Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Integrated Systems Laboratory

Mathieu Luisier

OMEN

Manual

October 2012

Contents

Chapter 1

Introduction

1.1 Simulator Purpose

The development of new transistors has always been supported by computer aided design (CAD) tools that can predict the device characteristics before their fabrication. As the gate length of MOSFETs was longer than 100nm and their cross section larger than 15nm, the classical drift-diffusion (DD) approach [1, 2, 3] was accurate enough to give good predictions about transistor performances. The major concern about the DD model is that it does not capture energy quantization, quantum mechanical tunneling, the wave nature of electrons and holes, and the atomistic granularity of the devices, which are all essential at the nanometer scale. Hence, it is becoming critical to replace DD by more advanced simulation approaches that will facilitate the discovery and the emergence of novel nanoelectronic devices.

A direct and self-consistent solution of the single-electron Schrödinger equation with open boundary conditions is more accurate than DD and fulfills the quantum mechanical requirement, but demands more computational power. However, the continuous increase of the CPU performances in the recent years represents a fantastic opportunity to re-think transistor simulation at the nanometer scale and go beyond standard approaches. In this context, we have developed OMEN, a next generation, multi-dimensional CAD tool based on quantum mechanical concepts and dedicated to the simulation of nanoelectronic devices [4, 5, 6, 7, 8]. Due to multiple parallelization levels, it can benefit from the largest available supercomputers to investigate electrical and thermal transport through nanostructures with an atomistic resolution.

The implementation of OMEN started in 2005 at the ETH Zürich, continued from 2008 to 2011 at Purdue University, and it now again pursued at the ETH Zürich. Since 2005, it has been used to study electron and phonon transport in a broad range of device structures, some of them being shown in Fig. 1.1. Its usefulness has been demonstrated in various applications such as studying the transport properties and the gate length scaling behavior of ideal and rough Si triple-gate nanowires with different crystal orientations [9, 10, 11], comparing the

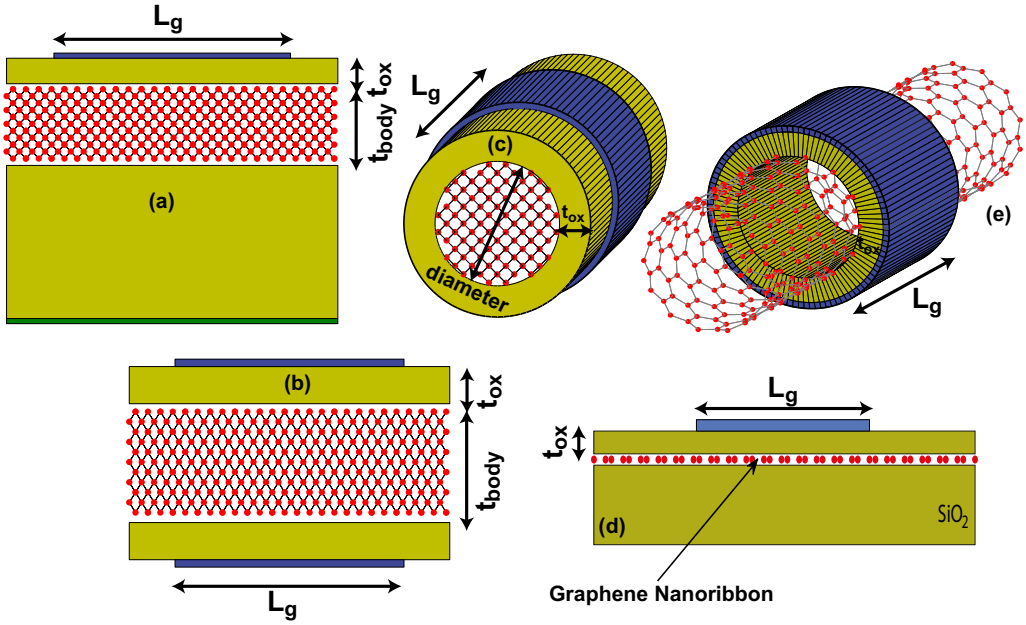


Figure 1.1: Schematic view of device structures that OMEN can simulate: (a) Single-Gate Ultra-Thin-Body (SG UTB), (b) Double-Gate Ultra-Thin-Body (DG UTB), (c) Gate-All-Around Nanowire (GAA NW), (d) Single-Gate Graphene Nanoribbon (SG GNR), and (e) Coaxially-Gated Carbon Nanotube (CG CNT). The UTB and NW transistors can be made of Si, Ge, or III-V semiconductors. The parameter L_g refers to the gate length, t_{body} to the body thickness, and t_{ox} to the oxide layer thickness.

output characteristics of n- and p-doped Si double-gate ultra-thin-body FETs with different transport and confinement directions [12], reproducing the experimental data of existing InGaAs high-electron mobility transistors [13, 14], investigating line-edge roughness in graphene nanoribbons [15, 16], analyzing and optimizing the performances of different types of band-to-band tunneling transistors (TFETs) [17, 18, 19, 20], understanding the effect of electron-phonon scattering in nanowires and extracting their phonon-limited mobilities [21, 22, 23], or investigating ballistic thermal transport through rough Si nanowires [24].

OMEN has been designed to accomplish three main tasks,

- calculating the electron/phonon bandstructure of bulk as well as 1-D, 2-D, and 3-D nanostructures with any crystal and/or confinement direction,
- computing the electron/phonon transmission probability through infinitely long [1] 1-D, 2-D, and 3-D nanostructures and the corresponding density-of-states (DOS),

[1] infinite means here that the simulation domain and the contacts are identical and have the same structure and properties

- calculating the self-consistent electronic and thermal current flowing through 1-D, 2-D, and 3-D nanostructures, either in the ballistic limit of transport or including interface roughness, electron-phonon, or phonon-phonon scattering.

1.1.1 Formulation of the quantum transport problem

OMEN solves multi-dimensional Schrödinger equations with open boundary conditions (OBCs) in the Non-Equilibrium Green's Function (NEGF) or Wave Function (WF) formalism. It is based on different flavors of the nearest-neighbor tight-binding model [25] (single s-orbital, p_z, sp^3, sp^3s^*, or $sp^3d^5s^*$, with or without spin-orbit coupling) for electrons and holes and on a modified valence-force-field method for phonons [26]. The carrier and current densities of the simulated nanostructures are obtained by self-consistently coupling the solution of the Schrödinger and Poisson equations till convergence is reached. This must be repeated for multiple bias points in order to obtain I-V (current vs. voltage) characteristics.

The core operation in OMEN is to solve the Schrödinger equation for each electron/hole/ phonon energy E and momentum k. In bulk-like structures (1-D), transport occurs along the x direction (convention) and the y and z axis are assumed periodic so that the momentum vector k=(k_y, k_z) is two-dimensional. In quantum wells or ultra-thin-bodies (2-D), y is a direction of confinement along the body or quantum well thickness and z is assumed periodic. In this case, the momentum vector k reduces to a single component k=k_z. Finally, in nanowires (3-D), y and z are directions of confinement and k=0 (no momentum dependence).

Open boundary conditions are introduced to couple the simulation domain to its surrounding environment (semi-infinite contacts or reservoirs) and allow for electrons/holes/phonons to enter and exit a finite domain. One of the most popular ways of treating the Schrödinger equation with OBCs is the NEGF formalism [27], which consists in solving the following system of equations for electrons or holes

$$\left(\mathbf{E} - \mathbf{H}(k) - \mathbf{V} - \mathbf{\Sigma}^{RB}(E,k) - \mathbf{\Sigma}^{RS}(E,k)\right) \cdot \mathbf{G}^{R}(E,k) \;=\; \mathbf{I} \tag{1.1}$$

$$\mathbf{G}^{<}(E,k) \;=\; \mathbf{G}^{R}(E,k) \cdot \left(\mathbf{\Sigma}^{<B}(E,k) + \mathbf{\Sigma}^{<S}(E,k)\right) \cdot \mathbf{G}^{A}(E,k). \tag{1.2}$$

In Eq. (1.1) and (1.2), the unknowns are the retarded $\mathbf{G}^{R}(E,k)$ and lesser $\mathbf{G}^{<}(E,k)$ Green's functions at energy E and momentum k. They are matrices of size $N_A \times N_{orb}$ where N_A is the number of atoms composing the simulation domain and N_{orb} the number of orbitals describing the properties of each atoms. For example, N_{orb}=10 in the sp^3d^5s* tight-binding model without spin orbit coupling, 1 s orbital, 3 p orbitals, 1 excited s orbital called $s*$, and 5 d orbitals. If spin-orbit coupling is included, N_{orb} increases to 20.

The single elements of the retarded $G_{ij}^{R\sigma_1\sigma_2}(E,k)$ and lesser $G_{ij}^{<\sigma_1\sigma_2}(E,k)$ Green's Functions represent the coupling or correlation between two orbitals σ_1 and σ_2 situated on two atoms i and j. The diagonal matrix $\mathbf{E}$ contains the electron/hole energy E, $\mathbf{I}$ is the identity matrix, $\mathbf{H}(k)$ refers to the device Hamiltonian matrix, which includes the orbital on-site energies as well as the coupling elements between

different orbitals and atoms ($H_{ij}^{\sigma_1\sigma_2}$). The entries of the diagonal matrix $\mathbf{V}$ are the electrostatic potential at each atomic site. The open boundary conditions are cast into the matrices $\mathbf{\Sigma}^{RB}(E,k)$ and $\mathbf{\Sigma}^{<B}(E,k)$. Finally, the scattering mechanisms such as electron-phonon are described by the scattering self-energies $\mathbf{\Sigma}^{RS}(E,k)$ and $\mathbf{\Sigma}^{<S}(E,k)$. Note that $G_{ij}^{A\sigma_1\sigma_2}(E,k)$=conj($G_{ji}^{R\sigma_2\sigma_1}(E,k)$) and that a greater Green's Function $\mathbf{G}^{>}(E,k)$ exists, defined as in Eq. (1.2), except that $<$ is replaced by $>$.

For phonons, Eq. (1.1) and (1.2) are slightly modified and take the following form

$$\left(\omega^2 - \mathbf{\Phi}(q) - \mathbf{\Pi}^{RB}(\omega,q) - \mathbf{\Pi}^{RS}(\omega,q)\right) \cdot \mathbf{D}^{R}(\omega,q) \;=\; \mathbf{I} \tag{1.3}$$

$$\mathbf{D}^{<}(\omega,q) \;=\; \mathbf{D}^{R}(\omega,q) \cdot \left(\mathbf{\Pi}^{<B}(\omega,q) + \mathbf{\Pi}^{<S}(\omega,q)\right) \cdot \mathbf{D}^{A}(\omega,q). \tag{1.4}$$

The main differences with Eq. (1.1) and (1.2) comes from the notation of the phonon Green's Functions, $\mathbf{D}^{<,>,R,A}(\omega,q)$ instead of $\mathbf{G}^{<,>,R,A}(\omega,k)$ and of the self-energies $\mathbf{\Pi}^{<,>,R,A}(\omega,q)$. The electron/hole energy E is replaced by ω^2, the square of the phonon frequency, the phonon momentum is denoted by q instead of k, and the Hamiltonian matrix $\mathbf{H}(k)$ becomes a dynamical matrix $\mathbf{\Phi}(q)$ of size $3{\times}N_A$. The valence-force-field method is employed to construct $\mathbf{\Phi}(q)$.

Equations (1.1)-(1.4), which have the form "A·B=C", must be solved for each possible energy E or frequency ω and momentum k (q). This can be done with a recursive Green's Function (RGF) algorithm instead of directly inverting or multiplying large matrices [28]. Although much more efficient than brute-force solutions, the RGF algorithm still induces high computational costs and limits the size of the device structure that can be treated. Furthermore, the presence of dissipative scattering mechanisms, cast into the self-energies $\mathbf{\Sigma}^{R,>,<,S}(E,k)$ and $\mathbf{\Pi}^{R,>,<,S}(\omega,q)$, couple different energies or frequencies and momentum altogether, adding another level of complexity to the problem.

Ballistic simulations represent a special case of Eq. (1.1)-(1.4) since all the scattering self-energies disappear and the equation for each energy and momentum can be treated independently from the others. Beside that simplification, the NEGF formalism can be replaced by a Wave Function approach where a single system of equations must be solved

$$\left(\mathbf{E} - \mathbf{H}(k) - \mathbf{\Sigma}^{RB}(k,E)\right) \cdot \mathbf{C}(k,E) \;=\; \mathbf{Inj}(k,E) \tag{1.5}$$

for electrons and holes. A similar equation can be derived for phonons. The multiple right-hand-side vector $\mathbf{Inj}(k,E)$ contains all the electronic states that are injected into the device structure from the contact regions. The advantage of Eq. (1.5) over Eq. (1.1) to (1.4) resides its form, a sparse linear system of equations "Ax=b", where the unknown is the coefficient vector $\mathbf{C}(k,E)$. Equation (1.5) can therefore be solved more efficiently using direct sparse linear solvers such as Umfpack [29], MUMPS [30], SuperLU$_{dist}$ [31], Pardiso [32], or a home-made block cyclic reduction (BCR) of the matrix "A" [33]. The Wave Function formalism thus allows for rapid simulations of large nanostructures, but only works in ballistic cases.

