[doesModelExist {material=String | region=String} name=String  
   [model=String]]} |  
  {[getCrystalOrientation] |  
   [getRegionMaterialMapping] |  
   [getSubbandNumbers] |  
   [print] [temperature=Double]}
```

Argument	Description
<code>doesModelExist</code>	Specify this option to return a string to check whether a model, given by <code>name</code> , has been specified in a material or region. This option also allows you to check the model type by using <code>model</code> . Note: This option must be used separately from all other options and arguments.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
getCrystalOrientation	Specify this option to return the crystal orientation as a RealMatrix3D. Note: This option must be used separately from all other options and arguments.
getRegionMaterialMapping	Specify this option to return a list of regions and corresponding materials for the current device as a List. Note: This option must be used separately from all other options and arguments.
getSubbandNumbers	Specify this option to return a list of subbands for each ladder as a List data structure. Note: This option must be used separately from all other options and arguments.
material	Specifies a material name.
model	Sets the type of the model for which to check.
name	Sets the name of a model for which to check whether it has been specified in a material or region.
print	Specify this option to print a list of all models in all regions. The list is printed to the screen and the output log file.
region	Specifies a region name.
surfaceOrientation	For 1D, sets the surface orientation relative to the crystal axes. Default: [list 0 0 1]
temperature	Sets the temperature in kelvin. Default: 300
wurtziteXOrientation	Sets the direction of the x-axis relative to the hexagonal crystal axes.
wurtziteYOrientation	Sets the direction of the y-axis relative to the hexagonal crystal axes.
wurtziteZOrientation	Sets the direction of the z-axis relative to the hexagonal crystal axes.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
xDirection	For 1D, sets the direction of the in-plane x-axis relative to the crystal axes. Default: [list 1 0 0] For 2D, sets the direction of the confinement x-axis relative to the crystal axes. Default: [list 1 0 0]
yDirection	For 2D, sets the direction of the confinement y-axis relative to the crystal axes. Default: [list 0 1 0]

Examples

For a 1D device, set the surface orientation to [011], the x-axis direction to [100], and the ambient temperature to 77 K:

```
Physics surfaceOrientation=[list 0 1 1] xDirection=[list 1 0 0] \
temperature=77.0
```

Print a list of the currently defined models in Sentaurus Band Structure:

```
Physics print
```

Print a list of subband numbers per ladder resolved within the last Schrödinger solution:

```
set ladderList [Physics getSubbandNumbers];
foreach ladder $ladderList {
    set ladderName [lindex $ladder 0];
    set Nsubbands [lindex $ladder 1];
    puts [format "Ladder %s; Nsubbands= %d" $ladderName $Nsubbands];
}
```

For a device containing wurtzite semiconductors, specify the Bravais–Miller indices to align the z-axis with the wurtzite c-axis and the y-axis with the wurtzite a2-axis:

```
Physics wurtziteZOrientation=[list 0 0 0 1] \
wurtziteYOrientation=[list 0 1 0 0]
```

Physics for Contacts

This command sets the workfunction at a contact.

Syntax

```
Physics contact=String workfunction=Double
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
contact	Sets the name of the contact.
workfunction	Sets the workfunction value for the contact in eV. Default: 4.10

Examples

Set the workfunction of the gate contact to 5.2 eV:

```
Physics contact=gate workfunction=5.2
```

Physics for Electrostatic Models

Various electrostatic models are used during the solution of the Poisson equation. These models are specified for particular regions or materials. Each model is identified by a model keyword for which a particular model can be selected. Model parameters specific to each selected model are also specified using the `Physics` command.

Syntax

```
Physics {material=String | region=String} <ModelKeyword>=String  
<paraName1>=<value1> ...
```

Argument	Description
material	Sets the name of a material in the device. All regions of this material will have the same model and parameters set. Specify either <code>material</code> or <code>region</code> but not both. Convenient aliases are provided: <ul style="list-style-type: none">• <code>all</code> for all regions• <code>insulator</code> for all insulator regions• <code>semiconductor</code> for all semiconductor regions
<ModelKeyword>	This is a keyword for a built-in model. The value of this argument selects a specific model to use for the calculation of the built-in model.
<paraName1>	This is the name of any valid parameter for the selected model and is used to set the value of that parameter.
region	Sets the name of a region in the device for which the given model and parameters are set. Specify either <code>material</code> or <code>region</code> but not both.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Examples

Set the model for the electron density, with the model keyword `eDensity`, to the Fermi–Dirac model for electrons, with the name `eFermiDensity`:

```
Physics material=Silicon eDensity=eFermiDensity Nc=2.9e19
```

This model has one adjustable parameter, `Nc`, the effective conduction band DOS at 300 K. Here, it is set to $2.9 \times 10^{19} \text{ cm}^{-3}$.