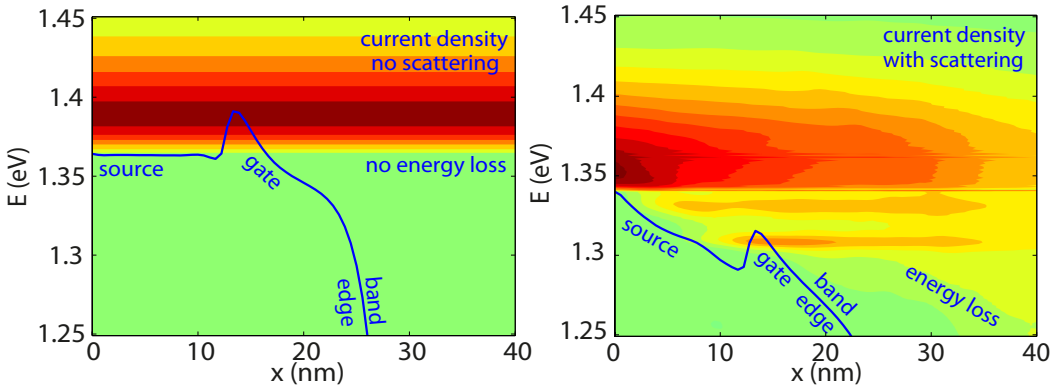


Figure 1.2: Spectral distribution of the electron current flowing through a nanowire transistor. (left) Ballistic limit of transport. (right) With electron-phonon scattering. Red indicates a high current concentration, green no current. The blue lines refer to the conduction band edge of these transistors.

1.1.2 Electron-phonon and phonon-phonon scattering

Very short devices do not necessarily operate close to their ballistic limit of transport (no interaction with their environment). Historically, it has been observed that Si transistors, irrespective of their gate length, operate at about 50% or less of their theoretical limit. Other semiconductor materials such as InGaAs can reach higher values (80-90%), but never 100%. It is therefore mandatory to include the interactions of electrons, holes, and phonons with each other and with their environment into the modeling effort in order to produce a tool capable of predicting the performance of nanoelectronic devices. Especially, the dissipative electron/hole-phonon and anharmonic phonon-phonon scattering mechanisms play an essential role at the nanoscale and should not be neglected. They have been therefore implemented in OMEN, making it one of the most advanced device simulators.

Electron-phonon scattering is expected to drastically reduce the ON-current of ultra-scaled transistors due to an effect known as backscattering [34]. By absorbing or emitting phonons, electrons are reflected back to their origin instead of propagating along one single direction. Electron-phonon scattering also modifies the spectral distribution of the current within a device structure and affects the electrostatic potential, as indicated in Fig. 1.2: while moving from the source to the drain contact of a transistor, electrons may loose energy by emitting phonons. Hence, in the presence of dissipative scattering, they enter and leave the device structure at a different energy. In the ballistic limit, each electron keeps the same energy and momentum from the beginning of its trajectory in the source contact till the end in the drain contact, on the opposite side of the device.

In the NEGF formalism the scattering self-energies $\mathbf{\Sigma}^{\gtrless S}(E,k)$ and $\mathbf{\Sigma}^{RS}(E,k)$

have the following form to account for electron-phonon scattering [21, 35]

$$\begin{aligned}\mathbf{\Sigma}^{\gtrless S}_{nn}(E,k) &= \sum_{l,i,j}\sum_{q,\omega_{ph}} \mathcal{V}^{ij}_{nlln}(\omega_{ph},q)\cdot\nabla_i\mathbf{H}_{nl}\cdot\Big(n^{ph}(\omega_{ph})\cdot\mathbf{G}^{\gtrless}_{ll}(E\pm\hbar\omega_{ph},k-q)+ \\ &\qquad (n^{ph}(\omega_{ph})+1)\cdot\mathbf{G}^{\gtrless}_{ll}(E\mp\hbar\omega_{ph},k-q)\Big)\cdot\nabla_j\mathbf{H}_{ln}, \\ \mathbf{\Sigma}^{RS}_{nn}(E,k) &\approx \frac{1}{2}(\mathbf{\Sigma}^{>S}_{nn}(E,k)-\mathbf{\Sigma}^{<S}_{nn}(E,k)). \end{aligned} \tag{1.6}$$

All $\mathbf{\Sigma}$'s are block diagonal matrices, where each block $\mathbf{\Sigma}^{S}_{nn}(E,k)$ is of size $N_{orb}\times N_{orb}$. In Eq. (1.6), ω_{ph} is the confined phonon frequency, $n^{ph}(\omega_{ph})$ the Bose distribution of equilibrium phonons with frequency ω_{ph} and momentum q. The quantity $\nabla_i\mathbf{H}_{nm}$ represents the derivative of the nearest-neighbor coupling matrix $\mathbf{H}_{nm}$ with respect to the coordinate i=x, y, or z along the bond $\mathbf{R}_m-\mathbf{R}_n$ connecting the atoms n and m. To simplify the calculation, all the self-energies are assumed local in position, i. e. only the diagonal $\mathbf{\Sigma}^{S}_{nn}(E,k)$ elements are considered [21]. The form factor $\mathcal{V}^{ij}_{nmmn}(\omega_{ph},q)$ contains information about the phonon modes [35].

It becomes clear from Eq. (1.6) that one energy-momentum (E,k) pair is connected to many other $(E-E',k-q)$. If the computation of the energy and momentum points is parallelized, as shown later, the required exchange of information between CPUs storing different (E',k') configurations makes the solution of Eq. (1.6) more tedious from a numerical point of view. Since also the expression for the scattering self-energy $\mathbf{\Sigma}^{\gtrless S}(E,k)$ in Eq. (1.6) depends on the Green's Function $\mathbf{G}^{\gtrless}(E-E',k-q)$ in Eq. (1.2), these two equations must be iteratively solved till convergence is achieved. This procedure is known as self-consistent Born approximation. It requires $N_{SC,iter}$ iterations to convergence, where $N_{SC,iter}$ might vary from 5 to 50 and is not known at the beginning of a simulation.

As electron-phonon scattering affects electronic transport, anharmonic phonon-phonon scattering has a huge influence on thermal transport processes: it strongly reduces the thermal conductivity of materials for temperatures about 100 K. In the framework of NEGF, anharmonic phonon-phonon scattering can be formulated with the help of scattering self-energies $\mathbf{\Pi}^{\gtrless S}(\omega,q)$ and $\mathbf{\Pi}^{RS}(\omega,q)$ defined as

$$\begin{aligned}\mathbf{\Pi}^{\gtrless S}_{nn}(\omega,q) &= 2i\hbar\sum_{lm}\sum_{q'}\int_{-\infty}^{\infty}\frac{d\omega'}{2\pi}dV^{(3)}_{nlm}(q')dV^{(3)}_{lmn}(q')\mathbf{D}^{\gtrless}_{ll}(\omega+\omega',q+q')\mathbf{D}^{\lessgtr}_{mm}(\omega',q'), \\ \mathbf{\Pi}^{RS}_{nn}(\omega,q) &\approx \frac{1}{2}(\mathbf{\Pi}^{>S}_{nn}(\omega,q)-\mathbf{\Pi}^{<S}_{nn}(\omega,q)). \end{aligned} \tag{1.7}$$

As for electron-phonon scattering, all the $\mathbf{\Pi}$ are block diagonal matrices. Each block $\mathbf{\Pi}^{S}_{nn}(\omega,q)$ is of size 3×3. In Eq. (1.7), the term $dV^{(3)}_{lmn}$ refers to the third derivative of the anharmonic potential energy V^{anh} as function of the atom positions $\mathbf{R}_l$, $\mathbf{R}_m$, and $\mathbf{R}_n$ [26]. For phonons too, one frequency-momentum pair (ω,q) is connected to many other $(\omega+\omega',q+q')$, making the parallelization of these quantities much more complicated than in ballistic simulations. Also, Eq. (1.7) depends on Eq. (1.3) and (1.4) so that both systems of equations must be solved self-consistently. This iterative process usually converges for $10\leq N_{SC,iter}\leq 100$.

1.1.3 Carrier and current densities

Once that Eq. (1.1) and (1.2) have been solved for each electron/hole energy E and momentum k, the charge and current densities of the considered systems can be calculated in the NEGF formalism according to the following equations

$$n(\mathbf{r}) = -i \sum_j \sum_k \int \frac{\mathrm{d}E}{2\pi} \mathrm{tr}\left(\mathbf{G}^<_{jj}(E,k)\right) \delta(\mathbf{r}-\mathbf{R_j}) \tag{1.8}$$

and

$$J_{i\rightarrow i+1} = \frac{e}{2\hbar} \sum_{\substack{i_1\in i \\ i_2\in i+1}} \sum_k \int \frac{\mathrm{d}E}{2\pi} \mathrm{tr}\left(\mathbf{H}_{i_1 i_2}\cdot\mathbf{G}^<_{i_2 i_1}(E,k) - \mathbf{G}^<_{i_1 i_2}(E,k)\cdot\mathbf{H}_{i_2 i_1}\right). \tag{1.9}$$

The trace operator tr runs over all the orbitals of an atom situated at $\mathbf{r} = \mathbf{R}_j$ in Eq. (1.8) and over the orbitals of all the atoms situated in the plane $x{=}x_i$ in Eq. (1.9). Hence, $J_{i\rightarrow i+1}$ represents the electron current flowing between the atomic plane $x{=}x_i$ and x_{i+1}.

The charge density $n(\mathbf{r})$ in Eq. (1.8) is then used to compute the electrostatic potential $V(\mathbf{r})$ that forms the diagonal entries of the matrix $\mathbf{V}$ in Eq. (1.1). This is achieved through Poisson equation that is self-consistently solved with Eq. (1.1), (1.2) and (1.8). This additional loop between $n(\mathbf{r})$ and $V(\mathbf{r})$ typically converges after $N_{Poiss,iter}{=}3$ to 10 iterations.

For phonons, the quantity of interest is the thermal current that flows through nanostructures whose contacts exhibit different temperatures. It is defined as

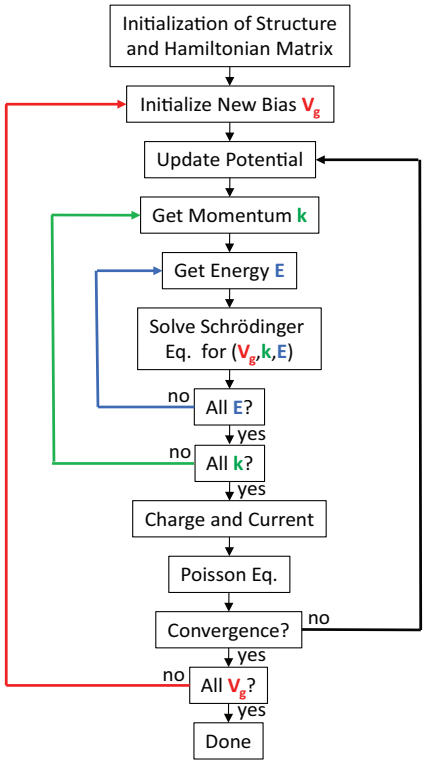
$$J_{ph,i->i+1} = \sum_{\substack{i_1\in i \\ i_2\in i+1}} \sum_q \int_0^\infty \frac{d\omega}{2\pi} \hbar\omega \ \mathrm{tr}\left(\mathbf{\Phi}_{i_1 i_2}\cdot\mathbf{D}^<_{i_2 i_1}(\omega,q) - \mathbf{D}^<_{i_1 i_2}(\omega,q)\cdot\mathbf{\Phi}_{i_2 i_1}\right). \tag{1.10}$$

In Eq. (1.10), the trace operator runs over the 3 degrees of freedom along which atoms can oscillate, x, y, and z. The phonon density, contrary to the electron density, does not play an important role since there is no Poisson equation to solve here. Hence, it is not given. The overall flow chart of OMEN is summarized in Fig. 1.3, both as a picture and as an algorithm.

1.2 Parallelization Scheme

Depending on the size of the simulated structure the simulation time on a single processor may become problematic. Hence, OMEN uses the message passing interface (MPI) [36] to parallelize the different tasks. For example, if N_k wavevector points are required to compute the bandstructure of an infinite wire and N CPUs are available the amount of k points calculated per CPU will be N_k/N. In a similar way, if the transmission and the DOS of a nanowire need to be evaluated at N_E different energy points, each processor will treat N_E/N points. In both cases a speed up factor of N can be reached.

More generally, the distribution of the tasks in OMEN is crucial to reduce the computational burden. As shown in Fig. 1.3, four natural levels of parallelism can



1: Construction of the atomistic simulation domain
2: Generation of the tight-binding Hamiltonian matrix H
3: Initialization of the FEM Poisson environment
4: **for** every bias point V_{gs}/V_{ds} (in parallel) **do**
5: Get source and drain Fermi levels
6: Get initial guess for electrostatic potential V
7: **repeat**
8: Update electrostatic potential $V \rightarrow H$
9: **for** every momentum point k_z (in parallel) **do**
10: Update momentum $H \rightarrow H(k)$
11: Compute contact bandstructure
12: Generate energy grid $E(k)$
13: Balance work load through CPUs
14: **for** every energy point E (in parallel) **do**
15: Compute open boundary Σ and S
16: Solve (in parallel) Schrödinger Eq.
17: Extract DOS and TE
18: **end for**
19: **end for**
20: Compute charge ρ and current J_d
21: Solve FEM Poisson equation $\rho \rightarrow V$
22: **until** self-consistent convergence of ρ and V
23: **end for**

Figure 1.3: (left) OMEN simulation flow chart. The loops on the left-hand-side (bias points, momentum, and energy) are parallelized while the loop on the right-hand-side (Poisson) cannot. (right) OMEN Algorithm with four levels of parallelization (lines 4, 9, 14, and 16).

be identified, (i) the bias points, (ii) momentum points, (iii) energy points, and (iv) a 1-D spatial domain decomposition as depicted in Fig. 1.4. All these parallelization levels have been implemented via MPI, but a mixed shared-distributed memory scheme is possible at the lowest parallelization level (domain decomposition). In ballistic transport simulations, the three highest loops can be embarrassingly parallelized, contrary to the fourth one, the spatial domain decomposition. When electron-phonon or phonon-phonon scattering is turned on, the coupling between different energies and momentum as induced by Eq. (1.6) and (1.7) makes the situation more complicated, the loop over momentum and energy in Fig. 1.3 can no more be embarrassingly parallelized, and a new parallelization scheme had to be developed [35].

The number of cores attributed to each parallelization level is selected by the user at the beginning of the simulation. Starting from a total of P_0 cores, N_{V_g} bias points are simultaneously treated on $P_{V_g}=P_0/N_{V_g}$ cores, where P_{V_g} is an input parameter. The number of momentum points per bias point N_k is chosen by the user, but the number of cores dedicated to each k group depends on the number of energy points $N_E(k)$ and is dynamically allocated by OMEN. An optimal balance of

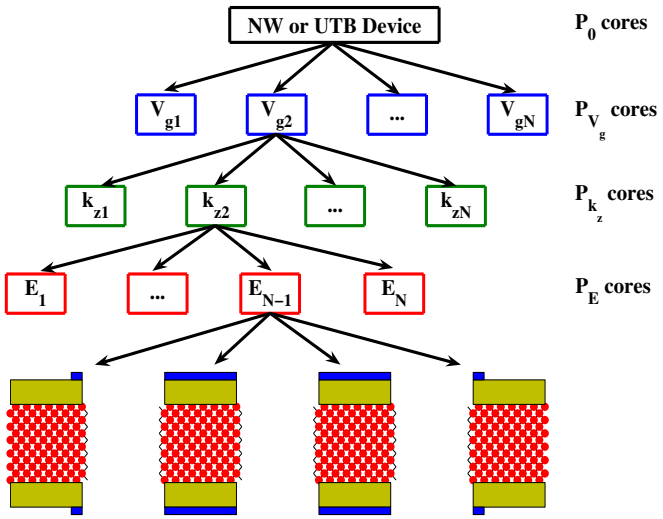


Figure 1.4: Parallel scheme of OMEN. The bias points, the momentum point, and the energy points are parallelized and form the three highest levels of parallelism. From the total number of cores P_0 there are P_{V_g} cores operating per bias point, P_{k_z} per momentum point, and P_E per energy point. If $P_E > 1$ (here 4) the simulation domain is spatially decomposed (fourth level of parallelism). In the ballistic limit of transport, the three highest levels form embarrassingly-parallelized problems, while the energies and momenta are coupled when electron-phonon or phonon-phonon scattering is included.

the work load results from this process [37]. At the lowest parallel level, the spatial domain decomposition is accomplished on P_E cores.

The computational performances and parallelization scheme of OMEN have been tested up to 59,904 cores on the SUN constellation star Ranger at TACC, up to 110,700 cores on the CRAY-XT5 Kraken at NICS, and up to 221,400 cores on the CRAY-XT5 Jaguar at Oak Ridge National Laboratory (ORNL). OMEN was probably one of the first engineering applications reaching a *sustained* petascale performance (1.44 PFlop/s on 221,400 cores on Jaguar at ORNL) [38]. The resulting paper won an honorable mention at the ACM Gordon Bell Prize competition held during the Supercomputing SC11 conference in Seattle, USA.

1.3 Compiling the Code

OMEN relies on several external packages that need to be compiled before it and then linked to it: AMD 2.0 [39], Arpack [40], Aztec 2.1 [41], Blas [40], Lapack [40], Metis [42], MUMPS 4.6.3 [43], Pardiso (not obligatory) [44], qhull 2003.1 [45], ScaLapack [40], SuperLU_*DIST* 2.0 [46], UFconfig 3.6.0 [47], and UMFPACK 5.0.1 [48]. The message passing interface (MPI) also needs to be installed. It is strongly recommended not to compile the Blas and Lapack libraries, but to use those pro-

vided by vendors (ACML, MKL, GotoBLAS, or ESSL). Significant speed-up can be obtained if high-performance numerical libraries are used.

The source code of OMEN is usually stored in a directory called "OMEN", which should be placed in a parent directory containing all the other required libraries. A file Makefile is available to compile each library, one after the other, terminating with OMEN. For that purpose, the user should define compiler options in a file make.inc placed in the parent directory where all the libraries and the source code of OMEN are stored. To summarize, three steps are necessary to compile OMEN

1. if not already available, downloading all the necessary solver libraries mentioned above (AMD, Arpack, Aztec, Blas,$\cdots$),

2. updating the file make.inc with compiler options and linking the Makefile of each library with this file,

3. entering the parent directory where the file Makefile is stored and typing "make all".

1.4 Starting a Job

The input of OMEN is a command file command.cmd including a list of parameters and tasks to perform, for example a bandstructure calculation or a self-consistent simulation. A job is launched with

$$mpiexec\ \text{-}n\ N\ \ OMEN_{platform}\ \ command.cmd$$

if N CPUs are available or

$$OMEN_{platform}\ \ command.cmd$$

on a single CPU. In both cases $platform$ is defined in the file make.inc and could be "amd64-ethz" or "XE6-pgi64" for example. In order that the parallel version works, the mpich2-1.0.4p1 package (or a later version) needs to be installed. More than one command file can be addressed to GreenSolver at the same time

$$mpiexec\ \text{-}n\ N\ \ OMEN_{platform}\ \ command1.cmd\ \ldots\ \ commandn.cmd$$

for the parallel case and

$$OMEN_{platform}\ \ command1.cmd\ \ldots\ \ commandn.cmd$$

for the sequential one.

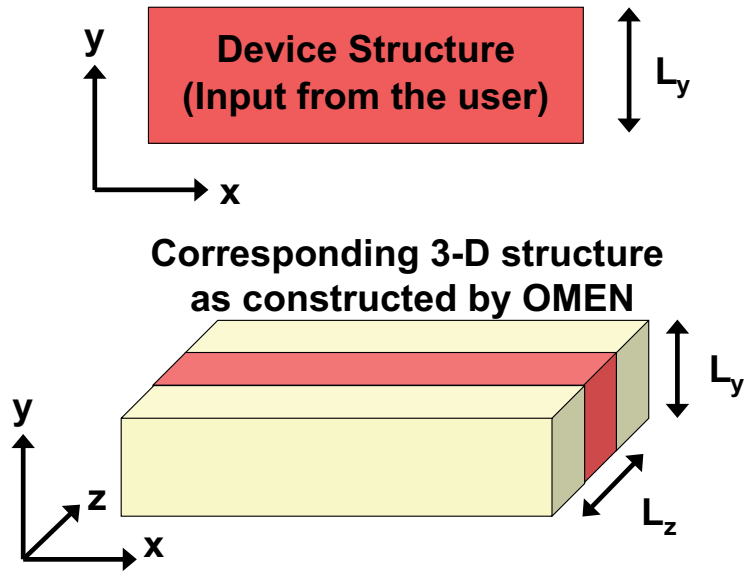


Figure 1.5: (Up) 2-D device structure as defined by a user in an input deck. Transport occurs along the x-axis (length L_x), y is a direction of confinement of with L_y, and z is assumed periodic. (Down) Corresponding 3-D simulation domain as constructed by OMEN. The period length along the z-axis, L_z, is used to construct a structure of size $L_x \times L_y \times 3L_z$.

1.5 Simulation Flow

Irrespective of the command type (bandstructure, transmission, or self-consistent), the first steps of the simulation flow remain the same

1. 1-D, 2-D, or 3-D nanostructures are created according to the specifications given in the command file. The primitive unit cell of the specified lattice type is shifted along the x, y, and z axis with the help of the displacement vectors v_1, v_2, and v_3. The position of the atoms within the unit cell as well as the three vectors v_1, v_2, and v_3 are adapted to the selected crystal orientation by a rotation. After each shift $n \times v_1 + m \times v_2 + l \times v_3$ the position of the unit cell atoms is checked: if it is within the simulation domain the atom is kept, otherwise it is neglected. Note that OMEN always constructs 3-D atomic structures, even for quantum well and bulk-like devices. However, along the directions assumed periodic, the thickness of the atomic structure corresponds to 3 times the period length, as illustrated in Fig. 1.5.

2. the Hamiltonian (electron) or dynamical matrix (phonons) corresponding to the atomic structure as well as the boundary matrices are generated. When the nanostructure is created, a nearest-neighbor table is written. It contains the x, y, and z positions of the atoms, their type, and their four nearest neighbors. The Hamiltonian/dynamical matrix constructor reads this table and replaces each connection by a $N_{orb} \times N_{orb}$ matrix, where N_{orb} is the number of orbitals per atom. If M CPUs work on the same energy point and the

nanowire contains N_A atoms, each processor receives just $(N_A \times N_{orb})/M$ rows of the total Hamiltonian/dynamical matrix. Hence, less memory is consumed.

After the initialization of the Hamiltonian/dynamical matrix the simulation flow goes on separated directions, depending on the task to accomplish. The format of the input and output variables corresponding to each possible task is explained in Chapter 2 and 3.

Chapter 2

Input Parameter Specifications

2.1 Introduction

To calculate the bandstructure, the transmission, and the density-of-states of an infinitely long nanostructure or to run a self-consistent simulation of a transistor or thermo-generator, the material, the tight-binding, and the structure parameters must be first selected. For example, OMEN should allow for a straightforward definition of the square and circular nanowire structures shown in Fig. 2.1. As a convention, electron/phonon transport occurs along the x-axis (which is assumed infinite in the bandstructure and transmission problems), while y and z are directions of confinement in nanowires, y is a direction of confinement and z is assumed periodic in quantum wells, and y and z are assumed periodic in 1-D structures.

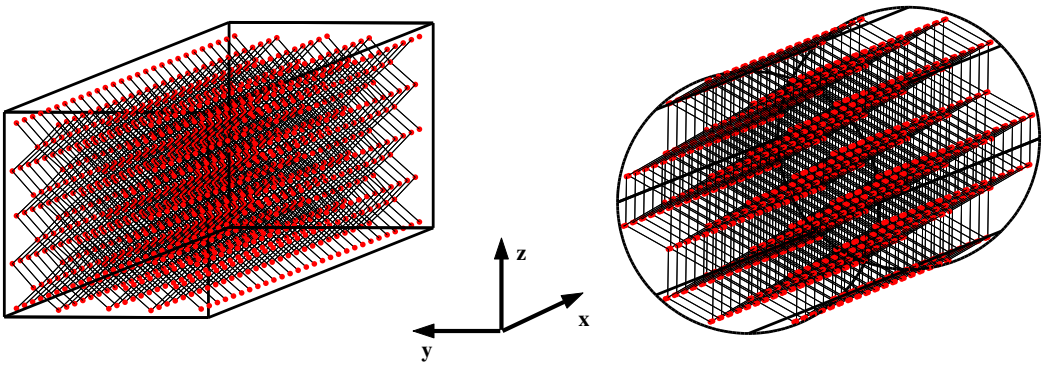


Figure 2.1: Schematic view of a square (left) and circular (right) nanowire. The transport direction is x, y and z are directions of confinement. Transport occurs along the <100> crystal axis for the square nanowire and along the <111> axis for the circular nanowire.

The different input parameters that exist in OMEN and that can be defined in command files are now summarized and classified in sub-groups. Parameters that cannot be found in this list are probably out-of-date or have been introduced after

the release of this manual.

2.2 General Parameters

This list contains parameters of general utility that are needed to construct the simulation domain and select the proper tight-binding/valence-force-field material parameters.

- *mat_name*: material composition of the nanostructure, possible values: Si (default), GaAs, GaAs_AlGaAs, Ge, SiGe, InAs_InGaAs_InAlAs, Si_P, Si_SiO2, Si_sp3ss_CB, Ge_Vogl_sp3ss, GaAs_Vogl_sp3ss, InAs_Vogl_sp3ss, InSb, InSb_sp3ss, InAs_GaSb_sp3ss, GaSb, GaAsSb_InP, AlN, GaN, InN, carbon_pz, cnt_pz, graphene_pzdxzdyz, graphene_pzdxzdyz_hydro_pz, graphene_py, graphene_sigma_pi, bilayer_pz, Bi2Te3, Si_new (no strain parameters), Si_InAs, CdS_CdSe, and ZnS_ZnSe. There is also a possibility to define a material in a file that OMEN can read. See the function Material.C for the file format.