Modifying Parameters

When you specify a physical model and an initial set of parameters, you can subsequently modify the model parameters with another `Physics` command by using only the model keyword. For example, if the `eDensity` model is already set to the `eFermiDensity` model, then the `Nc` parameter can be modified using:

```
Physics material=Silicon eDensity Nc=2.6e19
```

Switching Models

You can change the selected model for a model keyword with another `Physics` command that specifies the new model name. For example, the following commands first set the `eDensity` model to the Fermi–Dirac model and then change it to the multivalley model:

```
Physics material=Silicon eDensity=eFermiDensity Nc=2.9e19  
Physics material=Silicon eDensity=eMultiValleyDensity
```

Note:

When a new model is selected, the parameters of the old model are lost. Subsequently, reverting the model to the original model will not recover any modified parameters.

Physics for Valley Models

Valley models are a special category of model. They hold parameters representing the band structure around valley extrema in \mathbf{k} -space. Valley models are multimodels in that more than one `ValleyModel` can be defined for each region. To be able to refer to a particular `ValleyModel`, a name for the `ValleyModel` must be specified.

Syntax

```
Physics {material=String | region=String}  
degeneracy=Integer name=String ValleyModel=String  
<paraName1>=<value1> ...  
[remove] [removeAll]  
[useForEBulkDensity=Boolean | useForHBulkDensity=Boolean]  
[virtualBandEdgeShift=Double]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
degeneracy	Sets the valley degeneracy.
material	Sets the name of a material in the device. All regions of this material will have the same model and parameters set. Specify either <code>material</code> or <code>region</code> but not both. Convenient aliases are provided: <ul style="list-style-type: none">• <code>all</code> for all regions• <code>insulator</code> for all insulator regions• <code>semiconductor</code> for all semiconductor regions
name	Sets a unique name for this valley. This name is used by other models to refer to this valley.
<paraName1>	This is the name of any valid parameter for the selected valley model and is used to set the value of that parameter.
region	Sets the name of a region in the device for which the given model and parameters are set. Specify either <code>material</code> or <code>region</code> but not both.
remove	Specify this option to remove a particular valley model.
removeAll	Specify this option to remove all valley models.
useForEBulkDensity	Specify whether to create a bulk electron density model automatically. Options are: <ul style="list-style-type: none">• 0: Do not create bulk model (default).• 1: Create bulk model.
useForHBulkDensity	Specify whether to create a bulk hole density model automatically. Options are: <ul style="list-style-type: none">• 0: Do not create bulk model (default).• 1: Create bulk model.
ValleyModel	Sets the name of the selected valley model.
virtualBandEdgeShift	Specify a constant energy shift to the dispersion relation computation.

Examples

Create a valley model based on the `2kpEllipsoid` model with the name `Delta1`:

```
Physics material=Silicon ValleyModel=2kpEllipsoid name=Delta1 \
degeneracy=2 longAxis=100 useForEBulkDensity=1 ml=0.92
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

The valley degeneracy is specified as 2, and this command sets the longitudinal axis and `m1` parameters of the model to 100 and 0.92, respectively. Specifying `useForEBulkDensity=1` automatically creates a model to compute the bulk electron density for this valley.

Modifying Parameters

When you specify a valley model and an initial set of parameters, you can subsequently modify the model parameters with another `Physics` command by using only the `ValleyModel` keyword and the name of the valley.

For example, the `Delta1` valley created in the previous example can be modified using:

```
Physics material=Silicon ValleyModel name=Delta1 m1=0.916
```

Switching Models

You can change the selected model for a valley model with another `Physics` command that specifies the new model name along with the name of the original valley. For example, the model for the `Delta1` valley created in the previous examples can be changed using:

```
Physics material=Silicon ValleyModel=ConstantEllipsoid name=Delta1 \
degeneracy=2 m1=0.92
```

Note:

When a new `ValleyModel` is selected for an existing valley, the parameters of the old model are lost. Subsequently, reverting the model to the original model will not recover any modified parameters.

Removing Valley Models

You can remove a valley model from use by Sentaurus Band Structure by specifying the `ValleyModel` keyword and valley name along with `remove`. For example, using the previous `Delta1` example:

```
Physics material=Silicon ValleyModel name=Delta1 remove
```

To remove all valley models used by Sentaurus Band Structure, use `removeAll`:

```
Physics material=Silicon ValleyModel removeAll
```

Physics for Scattering Models

Scattering models are a special category of model. They represent the different scattering mechanisms used during the mobility calculation. Multiple scattering models can be specified using multiple `Physics ScatteringModel=String` commands.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Syntax

```
Physics {material=String | region=String}  
    hbarOmega=Double inelasticType=String name=String  
    ScatteringModel=String transitionType=String valleys=List  
    <paraName1>=<value1> ...  
    [remove] [removeAll]
```