- *mat_binary_x*: cation or anion concentration in ternary alloy semiconductors. For example, in the material InAs_InGaAs_InAlAs, the In concentration of InGaAs and of InAlAs can be specified. If the materials $In_{0.53}Ga_{0.47}As$ and $In_{0.52}Ga_{0.48}Al$ are desired, then *mat_binary_x* should be defined as [0.0 0.53 0.52], i. e. the first number (0.0) refers to the first material (InAs), for which nothing can be changed, the second one (0.53) to InGaAs, and the last one (0.52) to InAlAs.

- *lattice_type*: lattice type of the crystal structure. Possible values are: zincblende (default), wurtzite, graphene, rhombo(hedral), cnt, bilayer_graphene, and multilayer_graphene.

- *a0*: unstrained lattice constant of the material, value in [nm]. Default: 0.543. For wurtzite and rhombohedral crystals *a0* can be defined as [a0 c0] and as [a0 h] for bilayer graphene, where h is the distance between the 2 graphene layers.

- *transport_type*: type of transport (electron or phonon), 0 (default): electron, 1: phonon.

- *open_system*: boundary conditions along the x-axis, 0: closed boundary conditions, 1 (default): open boundary conditions.

- *first_atom*: first atom that is positioned at $(x,y,z) = (0,0,0)$, possible value: cation or anion.

- *NDim*: number of dimensions of the device structure, 3 (default) for 3-D (nanowire, quantum dots), 2 for 2-D (quantum well), 1 for 1-D, and 0 for bulk bandstructure.

- *Nky*: number of momentum points along k_y (periodicity along the y axis as in 1-D structures). Default: 1. The wave vector k_y extends from 0 to π for ballistic simulations and from -π to π when scattering is turned-on and the momentum points are coupled. If *Nky* is negative, then the k_y values stored in the file defined by *phiy_file* are taken.

- *phiy_file*: if *Nky*<0, then the k_y values stored in the file *phiy_file* are used. First column: k_y values (unit: phase, all the values must be comprised between -π and π). Second column: dk_y values as used to integrate over k_y. No units, i. e. $dk_y(n)$=($k_y(n+1)$-$k_y(n-1)$)/2π.

- *Nkz*: number of momentum points along k_z (periodicity along the z-axis as in 1-D and 2-D structures). Works as *Nky*.

- *phiz_file*: same as *phiy_file*, but for k_z.

- *rot_sym*: not used

- *tb*: parameter to turn on (*tb*=20) and turn off (*tb*=10, default) spin-orbit coupling. A simulation with spin-orbit coupling is much slower since the size of the matrices are doubled and the contact eigenvalue problems become complex instead of real.

- *dsp3*: passivation energy for the dangling bonds as in Ref. [49], value in [eV].

- *h_passivation*: parameter to determine whether or not hydrogen atoms should be added to the surface of a nanostructure to passivate the dangling bonds. 0 (default): no hydrogen passivation, 1: hydrogen passivation. Works only with materials Si and graphene_pzdxzdyz_hydro_pz.

- *max_bond_def*: maximum relative bond deformation caused by strain, double precision value. It is used to find the nearest-neighbors of one atom, should not be changed except if the strain is larger than 5%.

- x: crystal axis aligned with the transport direction, value [n_x m_x l_x].

- y: first direction of confinement, value [n_y m_y l_y]. x and y must be orthogonal, i.e. $\overrightarrow{x} \cdot \overrightarrow{y}$=0.

- z: second direction of confinement, value [n_z m_z l_z]. z must be orthogonal to x and y, i.e. $\overrightarrow{z}$=$\overrightarrow{x} \times \overrightarrow{y}$.

- *update_atoms*: if 1, the atom coordinates and types of the device structure are read from the file *atom_file* defined in the input deck. The nearest-neighbor connection table remains the same as the one created before the atom coordinates were replaced by the values stored in a file. If 2, the nearest-neighbor connection table is also read from a file. Default: 0.

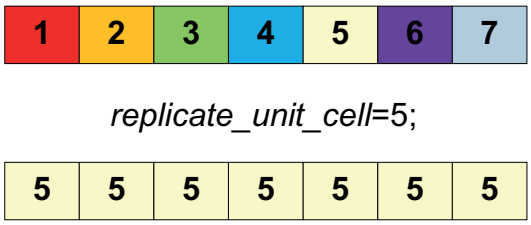


Figure 2.2: Illustration of the working principle of the input parameter *replicate_unit_cell*. (Up) Device structure composed of 7 different unit cells or slabs. (Down) With *replicate_unit_cell*=5, a new device structure composed of identical unit cells or slabs, all equal to the 5^{th} unit cell or the original structure, is created.

- *read_atom_pos*: if 1, the atom coordinates are read from the file *atom_file* defined in the input deck. A new nearest-neighbor connection table based on the atom coordinates stored in a file is generated.
- *atom_file*: file containing atomic coordinates and types that can be used to update the position of the atoms obtained from the structure generator of OMEN. It should contain at least 4 columns (3 atom coordinate and 1 type). It can also store additional columns containing the list of the nearest-neighbors of each atom.
- *replicate_unit_cell*: determines whether a new atomic structure should be constructed based on one specific unit cell of the existing one that will be replicated many times. The value of *replicate_unit_cell* is the index of the structure unit cell that should be replicated. Default: 0 (no unit cell replication).
- *QMregion*: if 1, indicates that a quantum mechanical region smaller than the device structure is defined. Default: 0.
- *QMstart*: x coordinate of the beginning of the quantum mechanical region if *QMregion*=1. Works only with the recursive Green's Function solver. Never successfully tested.
- *QMstop*: x coordinate of the end of the quantum mechanical region if *QMregion*=1. Works only with the recursive Green's Function solver. Never successfully tested.
- *no_mat*: number of elements that form the simulation domain (semiconductor+oxide+roughness parts), integer value. A simulated structure is always composed of *no_mat* cylindrical, spherical, and/or parallelepiped elements that are assembled together. Note that *no_mat* must be equal to the sum of *no_channel_mat*, *no_oxide_mat*, and *no_rough_mat*.
- *x0*: assistance variable, value in [nm]. It is recognized by the parser that reads the command file. Hence, it can be used in the structure definition instead

of writing a double precision value many times. A structure with the same shape but different dimensions can be easily constructed by just modifying pre-initialized variables. Its original goal is to set the smallest x coordinate of the device structure. The variables *y0* and *z0* also exist.

- *tc*: same as *x0*. It can for example describe the channel thickness. The variables *td* (drain thickness) and *ts* (source thickness) can also be used.
- *hc*: same as *x0*. It can be used to describe the channel height. Its counterparts *hd* and *hs* refer to the drain and source height, respectively.
- *yc*: same as *x0*. It can be used to set the y coordinate of the center point in a cylinder. For the z coordinate the variable *zc* is recognized, *rc* for the radius.
- *Lwq*: same as *x0*. It can be used to describe the width of a quantum well in a resonant tunneling structure.
- *Lb*: same as *x0*. It can be used to describe the width of a barrier in a resonant tunneling structure. If there are many barriers, the variables *Lb1* and *Lb2* are also available.
- *Lsp*: same as *x0*. It can be used to describe the length of a spacer in a transistor structure. If there are many spacers, the variables *Lsp1* and *Lsp2* are also available.
- *tox*: assistance variable, value in [nm]. It can be used to describe the thickness of lateral oxide layers while *hox* can play the same role for the height of oxide layers.
- *Lc*: channel length, value in [nm]. This is not an assistance variable and it must be set to a value.
- *Ls*: source length, value in [nm]. This is not an assistance variable and it must be set to a value.
- *Ld*: drain length, value in [nm]. This is not an assistance variable and it must be set to a value. The variables *Lc*, *Ls*, and *Ld* are needed to generate the initial guess for the electrostatic potential in self-consistent simulations. It is important that these values represent the real length of the source, the drain, and the channel. Otherwise, the initial solution will not be generated correctly.

2.3 Channel Structure

- *no_channel_mat*: number of elements that form the semiconducting channel of the considered nanostructures, integer value. Among the *no_mat* elements

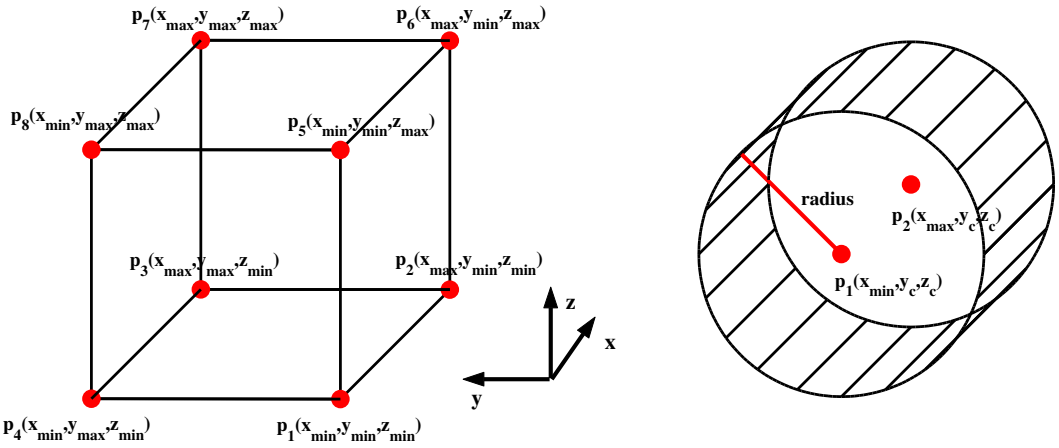


Figure 2.3: Definition of nanowires using 8 points (left) or 2 points and a radius (right). The first technique allows not only square cross section, but also rectangular, triangular, trapezoidal, and combination of these elements.

that form the simulation domain, *no_channel_mat* ones are made out of a semiconductor like Si, GaAs, or Ge and are used in the transport calculation (solution of the Schrödinger equation with open boundary conditions).

- *mat_type(n)*: type of material element, possible value: square, circle, or sphere. The index n goes from 1 to *no_channel_mat*. Each element that composes the semiconducting channel is either a parallelepiped, a cylinder, or a sphere.

- *mat_id(n)*: index characterizing the nature of the material n. For example, if the material InAs_InGaAs_InAlAs is selected, *mat_id(n)*=1 indicates that the n^{th} channel component is made of InAs. If *mat_id(n)*=2, the n^{th} channel component is made of InGaAs with an In concentration specified in *mat_binary_x*.

- *mat_cs(n)*: check if the element n determines the channel cross section, i.e. if it is directly connected to the semi-infinite contacts, possible value yes or no.

- *mat_coord(n,m)*: this variable stores the m^{th} coordinate of the n^{th} element composing the semiconductor channel. If *NDim*=3 (3-D nanostructure, nanowire or quantum dot), then the n^{th} element is either (i) a parallelepiped (defined by 8 points, m runs from 1 to 8, as shown in the left part of Fig. 2.3), (ii) a cylinder (defined by 2 points only, the center of the front and back disks, m runs from 1 to 2, as shown in the right part of Fig. 2.3), or (iii) a sphere (defined by one point only, the center of the sphere, m=1). In any case, *mat_coord(n,m)* = [x y z] where x, y, and z are the coordinates of the m^{th} point. If *NDim*=2 (2-D nanostructure, quantum well), then the n^{th} channel element must be a rectangle defined by 4 points (m runs from 1 to 4) and *mat_coord(n,m)* = [x y] where x and y are the coordinates of the m^{th} point. Finally, if *NDim*=1

(1-D bulk-like structure), the n^{th} channel element must be a line defined by 2 points (m runs from 1 to 2) and *mat_coord(n,m)* = x where x is the coordinate of the m^{th} point. Note that carbon nanotubes are defined as circular structures, but with *lattice_type*=cnt. Graphene nanoribbons are defined as 3-D structure with a height of 0.21 nm ($1.5a_0$).

- *mat_radius(n)*: radius of the front and back disks of the n^{th} cylindrical element, value in [nm]. To define an ellipsoid, *mat_radius(n)*=[a b] where a and b are the 2 axis of the ellipsoid.
- *Xi_wire*: semiconductor channel affinity, value in [eV].

2.4 Oxide Layer Structure

- *no_oxide_mat*: number of elements that form the oxide around the semiconducting channel, integer value. Among the *no_mat* elements that form the simulation domain, *no_oxide_mat* elements are made out of oxide and are used in the Poisson equation only. No solution of the Schrödinger equation is performed in the oxide layers. It is therefore assumed that no charge sits there.
- *ox_type(n)*: type of oxide element, possible values: square, circle, or sphere. The index n goes from 1 to *no_oxide_mat*. Otherwise same as *ox_type(n)*.
- *ox_id(n)*: index characterizing the nature of the oxide layer n. Must be used if there are different types of oxide layers in the device structure. The only parameter that characterize an oxide layer is their permittivity *Eps_ox*. See this parameter for more information.
- *ox_coord(n,m)*: coordinates of the oxide elements. Defined as *mat_coord(n,m)*.
- *ox_radius(n)*: radius of the front and back disks of the n^{th} cylindrical oxide element, value in [nm]. If an oxide layer surrounds the channel of a circular nanowire, then it radius should be defined *ox_radius=mat_radius+tox*, where *tox* is the thickness of the oxide layer. As pointed out earlier, OMEN moves atoms in the space and check whether they are inside one material or oxide region defined in the command file before keeping or discarding them. When an oxide layer is defined as a cylinder with a larger radius than the semiconducting channel, then a point that is inside the region defined as channel is also inside the region defined as oxide since they overlap. This might look like a conflict (what is the type of that atom?). In fact it is not since OMEN goes one region after the other when it checks whether an atom belongs to the device structure or not. The first time that OMEN finds a region the atoms belong too, it stops its search and does not look for a second eventual region. Hence, as long as the material region with a smaller radius is defined before the one with a larger radius, there will be no problem.

2.5 Gate Contact Structure

- *no_gate*: number of gates that control the potential barrier, integer value. For a triple-gate configuration, *no_gate*=3, for a double-gate configuration, *no_gate*=2, for a gate all-around, Ω, or single-gate configuration, *no_gate*=1. The same material parameters and voltages are used for all the gates.

- *gate_type(n)*: type of the gate, possible value: square, circle, or Omega. The index n goes from 1 to *no_gate*.

- *gate_coord(n,m)*: gate coordinates, same principle as *mat_coord(n,m)* except that m runs from 1 to 4 if *gate_type(n)*=square and *NDim*=3 (gate contact is a surface) and from 1 to 2 if *gate_type(n)*=square and *NDim*=2 (gate contact is a line).

- *gate_radius(n)*: radius of the all-around or Ω gate configuration, value in [nm].

- *gate_angle(n)*: minimum and maximum angle of the Ω gate, which does not completely cover circular nanowire. For example, if *gate_angle(n)*=[amin amax] (amin and amax in degrees), then the Ω gate covers the angular region going from amin to amax. The region from amax to amin it not covered by the gate contact.

- *phi_m*: work function of the gate metal, value in [eV].

2.6 Roughness Surfaces

- *roughness.on*: activate the presence of surface roughness if *roughness.on*=1, default: 0.

- *roughness.seed*: seed number for the random generator used to create rough surfaces. If not specified, OMEN will automatically generate a new seed number and print it at the beginning of the simulation.

- *roughness.form*: type of the rough surface, possible values: circle or square. Currently, only works with the option circle.

- *roughness.rms*: root mean square of the rough surface. An exponential autocorrelation function is assumed [50].

- *roughness.Lms*: correlation length of the rough surface.

- *no_rough_mat*: number of regions with a rough surfaces. Such regions should only be defined if *NDim*=3 and

- *roughness.form*=circle. For example, if a nanowire with a cylindrical shape and a radius r_0 has a rough surface, then, using *ro_type(n)*, *ro_coord(n,m)*,

and *ro_radius(n)* a cylinder with radius r_0+0.5 should be defined. This will automatically generate a rough surface to the inner cylinder with radius r_0. Note that the rough surface should start 3-5 nm after the beginning of the nanowire and stop 3-5 nm before its end. Without this security regions, OMEN will not be able to properly introduce open boundary conditions and/or self-consistent simulations will not converge.

- *ro_type(n)*: type of roughness element, possible value: square or circle. Currently, only the circle case works. The index n goes from 1 to *no_rough_mat*. Each element that composes the roughness domain is either a parallelepiped or a cylinder.
- *ro_coord(n,m)*: coordinates of the roughness elements. Defined as *mat_coord(n,m)*.
- *ro_radius(n)*: radius of the front and back disks of the n^{th} cylindrical roughness domain, value in [nm].

2.7 Strain Regions

- *relax_atoms*: 1 to relax the atom positions with a valence-force-field (VFF) solver, currently not used. Default: 0.
- *alpha*: harmonic bond stretching constant of the VFF model with Keating potential. Currently not used.
- *beta*: harmonic bond bending constant of the VFF model with Keating potential. Currently not used.
- *strain_bc*: not used.
- *strain_model*: parameter to switch between the 2 Boykin strain models. Possible values are 0: 2002 strain model [52] and 1 (default): 2010 strain model [51].
- *strain.on*: switch the strain on or off, possible value: 0 (default, no strain) or 1 (strain). The entire device structure gets strained with the atomic bonds deformed according to the parameters defined in *strain.Eps_xx*, *strain.Eps_xy*, *strain.Eps_xz*, $\cdots$.
- *strain.Eps_xx*: relative deformation of the atomic bonds along the x-direction, double precision value.
- *strain.Eps_xy*: off-diagonal element of the strain tensor.
- *strain.Eps_xz*: off-diagonal element of the strain tensor.
- *strain.Eps_yy*: relative deformation of the atomic bonds along the y-direction, double precision value.

- *strain.Eps_yz*: off-diagonal element of the strain tensor.
- *strain.Eps_zz*: relative deformation of the atomic bonds along the z-direction, double precision value.
- *no_strain_domain*: number of regions where a local strain tensor should be applied. The structure dimensions are automatically modified to account for the strain-induced deformations.
- *strain_domain(n).xmin*: minimum x coordinate of the n^{th} local strain region.
- *strain_domain(n).xmax*: maximum x coordinate of the n^{th} local strain region.
- *strain_domain(n).ymin*: minimum y coordinate of the n^{th} local strain region.
- *strain_domain(n).ymax*: maximum y coordinate of the n^{th} local strain region.
- *strain_domain(n).zmin*: minimum z coordinate of the n^{th} local strain region.
- *strain_domain(n).zmax*: maximum z coordinate of the n^{th} local strain region.
- *strain_domain(n).Eps_vec*: strain tensor of the n^{th} local strain region. It can be defined either as *strain_domain(n).Eps_vec*=[Eps_xx Eps_yy Eps_zz] in which case the off-diagonal elements of the tensor are assumed to be 0 or as *strain_domain(n).Eps_vec*=[Eps_xx Eps_yy Eps_zz Eps_xy Eps_xz Eps_yz].

2.8 Doping Regions

- *no_doping*: number of doping regions in the nanowire, integer value.
- *ND_S*: donor doping concentration in the source, value in [m^{-3}]. Must absolutely be defined since *ND_S*-*NA_S* will be used to determine the value of the left contact Fermi level.
- *NA_S*: acceptor doping concentration in the source, value in [m^{-3}]. Must absolutely be defined.
- *ND_D*: donor doping concentration in the drain, value in [m^{-3}]. Must absolutely be defined since *ND_D*-*NA_D* will be used to determine the value of the right contact Fermi level.
- *NA_D*: acceptor doping concentration in the drain, value in [m^{-3}]. Must absolutely be defined.
- *doping_type(n)*: type of the n^{th} doping region, possible value: square or circle. The index n goes from 1 to *no_doping*.

- *doping_ND(n)*: donor doping concentration in the n^{th} region, value in [m^{-3}]. The doping concentration is assumed to be homogeneously distributed over all the atoms present in the selected region.
- *doping_NA(n)*: acceptor doping concentration in the n^{th} region, value in [m^{-3}]. Same principle as for *doping_ND(n)*.
- *doping_coord(n,m)*: coordinates delimiting the n^{th} doping region. The index n goes from 1 to *no_doping*. Same principle as *mat_coord(n,m)*.
- *doping_radius(n)*: radius of the doping region if *doping_type(n)*=circle, value in [nm].
- *no_doping_domain*: number of doping regions with a Gaussian doping profile, i. e. with a doping profile defined according to

$$dop(x,y,z) = N_D\_N_A \cdot exp(-(x-x_{min})^2/\sigma_x^2 - (y-y_{min})^2/\sigma_y^2 - (z-z_{min})^2/\sigma_z^2)$$

  in the region delimited by x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, and z_{max}.
- *doping_domain(n).xmin*: x coordinate at which the doping profile starts to exponentially decay in the n^{th} doping region.
- *doping_domain(n).xmax*: x coordinate at which the doping profile stops to exponentially decay in the n^{th} doping region.
- *doping_domain(n).ymin*: y coordinate at which the doping profile starts to exponentially decay in the n^{th} doping region.
- *doping_domain(n).ymax*: y coordinate at which the doping profile stops to exponentially decay in the n^{th} doping region.
- *doping_domain(n).zmin*: z coordinate at which the doping profile starts to exponentially decay in the n^{th} doping region.
- *doping_domain(n).zmax*: z coordinate at which the doping profile stops to exponentially decay in the n^{th} doping region.
- *doping_domain(n).slope*: decay slope of the Gaussian doping profile in the n^{th} doping region, should be defined as *doping_domain(n).slope*=[σ_x σ_y σ_z].
- *doping_domain(n).ND_NA*: maximum value of the doping concentration in the n^{th} doping region before it starts decaying.
- *update_doping*: variable to determine whether the doping profile should be read from a file. Possible values: 0 (default, not read from a file) and 1 (read from file specified in doping_file).

- *doping_file*: file containing the doping concentration at each discretization point of the device structure. This file must have the same length as the matrix grid.dat as produced by the command Write_Grid_Matrix. Its entries must correspond to the grid point coordinates stored in grid.dat.

- *update_fermi*: determines whether the equilibrium Fermi level of the system under consideration should be computed in OMEN (default value 0) or defined in the command file (1).

- *fermi_level*: user-defined value of the equilibrium Fermi level of the device configuration.

2.9 Schottky Contact

- *Schottky.active*: if 1, the source and drain contacts become Schottky instead of ohmic contacts. Default: 0.

- *Schottky.type*: determines the type of the Schottky contact (0: n-type, 1: p-type). It must be defined as *Schottky.type*=0 (or 1) if both Schottky contacts are of the same type or as *Schottky.type*=[0/1 1/0] if they have different types.

- *Schottky.barrier*: barrier height of the Schottky contact at the metal/semiconductor interface. Same format as for *Schottky.type*. Unit: eV.

- *Schottky.virtual_CB*: virtual conduction band level of the metal contacts with respect to the semiconductor it is attached to. For example, if *Schottky.virtual_CB*=-0.5, OMEN shifts down the potential of the semi-infinite semiconductor reservoirs by 0.5 eV to mimic a metal. Same format as for *Schottky.type*. Unit: eV.

2.10 Bandstructure Options

- *bs_solver*: determines which eigenvalue solver should be used to compute the bandstructure of the semi-infinite reservoirs attached to the device structure. Possible values: sparse (Arpack) or full (Lapack).

- *n_of_modes*: number of modes (or bands) calculated for each wave vector point in the bandstructure problem, integer value.

- *Nk*: number of k_x points in the calculation of the reservoir bandstructure, integer value. A high number of wave vector points is important to find the states with zeros velocity. These special points are required in the construction of the energy vector.

- *last_first*: integer value, last wire unit cell is equal to the first one if *last_first*=1. Consequently, the open boundary conditions of one contact only are calculated. Does not work in the self-consistent case where (*last_first* automatically set to 0).

- *NxFold*: parameter to increase the size of the primitive unit cell of the nanostructure along the x-axis (transport direction). Integer value, default: 1. For example, in a nanowire with transport along the <100> axis, the primitive nanowire unit cell contains 4 atomic layers. If *NxFold*=2, then the size of the primitive unit cell along the x-axis is multiplied by 2 and it contains 8 instead of 4 atomic layers. Increasing *NxFold* automatically folds the bandstructure along the x-axis.

- *NyFold*: same as *NxFold*, but along the y axis. Only works if the y-direction is assumed periodic, i. e. if *NDim*=1.

- *NzFold*: same as *NxFold*, but along the z axis. Only works of the z direction is assumed periodic, i. e. if *NDim*=1 or 2.

2.11 Energy

- *dE_in*: distance between two energy points right after the presence of a state with zero-velocity in the contact bandstructure, value in [eV].

- *dE_f*: distance between two energy points in regions where no singularity are present, value in [eV].

- *Elimit*: energy interval that is refined after a state with zero-velocity is detected in the contact bandstructure, value in [eV]. In this interval the distance between two energy points growths from *dE_in* to *dE_f*.

- *Emin_tail*: energy interval that is considered below (above) the lowest (highest) conduction (valence) subband (taken from both the left and the right reservoir), value in [eV].

- *EOffset*: this is only defined for electrons. Unit: [eV]. In transmission calculations: total energy interval that is considered. The energy vector goes then from *Emin* which is the minimum energy of the system (automatically generated from the contact bandstructures and *Emin_tail*) to *Emax*=*Emin*+*EOffset*. In self-consistent simulations: energy interval that is added after the highest contact Fermi level (electrons) or below the lowest contact Fermi level (holes).

- *dE_sep*: energy distance between a state with zero-velocity in the contact bandstructure and the first upcoming discretization point, value in [eV]. The DOS of nanostructures exhibits significant peaks. To avoid the energy point E_{peak} where these peaks have their maximum (the matrices are ill-conditioned

there), no discretization point is present in the interval E_{peak}-*dE_sep* to E_{peak}+ *dE_sep*.