Argument	Description
hbarOmega	For inelastic phonon models, sets the phonon energy in eV. Default: 0.0
inelasticType	For inelastic models, specifies whether the transition is for absorption or emission. Options are: <ul style="list-style-type: none">• ABS: Absorption (default)• EMS: Emission
material	Sets the name of a material in the device. All regions of this material will have the same model and parameters set. Specify either <code>material</code> or <code>region</code> but not both. Convenient aliases are provided: <ul style="list-style-type: none">• <code>all</code> for all regions• <code>insulator</code> for all insulator regions• <code>semiconductor</code> for all semiconductor regions
name	Sets a unique name for the scattering model. This name can be used in subsequent <code>Physics</code> commands to modify model parameters.
<paraName1>	This is the name of any valid parameter for the selected model and is used to set the value of that parameter.
region	Sets the name of a region in the device for which the given model and parameters are set. Specify either <code>material</code> or <code>region</code> but not both.
remove	Specify this option to remove a particular scattering model.
removeAll	Specify this option to remove all scattering models.
ScatteringModel	Sets the name of the selected scattering model.
transitionType	Specifies the type of valley transition allowed. Options are: <ul style="list-style-type: none">• gIntervalley• Intervalley• Intrasubband• Intravalley (default)
valleys	Specifies a list of valley pairs denoting the allowed transitions.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Examples

Add a model for elastic acoustic phonon-scattering to the list of scattering models in all silicon regions:

```
Physics material=Silicon ScatteringModel=ElasticAcousticPhonon \
    name=AC1 valleys=[list Delta1 Delta2 Delta3] \
    transitionType=Intravalley Dac=6.8
```

This model is named `AC1` and is specified to apply only to intravalley transitions. Because only intravalley transitions are allowed, the valleys to which this model will be applied can be listed singly since the initial and final valleys are the same. Here, three `Delta` valleys are listed. One model-specific parameter, `Dac`, is also specified.

Add a model for f-type inelastic phonon scattering to all silicon regions:

```
set fValleys [list {Delta1 Delta2} {Delta1 Delta3} \
    {Delta2 Delta1} {Delta2 Delta3} \
    {Delta3 Delta1} {Delta3 Delta2} ]
Physics material=Silicon ScatteringModel=InelasticPhonon \
    name=fIV_ABS valleys=$fValleys transitionType=Intervalley \
    inelasticType=ABS hbarOmega=61.2e-3 DtK=1.0e8
```

The list of allowed transitions is specified as a list of initial/final valley pairs. The transition is specified as `Intervalley`, which applies to transitions between nonequivalent valleys. Since this transition is inelastic, the type of inelastic transition must be specified. Here, it is specified to be a transition involving phonon absorption. The phonon energy is set to 61.2 meV, and one model-specific parameter, `DtK`, is specified.

Modifying Parameters

When you specify a scattering model and an initial set of parameters, you can subsequently modify the model parameters with another `Physics` command by using only the `ScatteringModel` keyword and the unique name of the model. For example, the model named `fIV_ABS` created in the previous example can be modified using:

```
Physics material=Silicon ScatteringModel name=fIV_ABS DtK=2.0e8
```

Switching Models

You can change the selected model for a `ScatteringModel` with another `Physics` command that specifies the new model name along with the name of the original model.

Note:

When a new `ScatteringModel` is selected, the parameters of the old model are lost. Subsequently, reverting the model to the original model will not recover any modified parameters.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Removing Scattering Models

You can remove a scattering model from use by Sentaurus Band Structure by specifying `ScatteringModel` and the model name along with `remove`. For example, using the previous `fIV_ABS` example:

```
Physics material=Silicon ScatteringModel name=fIV_ABS remove
```

To remove all scattering models used by Sentaurus Band Structure, specify `removeAll`:

```
Physics material=Silicon ScatteringModel removeAll
```

Physics for Trap Models

Trap models are a special category of model that allow for the specification of interface charge. Two types of interface charge can be specified: fixed charge and donor (or acceptor) traps. You can specify multiple values of interface charge by using multiple `Physics TrapModel=String` commands.

Syntax

```
Physics {materialInterface=String | regionInterface=String}
    TrapModel=String [name=String]
    [remove] [removeAll]
    conc=String
    carrierType=String
    DitProfile=String
    [conc=Double] [DitTable=List]
    [EnergyMid=String] [EnergySig=Double]
    [Emin=String] [Emax=String] [g=Double]
    [refMaterial=String | refRegion=String]
```

For TrapModel=FixedCharge
} For
} TrapModel=Donor
} or
} TrapModel=Acceptor
}

Argument	Description
<code>carrierType</code>	Selects which quasi-Fermi energy is used in the distribution function for trap occupancy. Options are <code>electron</code> or <code>hole</code> . Default: <code>electron</code>
<code>conc</code>	Sets the interface trap concentration: <ul style="list-style-type: none">For fixed charge, this specifies a fixed charge in units of cm^{-2} and can be positive or negative.For donor or acceptor traps, <code>conc</code> is used by the analytic D_{it} profiles and is specified in units of $\text{eV}^{-1}\text{cm}^{-2}$ and must be positive.
<code>DitProfile</code>	Specifies the profile to use for the interface trap density. Options are <code>Exponential</code> , <code>Gaussian</code> , <code>Table</code> , or <code>Uniform</code> . See Energy Profile on page 197 .