- *EExt*: this is only defined for phonons. Unit: [eV]. The energy vector for phonons extends from the minimum of the lowest phonon band up the maximum of the highest phonon band. An energy interval equal *EExt* can be added added above the largest phonon energy.
- *NEmax*: maximum number of calculated energy points, integer value. If more than *NEmax* points are included in the energy interval *Emin* to *Emax* the *NEmax* first points are considered. This variable only works for simulations where transmission and DOS are calculated, not in self-consistent simulations where the number of energy points is automatically determined by OMEN.
- *eta_res*: imaginary part of the energy in the reservoir (contacts), value [eV]. If this parameter is not equal to 0, in nanowires, the time necessary to compute the boundary conditions increases since a complex eigenvalue problem is solved instead of a real one. In any case, the value of *eta_res* should be smaller than 1e-6 (or even 1e-8). Otherwise propagating states get a too large imaginary part and are considered as decaying states.
- *eta*: imaginary part of the energy in the device, value [eV]. This parameter does not influence the results and should be set to 0 or to a value smaller than 1e-8, otherwise current continuity may be broken. The basis compression algorithm does not work if *tb*=20 and *eta*>0.
- *update_energy*: determines whether the electron or phonon energy vector should be calculated by OMEN (*update_energy*=0, default value) or whether it should be read from a file (*update_energy*=1).
- *energy_file*: variable to define a file containing the energy points that OMEN should use instead of creating its own. The first line of the file should contain the number of energy points to be read, the second line the energy points.
- *regular_mesh*: if 0 (default), then the electron and phonon energy vector is automatically generated by OMEN and might be inhomogeneous, depending on the values of *dE_in* and *dE_f*. If 1, the energy vector becomes homogeneous. OMEN extracts the minimum *Emin* and maximum *Emax* energy values and creates an energy vector defined as Emin:*dE_f*:Emax. Each momentum group has a different energy vector. If 2, then OMEN looks for the global minimum and maximum energy values across all the momentum groups and then create an homogeneous energy vector.

2.12 Poisson Equation

- *poisson_solver*: if 0, then the solution of Poisson equation is turned off. Default: 1.

- *poisson_criterion*: convergence criterion for the outer loop of Poisson equation. Default: 1e-3.
- *poisson_iteration*: maximum number of outer Poisson iterations per bias point. Default: 15.
- *max_proc_poisson*: maximum number of processors solving Poisson equation per bias point. Default: 64.
- *poisson_inner_criterion*: convergence criterion for the inner loop of Poisson equation. Default: 1e-3.
- *poisson_inner_iteration*: maximum number of inner Poisson iterations per outer Poisson iterations.
- *charge_average*: if 1, the charge density of each atom is replaced by the average of its charge density as well as that of all its nearest-neighbors. Default: 0.
- *charge_transfer*: not used.
- *grid_accuracy*: number of mesh points added along the lines that connect two nearest-neighbor atoms. Default: 2.
- *update_perm*: if 1, then the permittivity of the device structure is read from a file instead of using the definition of the command file. Default: 0.
- *perm_file*: file containing the permittivity at each discretization point of the device structure. This file must have the same length as the matrix grid.dat as produced by the command Write_Grid_Matrix. Its entries must correspond to the grid point coordinates stored in grid.dat.
- *update_fitness*: creating a stiffness matrix to solve Poisson equation on a finite element grid is sometimes computationally and memory intensive, especially the generation of the tetrahedron mesh. This operation cannot be parallelized. OMEN offers therefore the possibility to run a first simulation on a single CPU to create a stiffness matrix (command: Write_Fitness_Matrix). Then, in a second step, OMEN can load this stiffness matrix. If *update_fitness*=1, all the CPUs that solve Poisson equation load the entire stiffness matrix. If *update_fitness*=2, each CPU dealing with Poisson equation only loads a small part of the total stiffness matrix. Default: 0 (OMEN generates a stiffness matrix and directly use it).
- *fitness_file*: file containing the stiffness matrix. With the command Write_Fitness_Matrix OMEN produces such a file called FitMat_dat by default.
- *update_grid*: not used.
- *grid_file*: not used.

- *update_tetra*: OMEN uses the qhull [45] package to produce a tetrahedron mesh based on existing grid points. If *update_tetra*=1, a file containing the list of tetrahedrons as produced by another mesh generator, can be provided to OMEN.

- *tetra_file*: file containing a list of tetrahedrons that OMEN should use to create the stiffness matrix required by Poisson equation. The first line should contain the number of tetrahedrons. Then, each line should contain 4 integers, each of them corresponding to the index of the mesh point in the file grid.dat that forms one of the tetrahedron corners.

- *Eps_wire*: relative dielectric constant of the material in the channel regions. If the selected material, for example InAs_InGaAs_InAlAs is selected, then *Eps_wire* must have 3 entries, [15.15 14.2 13.8] where 14.2 (13.8) corresponds to the dielectric constant of InGaAs (InAlAs) with an In concentration as defined in *mat_binary_x*.

- *Eps_ox*: relative dielectric constant of the oxide regions. If there regions with different oxide materials, then *Eps_ox* should have different entries, as *Eps_wire*. The oxide type in each region can be selected with the variable *ox_id(n)*. For example, if *Eps_ox*=[3.9 20.0], the n^{th} oxide region with *ox_id(n)*=2 will have a relative dielectric constant of 20.0.

2.13 Electron-Phonon and Phonon-Phonon Scattering

This section describes the different parameters that can be modified the parametrize electron-phonon and phonon-phonon scattering in nanostructures. Some of them are specific to electron or phonon transport only, which is explicitly stated.

- *ph_energy_file*: (electron transport only). File containing the phonon energies of the confined nanostructure. This option was introduced at the beginning of the development of the electron-phonon scattering model in OMEN. At this time, the phonon bandstructure was computed outside of OMEN and then loaded into it. This is no more necessary since OMEN can compute these quantities now. In case this option is still needed, the first line of the file provided to OMEN should contain the number of phonon sub-bands N_M, the second line the number of phonon wave vectors N_Q, then a $N_M \times N_Q$ matrix. Note that N_M must be equal to $3N_A$ where N_A is the number of the atoms in the unit cell used to compute the phonon bandstructure. This must be an ascii file.

- *ph_mode_file*: (electron transport only). Same as *ph_energy_file*, but for the phonon eigenmodes. The file must be of size $N_M \times (N_M \times N_Q)$. This must be a binary file that can be read by the function MPI_Read.

- *ph_file_type*: (electron transport only). Not used.

- *no_ph_k*: (electron transport only). Number of phonon wave vectors N_Q used to compute the phonon bandstructure along the q_x direction. Default: -1. If not defined (*no_ph_k*=-1), then OMEN looks for a file containing the phonon energies and eigenmodes. If they are not found, then the simulation exits. Recommended value: *no_ph_k*=51.

- *no_ph_energy*: (electron transport only). As explained in Ref. [21], OMEN divides the phonon spectrum in *no_ph_energy* regions and attributes an energy and an electron-phonon scattering interaction to each of them. By default, the phonon spectrum is divided in 30 regions of equal size.

- *sc_max_iteration*: Maximum number of self-consistent Born iterations between the Green's Functions and the scattering self-energies. Default: 100.

- *sc_restart*: parameter to make OMEN write designated scattering self-energies to a file so that they can be read again if the simulation has to be restarted. If *sc_restart*=[-1 n], then each n self-consistent Born iterations, OMEN will write the scattering self-energies to binary files. By default, the files will be written in a directory called data that must be placed in the same directory as the one where the results are stored. It is the user responsibility to create the data directory. If *sc_restart*=2, OMEN, at the first self-consistent Born iterations, will read the scattering self-energies stored in the directory data instead of initializing them to 0.

- *sc_rh_crit*: (electron transport only) convergence criterion for the charge density in the self-consistent Born iterative process. Default: 1e-3. Usually, the charge density requires less self-consistent Born iterations than the current density to converge. Hence, as long as Poisson equation has not fully converged, OMEN only ensures that the charge density is OK according to the self-consistent Born iterative process. This approach allows reducing the total number of Born iterations required per bias point.

- *sc_id_crit*: convergence criterion for the electron or thermal current. Default: 1e-2. This means that the difference between the current extracted at the beginning and at the end of the simulation domain must be smaller than 1% and that current variations between 2 consecutive Born iterations must be smaller than 1%.

- *sc_dist_dep*: (electron transport only). To calculate the electron-phonon scattering interactions, the derivative of the tight-binding overlap matrix elements must be computed [21]. If *sc_dist_dep*=0 (default), only the derivative with respect to angular variations between 2 nearest-neighbor atoms is calculated. If *sc_dist_dep*=1, the derivative with respect to bond length variations is also taken into account.

- *sc_diag_def*: (electron transport only). Not used.

- *sc_k_coupling*: (electron transport only). If *NDim*=2, electron-phonon scattering couples different electron momentum together. If *sc_k_coupling*=0, all the momentum points are treated independently from each other (computationally more efficient). If *sc_k_coupling* (default), all the momentum points are coupled altogether.

- *sc_scale_fact*: factor that can be used to multiply the electron-phonon and phonon-phonon scattering interactions. Default value: 1.0.

- *sc_vbound*: (phonon transport only). Strength of the boundary scattering interaction in phonon transport calculation with anharmonic phonon-phonon scattering. Default value: 0.0.

- *incoherent_injection*: if 1, then scattering is introduced in the semi-infinite reservoirs from which electrons are injected into the simulation domain. This removes artificial reflection at the contact-device interfaces. If 0 (default), the semi-infinite reservoirs are treated in the ballistic limit of transport.

2.14 Voltages and Temperatures

- *Temp*: simulation temperature, value in [K]. In thermal simulations, *Temp* is the temperature of the source contact only.

- *NTemp*: number of drain temperatures in thermal simulations, integer value greater or equal to 1.

- *Tmin*: absolute minimum of the drain temperature, value in [K].

- *Tmax*: absolute maximum of the drain temperature, value in [K]. The drain temperatures are ramped from *Tmin* to *Tax* in steps of (*Tmax*-*Tmin*)/(*NTemp*-1).

- *NVG*: number of gate voltage points, integer value greater or equal to 1.

- *Vgmin*: absolute minimum of the gate voltage, value in [V].

- *Vgmax*: absolute maximum of the gate voltage, value in [V]. The bias points are ramped from *Vgmin* to *Vgmax* in steps of (*Vgmax*-*Vgmin*)/(*NVG*-1).

- *NVS*: number of source voltage points, integer value greater or equal to 1.

- *Vsmin*: absolute minimum of the source voltage, value in [V].

- *Vsmax*: absolute maximum of the source voltage, value in [V]. The bias points are ramped from *Vsmin* to *Vsmax* in steps of (*Vsmax*-*Vsmin*)/(*NVS*-1).

- *NVD*: number of drain voltage points, integer value greater or equal to 1.

- *Vdmin*: absolute minimum of the drain voltage, value in [V].
- *Vdmax*: absolute maximum of the drain voltage, value in [V]. The bias points are ramped from *Vdmin* to *Vdmax* in steps of (*Vdmax*-*Vdmin*)/(*NVD*-1).

2.15 Parallelization

- *NPROW*: not used.
- *NPCOL*: not used.
- *NPCS*: number of CPUs working on the same 1-D structure partition. This option can only be used in scattering cases solved in the NEGF formalism. If *NPCS*=1 (default), each CPU solves first the retarded Green's Function equation, then the lesser one, and finally the greater one. If *NPCS*=2, half of the CPUs solves the retarded Green's Function equation and then the lesser one while half of the CPUs solves the retarded Green's Function equation and then the greater one. A speed up factor of roughly 1.5 can be obtained with *NPCS*=2 and 2 times more CPUs than with *NPCS*=1.
- *spec_decomp*: determine how the Hamiltonian/dynamical matrix can be decomposed in sub-domains. If *spec_decomp*=0 (default), the matrix is cut at the end of a block corresponding to an atomic layer. Must be used in the Wave Function approach. If *spec_decomp*=1, the matrix is cut at the end of a block corresponding to a unit cell or slab. Must be used with the PDIV algorithm. If *spec_decomp*=2, the matrix is also cut at the end of a block corresponding to a unit cell or slab. Mus be used if the parallel RGF algorithm is used to solve simulations with electron-phonon or phonon-phonon scattering.
- *CPU_ppoint*: number of CPUs per energy point, integer value. In large device structures, OMEN offers the possibility to spatially decompose the simulation domain in *CPU_ppoint* slices along the x-axis, each of them having roughly the same width. This means that the corresponding Schrödinger equation is solved in parallel on *CPU_ppoint* different CPUs. This option is only possible if a parallel sparse linear solver such as MUMPS, SuperLU$_{dis}$, or the basis compression algorithm is selected with the *command* option.
- *CPU_per_kz_point*: number of CPUs per momentum point. Default: all (momentum points are solved one after the other).
- *CPU_per_sample*: number of CPUs per device samples when several samples with different structures must be simulated. Default: all (samples points are solved one after the other).
- *CPU_per_vg_point*: number of CPUs per gate bias point. Default: all. (gate bias points are solved one after the other) Works only in electron transport simulations.