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
DitTable	For <code>DitProfile=Table</code> , this argument specifies the D_{it} profile as a Tcl list of {Energy Dit} pairs in which the Energy is in eV relative to the relaxed valence band and Dit is in units of $\text{eV}^{-1}\text{cm}^{-2}$.
Emax	Sets the maximum energy of the D_{it} profile, which is specified relative to the relaxed valence band. Options are: <ul style="list-style-type: none">• CondBand: Relaxed conduction band edge• ValBand: Relaxed valence band edge• MidBandGap: Middle of the relaxed band gap• Numeric value: Value interpreted as relative to the relaxed valence band edge The default for analytic profiles is CondBand. The default for <code>DitTable</code> is the maximum energy from the list.
Emin	Sets the minimum energy of D_{it} profile, which is specified relative to the relaxed valence band. Options are: <ul style="list-style-type: none">• CondBand: Relaxed conduction band edge• ValBand: Relaxed valence band edge• MidBandGap: Middle of the relaxed band gap• Numeric value: Value interpreted as relative to the relaxed valence band edge The default for analytic profiles is ValBand. The default for <code>DitTable</code> is the minimum energy from the list.
EnergyMid	Middle energy parameter used by exponential and Gaussian profiles. Options are: <ul style="list-style-type: none">• CondBand: Relaxed conduction band edge• ValBand: Relaxed valence band edge• MidBandGap: Middle of the relaxed band gap• Numeric value: Value interpreted as relative to the relaxed valence band edge Default: MidBandGap
EnergySig	Sets the decay parameter used by exponential and Gaussian profiles. Default: 0.1 eV
g	Sets the degeneracy parameter for trap occupancy distribution functions. Default: 1

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
materialInterface	Sets the name of region interfaces between two materials. Material names must be separated by a slash (/), for example, "Silicon/Oxide". All interfaces between regions of these two materials will have the same model and parameters set. Specify either <code>materialInterface</code> or <code>regionInterface</code> , but not both.
name	Sets a unique name for the trap model. This name can be used in subsequent <code>Physics</code> commands to modify model parameters.
refMaterial	Sets the material that supplies the reference energy bands for the D_{it} profile. For semiconductor-insulator interfaces, this is given by the semiconductor region by default.
refRegion	Sets the region that supplies the reference energy bands for the D_{it} profile. For semiconductor-insulator interfaces, this is given by the semiconductor region by default.
regionInterface	Sets the name of an interface between two regions. Region names must be separated by a slash, for example, "sil/ox1". Specify either <code>materialInterface</code> or <code>regionInterface</code> , but not both.
remove	Specify this option to remove a particular trap model.
removeAll	Specify this option to remove all trap models.
TrapModel	Sets the name of the selected trap model. Options are <code>Acceptor</code> , <code>Donor</code> , or <code>FixedCharge</code> .

Examples

Add a model for a fixed interface charge to the list of interface charge models at the interface between the regions named `sil` and `ox1`:

```
Physics regionInterface="sil/ox1" TrapModel=FixedCharge name=Nss1 \
conc=le12
```

This model is named `Nss1` and is given a surface charge concentration of 10^{12} cm^{-2} using the `conc` model argument.

Create a donor interface trap at the boundary between the `sil` and `ox1` regions.

```
Physics regionInterface="sil/ox1" TrapModel=Donor \
name=Dit1 DitProfile=Table \
DitTable=[list {0.0 5.0e12} {0.5 2.0e12} {1.0 1.0e12}] \
Emax=MidBandGap carrierType=electron
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

The D_{it} profile is specified as a table with three entries and extends from 0.0, that is, the valence band edge, since energies are specified relative to the valence band, to mid-gap. The `carrierType` argument indicates that the electron quasi-Fermi energy should be used to compute the trap occupancy.

Modifying Parameters

When you specify a trap model and an initial set of parameters, you can subsequently modify the model parameters with another `Physics` command by using only `TrapModel` and the unique name of the model. For example, the model named `Nss1` created in the previous example can be modified using:

```
Physics regionInterface="sil/ox1" TrapModel name=Nss1 conc=2e12
```

Removing Trap Models

You can remove a trap model from use by Sentaurus Band Structure by specifying `TrapModel` and the model name along with `remove`. For example, using the `Nss1` example:

```
Physics regionInterface="sil/ox1" TrapModel name=Nss1 remove
```

To remove all trap models used by Sentaurus Band Structure, specify `removeAll`:

```
Physics regionInterface="sil/ox1" TrapModel removeAll
```

Physics for Interface Potential Spike Models

Potential spike models are a special category of model that allows for the specification of spike-like potential barriers or wells at region interfaces. Multiple interface spike models can be specified by using multiple `Physics SpikeModel=String` commands.

Syntax

```
Physics {materialInterface=String | regionInterface=String}
        SpikeModel=String valleys=List value=Double
        [name=String] [remove] [removeAll]
```

Argument	Description
materialInterface	Sets the name of region interfaces between two materials. Material names must be separated by a slash (/). For example: "Silicon/Oxide" All interfaces between regions of these two materials will have the same model and parameters set. Specify either <code>materialInterface</code> or <code>regionInterface</code> , but not both.
name	Sets a unique name for the spike model. This name can be used in subsequent <code>Physics</code> commands to modify model parameters.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
regionInterface	Sets the name of an interface between two regions. Region names must be separated by a slash. For example: "sil/ox1" Specify either materialInterface or regionInterface, but not both.
remove	Specify this option to remove a particular spike model.
removeAll	Specify this option to remove all spike models.
SpikeModel	Sets the name of the selected spike model. The only option is ConstantSpike.
valleys	Specifies a list of valleys to apply to this SpikeModel.
value	Sets the value of the potential spike in units of Vcm. Positive values produce a spike-like barrier at the interface; negative values produce a spike-like well at the interface.

Examples

Add a spike model at the interface between the regions named sil and ox1:

```
Physics regionInterface="sil/ox1" SpikeModel=ConstantSpike \
    name=DeltaSpike value=2.0e-9 valleys=[list Delta1 Delta2 Delta3]
```

The model is named DeltaSpike and is given a potential spike value of $2.0 \cdot 10^{-9}$ Vcm using the value argument. Using the valleys argument, this SpikeModel is restricted to the valleys named Delta1, Delta2, and Delta3.

Modifying Parameters

After you specify a spike model and an initial set of parameters, you can subsequently modify the model parameters with another Physics command by using only SpikeModel and the unique name of a model. For example, the model named DeltaSpike previously created can be modified using:

```
Physics regionInterface="sil/ox1" SpikeModel name=DeltaSpike \
    value=-1.0e-9
```

Removing Spike Models

You can remove a spike model from use by Sentaurus Band Structure by specifying SpikeModel, the model name, and remove. For example, using the DeltaSpike model:

```
Physics regionInterface="sil/ox1" SpikeModel name=DeltaSpike remove
```

To remove all spike models used by Sentaurus Band Structure, specify removeAll:

```
Physics regionInterface="sil/ox1" SpikeModel removeAll
```

Physics of Nonlocal Lines or Nonlocal Areas

Nonlocal lines or nonlocal areas are used to indicate where a Schrödinger solve and mobility calculation should be performed. Nonlocal lines or nonlocal areas are created using the `Math` command.

Using the `Physics` command, one Schrödinger solver and one mobility calculator can be assigned to a nonlocal line or nonlocal area. In addition, various parameters used by the Schrödinger solver and mobility calculator can be specified.

Note:

You must specify the Schrödinger solver and the mobility calculator for a nonlocal line in separate `Physics` commands. The Schrödinger solver must be defined first.

The only polar grid parameters that you can specify for a mobility calculator are `kGrid`, `Nk`, and `Nphi`. The other parameters are taken from the corresponding Schrödinger solver previously defined for the same nonlocal line.

Syntax

For specifying a Schrödinger solver:

```
Physics nonlocal=String
  {eSchrodinger=String | hSchrodinger=String}
  {kGrid=List/Double | {[Kmax=Integer] [Nk=Integer]}}
  valleys=List/String
  <paraName1>=<value1> ...
  [a0=Double] [degenSubbandEnergySplit=Double]
  [dynamicSubbandEnergyCutoff=Double]
  [iwSymmetry=String]
  [kGridForNinv=List/Double | NkForNinv=Integer]
  [maxSubbands=Integer] [minSubbands=Integer]
  [Nphi=Integer] [Nsubbands=Integer] [phi0=Double]
  [remove] [reorderDispersion:Boolean]
  [useDynamicSubbands:Boolean] [useKdependentWF:Boolean]
  [useOverlapInterpolation:Boolean]
```

For specifying a mobility calculator:

```
Physics nonlocal=String
  {eMobilityCalculator=String | hMobilityCalculator=String}
  {kGrid=List/Double | {[Kmax=Integer] [Nk=Integer]}}
  <paraName1>=<value1> ...
  [Nphi=Integer] [remove]
  [sdfDegenTol=Double] [tdfDegenTol=Double]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
a0	Scales the k -vectors in the polar grid to units of cm^{-1} by multiplying the k -vectors by $2\pi/a_0$. Default: 5.43e-8 cm
degenSubbandEnergySplit	Sets a value (in eV) that is added to the diagonal of the $k \cdot p$ Hamiltonians to split degenerate subbands (see Handling Subband Degeneracies on page 223). Default: 0
dynamicSubbandEnergyCutoff	Sets the energy cutoff value for the evaluation of the dynamic subband number (unit: eV). This cut-off energy is added to the Fermi level. Default: 5 $k_B T$
eMobilityCalculator	Sets the name of a particular mobility calculation approach to use for electrons on the nonlocal line.
eSchrodinger	Sets the name of a Schrödinger solver to use for electrons on the nonlocal line.
hMobilityCalculator	Sets the name of a particular mobility calculation approach to use for holes on the nonlocal line.
hSchrodinger	Sets the name of a Schrödinger solver to use for holes on the nonlocal line.
iwSymmetry	For 1D, specifies the polar symmetry to apply to the calculation of the dispersion. The subband dispersion is computed only over the irreducible wedge (IW) determined by <code>iwSymmetry</code> and <code>phi0</code> . Options are: <ul style="list-style-type: none">• AUTO: Symmetry is detected automatically; only used for six-band $k \cdot p$ Schrödinger solver (<code>hSchrodinger=6kp</code>)• IW12: Twelve-fold symmetry• IW8: Eight-fold symmetry• IW4: Four-fold symmetry• IW2: Two-fold symmetry (default)
kGrid	Specifies a list of values to use for the k -grid along the radial direction for the calculation of the dispersion (unit: $2\pi/a_0$).
kGridForNinv	Specifies a list of values to use for the k -grid along the radial direction for the calculation of the carrier density (unit: $2\pi/a_0$).
Kmax	Sets the maximum value of the radial k -vector magnitude for the calculation of the dispersion (unit: $2\pi/a_0$). Default: 0.3