- *CPU_per_vd_point*: number of CPUs per drain bias point. Default: all. (drain bias points are solved one after the other) Works only in electron transport simulations.
- *CPU_per_temp_point*: same as *CPU_per_vg_point*, but for temperatures instead of gate bias points. Works only in phonon transport simulations.
- *CPU_per_wire*: number of CPUs sharing the same MPI communicator to construct the device atomic structure. Default: 512. This option was introduced because if too many CPUs share the same communicator to construct a device structure, deadlocks sometimes occur. By limiting the number of CPUs per communicator, this problem can be avoided.
- *CPU_per_bc*: not used.

2.16 Other Parameters

- *plot_all_k*: if 1, data for all the momentum points are stored (transmission, density-of-states, contact bandstructure). If 0 (default), only the data corresponding to k_y=k_z=0 are stored.
- *full_current*: if 0 (default), only the current between two structure slabs is computed. If 1, the current along each bond separating two nearest-neighbor atoms is computed.
- *robust_numerics*: if 1, blocks involving all the atoms within a slab instead of an atomic layer are used in the recursive Green's Function (RGF) algorithm. If 0 (default), the minimum block size is used. This option works only if the RGF algorithm is used. It stabilizes the computation of, for example, carbon nanotubes including scattering.
- *convergence_booster*: if 1, energy points for which the DOS and/or transmission exhibit unphysical (spurious) peaks are removed from the calculation of the charge and current density, respectively. If 0, all the energy points are kept. Was primarily introduced to remove spurious solutions of the PDIV algorithm.
- *restart*: parameter to restart OMEN from existing results. In the input deck, *restart*=[restart_opt IG_start IS_start ID_start] where restart_opt=1, 2, 3, or 4. If restart_opt=1, OMEN needs to restart the simulation at the beginning of the bias point with index (IG_start,IS_start,ID_start). Some bias points were already computed and the corresponding electrostatic potentials are stored in a file passed to the input *vtot_file* (see below). If restart_opt=2, OMEN needs to restart in the middle of the bias point with index (IG_start,IS_start,ID_start), which usually corresponds to the first bias point of the simulation. The latest electrostatic potential that OMEN calculated is stored in the file passed to

the parameter *vact_file*. If restart_opt=3, OMEN needs to restart in the middle of the bias point with index (IG_start,IS_start,ID_start), some bias points were already computed (their electrostatic potential is passed to *vtot_file*), and the latest electrostatic potential OMEN generated is passed to *vact_file*. If restart_opt=4, the electrostatic potential is set to 0 everywhere.

- *vtot_file*: file containing the electrostatic potential of all the bias points that were already successfully computed by OMEN before a simulation is restarted
- *vact_file*: file containing the latest electrostatic potential that OMEN produced before a simulation is restarted. This option can also be used to load a pre-defined potential in bandstructure, transmission, or DOS calculations.
- *directory*: directory where the results are saved. If it is not defined, the results are saved in the current directory.
- *command(n)*: task that the simulator must accomplish, possible values are given in the next Chapter. A maximum of 20 commands for a given structure configuration is possible. However, many command files describing different structures can be started at the same time (see Chapter 1).

Chapter 3

Command Options and Output Files

3.1 Introduction

As mentioned in Chapter 1, OMEN can perform multiple different tasks and use different numerical solvers to achieve some of them. With the input parameter *command(n)* defined in the previous Chapter, the user can specify what task OMEN should accomplish. Here, a list of all the possible tasks is reported including the required command and the corresponding output data produced by the code.

3.2 Structure Characterization

All the commands listed in this Section provide information about the simulation domain created by OMEN and based on the input parameter specified by the user.

3.2.1 Nearest-Neighbor Table

Input Command

Write_Layer_Matrix: write a table containing information about the atom positions (coordinates), type, and nearest-neighbor connections.

Output File

Layer_Matrix.dat: $NA \times NC$ matrix where NA is the number of atoms in the channel region of the device (regions defined with *mat_* parameters) and NC the number of columns (NC=4 plus the maximum number of nearest-neighbor atoms per atom, for example 4 in zincblende structure or 3 in graphene). The first 3 columns of *Layer_Matrix.dat* are the x, y, and z coordinates of the atoms, the 4th column the type of the atom (1 for first anion type, 2 for first cation type, 3 for second anion

type, 4 for second cation type and so on). Columns 5 to *NC* contain the index of the nearest-neighbors. Note that x, y, and z automatically align themselves with the crystal orientations defined by the input deck. For example, if a user specifies x=[1 1 1], then the first column of *Layer_Matrix.dat* will be the x coordinate along the [1 1 1] crystal axis. If for example the line 25 of *Layer_Matrix.dat* has the following entries, assuming a zincblende crystal

0.243 2.67 1.25 2 17 0 26 31

then the atom 25 is situated at [0.243 2.67 1.25], is of type 2 (first cation type), and has 3 nearest-neighbors (atoms situated at line 17, 26, and 31 in the table) and one dangling bond (0 denotes a dangling bond).

3.2.2 Grid Matrix

Input Command

Write_Grid_Matrix: write information about the finite element grid that OMEN uses to solve Poisson equation and other general informations.

Output Files

This command produces several output files:

- *grid.dat*: *NGrid*× 3 matrix where *NGrid* is the number of mesh points in the simulation domain (number of nodes in the finite element grid). Each line contains the [x y z] coordinates of a different mesh point.
- *atom_index.dat*: *NA*× 1 vector whose entries are the index of the *NA* atoms in the grid matrix. For example *atom_index*(n)=m means that the n^{th} atom corresponds to the m^{th} grid point. Caution: m goes from 0 to *NGrid*-1.
- *gate_index.dat*: 1×*NGate* vector where *NGate* is the number of mesh points that belong to a gate contact and whose electrostatic potential gets fixed. Its entries are the index of the *NGate* mesh points that belong to a gate contact. For example *gate_index*(n)=m means that the n^{th} gate point corresponds to the m^{th} grid point. Caution: m goes from 0 to *NGrid*-1.
- *layer_per_slab.dat*: number of atomic layers per structure unit cell (or slab). For example, in a nanowire with transport along the <100> crystal axis, there are 4 atomic layers per slab, 6 if transport occurs along <111>.
- *number_of_slabs.dat*: number of unit cells (or slabs) composing the channel structure.
- *Lx.dat*: width of a structure unit cell (or slab) along the x-axis.

- *Ly.dat*: width of a periodic structure unit cell (or slab) along the y-axis. Its value is different from ∞ only if *NDim*=1.

- *Lz.dat*: width of a periodic structure unit cell (or slab) along the z-axis. Its value is different from ∞ only if *NDim*=1 or 2.

3.2.3 Doping Concentration

Input Command

Write_Doping: write the doping concentration to a file.

Output File

doping.dat: $1\times NGrid$ vector containing the doping concentration (unit: number of doping charge per atom, positive for donors, negative for acceptors). The entry orders follows that of *grid.dat*.

3.2.4 Tetrahedron Mesh

Input Command

Write_Tetrahedron: write the list of tetrahedron generated by the qhull library [45] based on the structure grid points to a file.

Output File

tetrahedron_dat: $(NTetra+1)\times 5$ matrix where *NTetra* is the number of tetrahedron created by qhull. The first line of this file contains *NTetra*. The 4 first columns of the other lines are the indices of the 4 points that compose each tetrahedron. These indices refer to the position of the mesh point within the *grid.dat* matrix. The 5th column of *tetrahedron_dat* represents the volume of each tetrahedron (unit: nm^3).

3.2.5 Stiffness Matrix

Input Command

Write_Fitness_Matrix: write the stiffness matrix of Poisson equation to a file.

Output File

FitMat_dat: file containing the stiffness matrix of Poisson equation. The first line is the size of the matrix, the second line the number of non-zero entries, the third

line the fortran index, then each line contains the row and column index as well as the value of all the non-zero entries.

3.2.6 Fermi Level

Input Command

Write_Fermi_Level: write the equilibrium Fermi level of the right and left contacts according to their respective doping concentration.

Output Files

This function produces two output files:

- *Efl.dat*: equilibrium Fermi level of the left contact (unit: eV).
- *Efr.dat*: equilibrium Fermi level of the right contact (unit: eV).

3.3 Bandstructure Calculation

One of the principle features of OMEN is its capabilities to compute the electron/hole and phonon bandstructures of different types of nanostructures. This Section presents the different available options.

3.3.1 Contact Bandstructure

After setting up a device structure, OMEN automatically extracts its first and last unit cells and assume that they form the semi-infinite left and right reservoirs (contacts) needed to inject and collect electrons/holes/phonons. OMEN can calculate the bandstructure of these semi-infinite reservoirs with the following commands:

Input Commands

- *XB_Bandstructure*: (only for electrons/holes, i. e. *transport_type*=0). The conduction subbands are calculated if X=C, the valence ones if X=V.
- *Bandstructure*: (only for phonons, i. e. *transport_type*=1). Calculate the phonon bandstructure of the semi-infinite reservoirs.

Output Files

- *XB_k.dat*: (only for electrons/holes). The wave vector k_x is saved in this $1 \times Nk$ vector. It goes from -π to π. X is equal to C if the conduction band is calculated, to V for the valence band.

- *XB_Ekl.dat*: (only for electrons/holes). The bandstructure of the left semi-infinite contact is stored in this *n_of_modes*×*Nk* matrix where X is equal to C if the conduction band is calculated, V otherwise. In nanostructures with *NDim*<3, if *Nky* and/or *Nkz* is larger than one, then the size of *XB_Ekl.dat* becomes (*Nky*×*Nkz*×*n_of_modes*)×*Nk*. OMEN goes one (k_y,k_z) configuration after the other and keeps appending the resulting band structure of size *n_of_modes*×*Nk* at the end of the *XB_Ekl.dat* file.

- *XB_Ekr.dat*: (only for electrons/holes). Same as *XB_Ekl.dat*, but for the right semi-infinite reservoir. If *last_first*=1, then only *XB_Ekl* is calculated and *XB_Ekr*=*XB_Ekl*.

- *PH_kx.dat*: (only for phonons). Same as *XB_k.dat*, but for phonons.

- *PH_ky_n.dat*: (only for phonons). OMEN calculates *Nky*×*Nkz* contact bandstructures, as for electrons, but instead of writing all of them to a single file, store them in different files whose indices n range from 0 to *Nky*×*Nkz*-1. *PH_ky_n.dat* stores therefore the k_y value for the n^{th} bandstructure calculation. Unite: phase.

- *PH_kz_n.dat*: (only for phonons). Same as *PH_ky_n.dat*, but for k_z.

- *PH_Ekl_n.dat*: (only for phonons). File of size 3*NA*×*Nk* containing all the phonon modes as function of k_x for the n^{th} (k_y,k_z) configuration. Unit: eV.

- *PH_Ekr_n.dat*: (only for phonons). Same as *PH_Ekl_n.dat*, but for the right contact. If *last_first*=1, then only *PH_Ekl_n.dat* is calculated and *PH_Ekr_n.dat*= *PH_Ekl_n.dat*.

3.3.2 Closed Systems

This option is currently only applicable to electrons and holes, not phonons.

Input Command

XB_Closed: (only for electrons/holes). Calculate the bandstructure of a closed system (no open boundary conditions). The parameter *open_system* must be set to 0 in such cases. The conduction subbands are calculated if X=C, the valence ones if X=V.

Output Files

This command produces several output files:

- *X_E_n_m.dat*: file of size *n_of_modes*×1 containing the eigenenergies of the considered closed system for the n^{th} applied electrical field (with parameters

NVD, *VDmin*, and *VDmax*) and m^{th} (k_y,k_z) configuration. X is equal to C if the conduction band is calculated, V otherwise.

- *X_V_n_m.dat*: file of size $(NA \times N_{orb})\times 2$*n_of_modes* containing the eigenmodes of the considered closed system for the n^{th} applied electrical field (with parameters *NVD*, *VDmin*, and *VDmax*) and m^{th} (k_y,k_z) configuration. Here, NA is the number of atoms in the closed system and N_{orb} the number of orbitals per atom. The file has 2*n_of_modes* columns, because the even ones store the real part of each eigenmode and the odd ones the imaginary part. X is equal to C if the conduction band is calculated, V otherwise.

3.3.3 Optical Matrix Elements

This option is only for electrons and holes, not phonons.

Input Command

Optical_Matrix_Element: calculate the optical matrix element of a 3-D closed system (*NDim*=3 and *open_system*=0).

Output Files

- *CB_E_n.dat*: file of size *n_of_modes*$\times 1$ containing the conduction band eigenenergies of the considered 3-D closed system for the n^{th} applied electrical field (with parameters *NVD*, *VDmin*, and *VDmax*).

- *CB_V_n.dat*: file of size $(NAN_{orb})\times 2$*n_of_modes* containing the conduction band eigenmodes of the considered 3-D closed system for the n^{th} applied electrical field (with parameters *NVD*, *VDmin*, and *VDmax*). Here, NA is the number of atoms in the closed system and N_{orb} the number of orbitals per atom. The file has 2*n_of_modes* columns, because the even ones store the real part of each eigenmode and the odd ones the imaginary part.