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
maxSubbands	Sets the maximum number of subbands that can be resolved during the dynamic subband evaluation. To deactivate this feature, set a value of -1. Default: -1
minSubbands	Sets the minimum number of subbands that can be resolved during the dynamic subband evaluation. Default: 2
Nk	Sets the number of uniformly spaced points along the radial direction for the calculation of the dispersion. Default: 11
NkForNinv	Sets the number of uniformly spaced points in the radial direction for the calculation of the carrier density. Default: 22
nonlocal	Sets the name of an existing nonlocal line or nonlocal area.
Nphi	Sets the number of uniformly spaced points along the angular direction from 0 to 2π in the polar representation of the dispersion. Ignored for 2D. Default: 48
Nsubbands	Sets the number of subbands for which to solve. Default: 8
<paraName1>	This is the name of any valid parameter for the selected Schrödinger solver or mobility calculator, and it is used to set the value of that parameter.
phi0	Sets the starting angular φ value (in radians) to use in conjunction with <code>iwSymmetry</code> . Ignored for 2D. Default: 0.0
remove	Specify this option to remove a Schrödinger solver or mobility calculator.
reorderDispersion	Specifies whether the default order of the subbands for each k -point is based on the energy value. If reordering of the subband dispersion is switched on, then the order of the subbands is determined through wavefunction overlap in k -space. Options are: <ul style="list-style-type: none">• 0: Do not reorder subband dispersion (default).• 1: Reorder subband dispersion.
sdfDegenTol	Sets the subband energy difference tolerance for screening inter-subband transitions with the scalar Lindhard screening model. Setting <code>sdfDegenTol</code> to a negative value means degenerate subbands are ignored when computing and applying the scalar dielectric function. Default: 1.0e-4 eV

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
tdfDegenTol	Sets the subband energy difference tolerance for screening inter-subband transitions with the tensor-based Lindhard screening model. Default: -1
useDynamicSubbands	Specifies whether to dynamically determine subband numbers during the Schrödinger solution. Options are: <ul style="list-style-type: none">• 0: Use static subband numbers determined by Nsubbands (default).• 1: Allow dynamic subband numbers.
useKdependentWF	Specifies whether to compute wavefunctions at each k -point in the k -space grid. Options are: <ul style="list-style-type: none">• 0: Only evaluate wavefunctions at the subband minimum (default).• 1: Evaluate wavefunctions at each k-point.
useOverlapInterpolation	Specifies whether to linearly interpolate between k -dependent values of the isotropic overlap cache. Options are: <ul style="list-style-type: none">• 0: Use constant value interpolation (default).• 1: Use linear interpolation.
valleys	Specifies a list of valleys by name that the Schrödinger solver will use.

Examples

Assign the Parabolic Schrödinger solver to the nonlocal line NL1:

```
Physics nonlocal=NL1 eSchrodinger=Parabolic \
    valleys=[list Delta1 Delta2 Delta3] Nk=8 Nphi=16 phi=0.0 \
    iwSymmetry=IW4 Kmax=0.25 Nsubbands=8 correction=3
```

It is specified to solve the Schrödinger equation for the valleys named Delta1, Delta2, and Delta3. A polar grid with 8 radial points from 0 to Kmax=0.25 [$2\pi/a_0$] and 16 angular points starting from phi=0 radians is specified. The IW symmetry is set to IW4, that is, a four-fold symmetry. The lowest eight subbands will be computed for each valley. The correction argument for the Parabolic Schrödinger solver specifies that model 3 for the nonparabolic correction should be used.

Assign the KGFromK mobility calculator to the nonlocal line NL1 for computing the electron mobility:

```
Physics nonlocal=NL1 eMobilityCalculator=KGFromK Nk=64 Nphi=48
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

A polar grid with 64 uniformly spaced grid points in k and 48 uniformly spaced grid points in ϕ is specified. The `Kmax` value for the radial grid is taken from that previously defined for the Schrödinger solver on the same nonlocal line.

Modifying Parameters

After you define a Schrödinger solver or mobility calculator for a nonlocal line or nonlocal area, and specify an initial set of parameters, you can subsequently modify the parameters with another `Physics` command by using only `eSchrodinger`, `hSchrodinger`, `eMobilityCalculator`, or `hMobilityCalculator` along with the name of the nonlocal line.

For example, a previously defined Schrödinger solver for electrons on the nonlocal line `NL1` can be modified using:

```
Physics nonlocal=NL1 eSchrodinger Nk=22
```

Switching Schrödinger Solvers and Mobility Calculators

You can change the selected Schrödinger solver or mobility calculator for a nonlocal line or nonlocal area with another `Physics` command that specifies the new Schrödinger solver or mobility calculator name along with the name of the nonlocal line.

Note:

When a new Schrödinger solver or mobility calculator replaces an existing one, the original parameters are lost. Subsequently, reverting the model to the original Schrödinger solver or mobility calculator will not recover any modified parameters.

Removing Schrödinger Solvers and Mobility Calculators

You can remove a Schrödinger solver or mobility calculator from use by Sentaurus Band Structure by specifying `eSchrodinger`, `hSchrodinger`, `eMobilityCalculator`, or `hMobilityCalculator` along with the name of the nonlocal line or nonlocal area followed by `remove`. For example, using the previous `NL1` example:

```
Physics nonlocal=NL1 eSchrodinger remove
```

Save

This command saves a real-space TDR file or a text file with the specified models as a function of the real-space device coordinates.

Syntax

To save to a text file:

```
Save dataFile=String models=List/String [highPrecision] [noHeader]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

To save to a TDR file:

```
Save tdrFile=String models=List/String [noXYTDR]
```

Argument	Description
dataFile	Sets the name of the text file to save.
highPrecision	If this option is not specified, then data with low precision is saved. If this option is specified, then data with higher precision is saved.
models	Specifies a Tcl list of models to save, given by their model keywords.
noHeader	Specify this option to omit the header of the text file. This option is especially useful for saving the vacuum potential used for the initial guess of the self-consistent process (since the <code>Solve biasFromPotFile=<i>String</i></code> or <code>Solve initialPotFile=<i>String</i></code> command accepts only a simple text file without header).
noXYTDR	For 1D structures, specify this option to save a structure-based TDR file that can be reread. By default, 1D structures are saved as TDR files, which are more convenient for visualization.
tdrFile	Sets the name of the TDR file to save.

Examples

Save a TDR file containing the solutions for the electron and hole densities and the conduction band energy over the device structure:

```
Save tdrFile=solution.tdr \
    models=[list eDensity hDensity ConductionBandEnergy]
```

Save to the text file `mypot.xy` the solutions for the vacuum potential with high precision as a function of the z-coordinate of the 1D device structure:

```
Save dataFile=mypot.xy noHeader highPrecision models=VacuumPotential
```

Here, the header of the file is omitted and only data is saved. The saved file `mypot.xy` can be used for the `Solve biasFromPotFile=mypot.xy` or `Solve initialPotFile=mypot.xy` command (see [Solve on page 386](#)).

SaveDitProfile

This command saves a TDR file containing profile information for interface trap-related quantities. These quantities are extracted for the interface node closest to the specified coordinates.

Syntax

```
SaveDitProfile tdrFile=String trapName=String
    { {x=Double y=Double} | z=Double }
```

Argument	Description
tdrFile	Sets the name of the TDR file.
trapName	Sets the name of the trap for which the interface trap quantities are extracted.
x	Specifies the x-coordinate in μm for 2D simulations.
y	Specifies the y-coordinate in μm for 2D simulations.
z	Specifies the z-coordinate in μm for 1D simulations.

SaveK

This command saves a TDR file containing k -space models for the subbands over a uniformly spaced, Cartesian, tensor-product mesh.

Note:

For a 1D device, while the k -space mesh used in the solution of the Schrödinger equation and the calculation of mobility is a polar grid, the mesh used to save data to the TDR file is a Cartesian tensor-product mesh. The extent and mesh spacing of this mesh can be controlled using the `Nk` and `Kmax` arguments.

The name of each k -space model saved must include a unique identifier for the required subband. This identifier is composed of the valley name, an underscore (`_`), and then the subband index. For example, `Gamma_0` would be used to identify the 0th subband of the `Gamma` valley. You can save the dispersion for this subband by specifying `Gamma_0_Dispersion`.

Syntax

```
SaveK tdrFile=String models=List/String
    [Kmax=Double] [Nk=Integer] [nonlocal=String]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
Kmax	Sets the maximum k -value in the Cartesian tensor-product mesh for the TDR file. The default value is determined from the k -space grid of the nonlocal line or nonlocal area.
models	Specifies a list of k -space models to save, prepended by the valley name and subband index. Valid models are Dispersion, IMRT, or LinearBTE for the linear BTE mobility calculator.
Nk	Sets the number of k -points to use in the Cartesian tensor-product mesh for the TDR file. Default: 101
nonlocal	Sets the name of the nonlocal line for which to save the subband models. By default, the first found nonlocal line is used.
tdrFile	Sets the name of the TDR file.

Examples

Save a TDR file with k -space models from the nonlocal line named NL1:

```
SaveK tdrFile=kspace.tdr nonlocal=NL1 \
      models=[list Delta1_1_Dispersion Delta2_0_IMRT]
```

The dispersion for the subband with index 1 for the valley named Delta1 is saved along with the IMRT for the Delta2 valley, subband 0.

Solve

This command solves the Poisson and Schrödinger equations.

Note:

No built-in bias-ramping capability is supplied. You can use a Tcl `foreach` loop to perform a bias ramp.

Syntax

For a self-consistent solve where biases are specified:

```
Solve V(Contact1)=Double V(Contact2)=Double \
      [initial] [logFile=String] [saveVacuumPotToFile=String]
```

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

For a self-consistent solve where the initial solution of the vacuum potential is read from a text file:

```
Solve initialPotFile=String \
    [initial] [logFile=String] [saveVacuumPotToFile=String]
```

For a self-consistent solve where biases are read from a text file:

```
Solve biasFromPotFile=String \
    [initial] [logFile=String] [saveVacuumPotToFile=String]
```

For a triangular potential well:

```
Solve Esurface=Double [logFile=String] [Ninv=Double]
```

For a non-self-consistent solve, based on the potential read from a TDR file:

```
Solve potentialFile=String [offset=Double] [potentialModel=String]
```

Argument	Description
biasFromPotFile	Sets the name of the vacuum potential file from which the biases of contacts are read. The vacuum potential file is a text file with two columns: The first column is the spatial coordinate, and the second column is the vacuum potential.
Esurface	Sets the value of the vertical surface field (in V/cm) that is used to create a triangular potential well. Use positive values for NMOS devices, and use negative values for PMOS devices.
initial	Specify this option to use a charge-neutral approximation as the initial guess for solving the Poisson equation.
initialPotFile	Sets the file name for the initial guess of the vacuum potential. The vacuum potential file is a text file with two columns: The first column is the spatial coordinate, and the second column is the vacuum potential.
logFile	Sets the name of the bias log file to write results to after the solve is completed. If not specified, then the currently open bias log file is used.
Ninv	Sets the target value of the total inversion density used in conjunction with Esurface. The Fermi level is adjusted automatically to meet this target value, in units of cm ⁻² .
offset	Sets the constant value (in V) by which to shift the specified potentialModel when mapping to the internal VacuumPotential. Default: 0 V
potentialFile	Sets the name of the TDR file from which to read the potential for a non-self-consistent solve.

Chapter 14: Sentaurus Band Structure/Tcl Commands

Subband and Mobility Calculations

Argument	Description
potentialModel	Sets the name of the potential model to read from the specified potentialFile for a non-self-consistent solve. Default: VacuumPotential
saveVacuumPotToFile	Sets the file name to which the vacuum potential after each iteration step is saved. The vacuum potential file is a text file with two columns: The first column is the spatial coordinate, and the second column is the vacuum potential.
V(Contact)	Sets the bias to apply to the named contact.

Examples

Solve the Poisson equation at a gate bias of 1.0 V and substrate bias of 0.0 V:

```
Solve V(Gate)=1.0 V(Substrate)=0.0 initial logfile=mydata.plt
```

A charge-neutral approximation is used as the initial guess, and the resulting data after convergence is written to the bias log file mydata.plt.

Solve the Poisson equation where the biases are read from the mypot.xy text file:

```
Solve biasFromPotFile=mypot.xy initial logfile=mydata.plt
```

A charge-neutral approximation is used as the initial guess, and the resulting data after convergence is written to the bias log file mydata.plt.

Solve self-consistently the Schrödinger equation–Poisson equation system where the initial guess for the vacuum potential is read from the text file mypot.xy, and the vacuum potential after each iteration step is overwritten to the same mypot.xy file:

```
Solve initialPotFile=mypot.xy saveVacuumPotToFile=mypot.xy
```

Using this command, the intermediate solution of the vacuum potential is always saved, regardless of whether the simulation is stopped either accidentally or intentionally. Therefore, when the simulation is restarted, the unnecessary repetition of previous iteration steps is avoided, and the self-consistent process continues with the most-updated vacuum potential. Consequently, the total CPU time consumption is reduced.

Solve any specified models, including Schrödinger solvers, non-self-consistently with the Poisson equation:

```
Solve potentialFile=myfile.tdr potentialModel=ElectrostaticPotential \
offset=0.25
```

In this example, the ElectrostaticPotential field from the myfile.tdr file is read into the tool, it is shifted by a constant offset of 0.25 V and is interpolated onto the

VacuumPotential model of the internal device mesh. Then, this VacuumPotential is used unmodified when solving any specified Schrödinger solvers or computing other models.

References

- [1] J. K. Ousterhout, *Tcl and the Tk Toolkit*, Reading, Massachusetts: Addison-Wesley, 1994.
- [2] M. M. Rieger and P. Vogl, “Electronic band parameters in strained $\text{Si}_{1-x}\text{Ge}_x$ alloys on $\text{Si}_{1-y}\text{Ge}_y$ substrates,” *Physical Review B*, vol. 48, no. 19, pp. 14276–14287, 1993.
- [3] V. Sverdlov *et al.*, “Effects of Shear Strain on the Conduction Band in Silicon: An Efficient Two-Band $k \cdot p$ Theory,” in *Proceedings of the 37th European Solid-State Device Research Conference (ESSDERC)*, Munich, Germany, pp. 386–389, September 2007.