- *VB_E_n.dat*: Same as *CB_E_n.dat*, but for the valence band.

- *VB_V_n.dat*: Same as *CB_V_n.dat*, but for the valence band.

- *Px_n.dat*: file of size *n_of_modes*$\times$*n_of_modes* containing the overlap matrix elements $< CB|p_x|VB >$ (p_x is the impulse operator along the x direction) between all the conduction $< CB|$ and valence $|VB >$ sub-bands that were calculated. The index n refers to the n^{th} applied electric field.

- *Py_n.dat*: Same as *Px_n.dat*, but for p_y.

- *Pz_n.dat*: Same as *Px_n.dat*, but for p_z.

3.4 Transmission and DOS Calculations

Another important feature of OMEN is the calculation of the electron/hole/phonon density-of-states (DOS) and transmission coefficient (TE) in multi-dimensional nanostructures. The required commands to perform these tasks and the produced output files are summarized in this Section.

Input Command

XXXTransmission_YZZ: The transmission and DOS of the considered nanostructures are computed with XXX=$CB\_$ for electrons, $VB\_$ for holes and "$\emptyset$" for phonons. Y and ZZ determine the solver and the computational approach, respectively. If the calculations are done in the Wave Function formalism, then ZZ=WF and Y=U for Umfpack [29], P for Pardiso [32], S for SuperLU$_{dist}$ [31], M for MUMPS [30], PM for MUMPS with PORD as reordering algorithm [30], C for the basis compression (or renormalization) algorithm [33], and O for the same algorithm, but with a shared instead of distributed memory parallelization. Note that the basis compression algorithm cannot be used with phonons. If the NEGF formalism is selected ZZ=GF and Y=R for the recursive Green's Function algorithm [28] and $PDIV$ for the algorithm developed by Steve Cauley [53] (no more supported for ballistic cases).

Output Files

Several output files are produced by this command:

- *YX_E_n.dat*: 1× *NEmax* vector containing the energy discretization points in [eV] for which TE and DOS are calculated. The parameter Z is equal to CB for electrons, VB for holes, and ph for phonons. Y refers to the solver and is equal to the values defined above (U, P, S, M, PM, C, O, R, or $PDIV$, capital letters for electrons/holes, small letters for phonons). The index n refers to the (k_y,k_z) configuration and runs from 0 to *Nky*×*Nkz*-1. OMEN computes all the different k_z points at a given k_y before moving to the next k_y point.
- *YX_TEl_n.dat*: *NSlab*×*NEmax* matrix describing the transmission from the left contact to all the structure unit cells (or slabs). For example the first line represents the transmission from the left contact to the first structure unit cell as function of the energy, the n^{th} line the transmission from the left contact to the n^{th} unit cell. In perfect structures without any applied bias, the transmission is the same for all the slabs and should therefore take only integer values.
- *YX_TEr_n.dat*: same as *YX_TEl_n.dat*, but the transmission from the right instead of the left contact to any device unit cell is stored.

- *YX_ZEl_n.dat*: $NSlab \times NEmax$ matrix whose elements are the sum of the density-of-states (injected from the left contact) of all the atoms situated in a device unit cell. For example the n^{th} line is the DOS contribution coming from all the atoms in the n^{th} unit cell as function of the energy. As for the transmission the DOS is constant along perfect structures without any applied bias.
- *YX_ZEr_n.dat*: Same as *YX_ZEl_n.dat*, but for the states injected from the right contact instead of the left one.
- *YX_Ekl_n.dat*: the bandstructure of the left semi-infinite contact is stored in this $n\_of\_modes \times Nk$ matrix. It corresponds to the n^{th} (k_y,k_z) configuration.
- *YX_Ekr_n.dat*: same as *YX_Ekl_n.dat*, but for the right semi-infinite contact.
- *YX_ky_n.dat*: value of the n^{th} k_y point. Unit: phase.
- *YX_kz_n.dat*: value of the n^{th} k_z point. Unit: phase.

3.5 Self-consistent Simulations

The last important feature of OMEN resides in its capacity to perform self-consistent Schrödinger-Poisson simulations for electrons/holes and to compute thermal currents for phonons, either in the ballistic limit of transport or in the presence of electron-phonon or anharmonic phonon-phonon scattering. The command to run such simulations and the output files generated by OMEN are summarized below:

Input Command

XXXSC_YZZ: the options for XXX are $EL\_$ for electron, $HO\_$ for hole, and "$\emptyset$" for phonon simulations. Note that for band-to-band tunneling transistors where both electrons and holes co-exist, either $EL\_$ or $HO\_$ can be selected. For ballistic simulations, as in Section 3.4, Y refers to the solver type and ZZ to the computational approach, with the same possible values as there. For simulations with either electron/hole-phonon scattering (electrons and holes) or anharmonic phonon-phonon scattering (phonons), YZZ can have the following values: $SCATT$ (recursive Green's Function algorithm [28]) or $PDIVSCATT$ (only for electrons/holes, PDIV algorithm developed by Steve Cauley [54]).

Output Files

Self-consistent simulations produce several output files. First, the parameters that are only defined in electron/hole simulations are given:

- *YXX_n_i.dat*: matrix of size $(NVG \times NVS \times NVD) \times NA$ where NA is the total number of atoms that compose the device structure. This matrix contains the number of electrons per atom. If several bias points are computed

in parallel (parameter *CPU_per_vg_point* is smaller than the total number of CPUs N_{CPU} used to run the simulation), then *YXX_n_i.dat* contains only the bias points that are computed by the i^{th} group of CPUs. The index i runs from 0 to $N_{CPU}/CPU\_per\_vg\_point$-1. The variable Y is the same as in the command definition, XX=EL for electron simulations and HO for holes.

- *YXX_n_act_i.dat*: vector of size $NA\times1$. Number of electrons per atom as obtained from the latest Schrödinger-Poisson iteration by the i^{th} group of CPUs.
- *YXX_p_i.dat*: same as *YXX_n_i.dat*, but for holes.
- *YXX_p_act_i.dat*: same as *YXX_n_act_i.dat*, but for holes.
- *YXX_Vpot_i.dat*: $(NVG \times NVS \times NVD)\times NGrid$ matrix where $NGrid$ is the total number of grid points (or nodes) in the entire device (channel and oxide layers). This matrix contains the electrostatic potential corresponding to all the bias points after the Schrödinger-Poisson iterations have converged. Same conventions as *YXX_n_i.dat* for YXX and i.
- *YXX_Vpot_act_i.dat*: $NGrid\times1$ vector containing the latest electrostatic potential produced by the i^{th} group of CPUs in OMEN. This potential can be used to restart a self-consistent simulation that stopped with the option *restart*_opt=2.
- *YXX_Vpot_old_i.dat*: $NGrid\times1$ vector containing the second-to-last electrostatic potential produced by the i^{th} group of CPUs in OMEN.
- *YXX_Id_i.dat*: $(NVG \times NVS \times NVD)\times NSlab$ matrix containing the device current in [A] if $NDim$=3, [A/m] if $NDim$=2, and [A/m^2] if $NDim$=1. The current is resolved at each slab (or unit cell) along the x-axis of the device structure. Due to current conservation the values in each unit cell should be the same.
- *YXX_Id_act_i.dat*: $NSlab\times1$ vector containing the latest device current that the i^{th} group of CPUs computed.
- *YXX_Id_xyz_i.dat*: $(NVG \times NVS \times NVD \times NA)\times NBond$ matrix containing the bond-resolved device current with the same units as *YXX_Id_i.dat*. Here, NA is the number of atoms in the structure and $NBond$ the number of bonds per atom. Hence, each line of this file represents the current starting from an atom and flowing along the bonds connecting it to its nearest-neighbors. The contain of this file is different from 0 only if the option *full_current* is turned on and if the Wave Function approach is selected (ballistic simulations).
- *YXX_Id_xyz_act_i.dat*: $NBond\times NA$ matrix containing the latest bond-resolved current that the i^{th} group of CPUs produced.

- *YXX_Id_kykz_i.dat*: $(NVG \times NVS \times NVD \times NA)\times(Nky \times Nkz)$ matrix containing the momentum-resolved current for each bias point. The sum of each line of this file gives the value stored in *YXX_Id_i.dat* at the same line.

- *YXX_Id_kykz_i.dat*: $(Nky \times Nkz)\times 1$ vector containing the latest momentum-resolved device current produced by the i^{th} group of CPUs.

- *YXX_E_l_m_n_k.dat*: $1\times$ *NE* vector containing the *NE* energy points in [eV] at which the transmission and density-of-states are calculated for the l^{th} gate bias, m^{th} source bias, n^{th} drain bias, and k^{th} (k_y,k_z) configuration. By default, OMEN stores only the data for k_y=k_z=0. To obtain the data for all the possible momentum, the parameter *plot_all_k* must be set to 1.

- *YXX_TE_l_m_n_k.dat*: $1\times$ *NE* vector containing the values of the transmission from the left to the right contact for the l^{th} gate bias, m^{th} source bias, n^{th} drain bias, and k^{th} (k_y,k_z) configuration. In simulations including electron-phonon scattering, this file does not contain the transmission probability, but the spatially- and energetically-resolved spectral current of the device.

- *YXX_ZEl_l_m_n_i.dat*: *NSlab*$\times$*NE* matrix whose elements are the sum of the density-of-states (injected from the left contact) of all the atoms situated in a device unit cell for the l^{th} gate bias, m^{th} source bias, n^{th} drain bias, and k^{th} (k_y,k_z) configuration. In simulations with electron-phonon scattering, this file contains the spatially- and energetically-resolved electron density of the device.

- *YXX_ZEr_l_m_n_i.dat*: same as *YXX_ZEl_l_m_n_i.dat*, but for the states injected from the right contact. In simulations with electron-phonon scattering, this file contains the spatially- and energetically-resolved hole density of the device.

- *YXX_Ekl_l_m_n_i.dat*: *n_of_modes*$\times$*Nk* matrix containing the bandstructure of the left semi-infinite contact for the l^{th} gate bias, m^{th} source bias, n^{th} drain bias, and k^{th} (k_y,k_z) configuration.

- *YXX_Ekr_l_m_n_i.dat*: same as *YXX_Ekl_l_m_n_i.dat*, but for the right contact.

- *YXX_condition_i.dat*: *NI*$\times 1$ vector containing the relative error of the electrostatic potential for the *NI* Schrödinger-Poisson iterations that were calculated by the i^{th} group of CPUs.

- *YXX_Vg_Vs_Vd_i.dat*: $(NVG \times NVS \times NVD)\times$ 3 matrix storing the value of the gate, source, and drain bias points that were simulated by the i^{th} group of CPUs.

- *YXX_CPU_mapping_i.dat*: matrix of size $3\times(Nky \times Nkz)$ containing information about the distribution of the CPUs across the different momentum groups that are simultaneously treated by OMEN. It must be recalled here that OMEN automatically redistributes the number of CPUs attributed to each momentum group according to the number of energy points they have to deal with. Hence, the first line of this file reports the index of the first CPU working on each momentum group, the second line the number of CPUs per momentum point, and finally, the third line, the number of energy points per momentum.

- *condition_i.dat*: (only for simulations with electron-phonon scattering). Matrix of size $NBorn\times3$ containing the value of the electron/hole current in the first (first column) and last (second column) unit cell of the device structure as well as the change in the charge density (third column) between two consecutive self-consistent Born iterations. Here, $NBorn$ is the number of self-consistent Born iterations that were required for the Green's Functions and scattering self-energies to converge. The file *condition_i.dat* monitors therefore the convergence of the self-consistent Born iterations for the i^{th} group of CPUs.

OMEN creates also output files that are specific to phonon transport simulations. Note that no self-consistent Schrödinger-Poisson iterations are needed for phonons, only one solution of a dynamical equation per temperature. The filenames still contain the SC denotation although no self-consistency is present (except for simulations with scattering). The list of phonon parameters include:

- *YXX_phrhoE_i_j.dat*: matrix of size $NTemp \times NA$ where NA is the total number of atoms that compose the device structure and $NTemp$ the number of drain contact temperatures that are considered. This matrix contains the phonon density per atom. As for electrons/holes, the letter Y refers to the solver (U, P, M, $\cdots$) and XX to the computational approach (WF or GF) in ballistic simulations. In simulations with anharmonic phonon-phonon scattering, YXX is replaced by $SCATT$. If several drain temperature points are computed in parallel (parameter *CPU_per_temp_point* is smaller than the total number of CPUs N_{CPU} used to run the simulation), then *YXX_phrhoE_i_j.dat* contains only the temperature points that are computed by the i^{th} group of CPUs. The index i runs from 0 to $N_{CPU}/$*CPU_per_temp_point*-1. Since OMEN can simulate different samples defined by the same command file, but with some properties that are randomly generated (rough surfaces for example), the last digit j in *YXX_phrhoE_i_j.dat* refers to the sample index. These samples can be treated either sequentially or in parallel if *CPU_per_sample* is smaller than N_{CPU}.

- *YXX_phIdE_i_j.dat*: matrix of size $NTemp \times NSlab$ containing the thermal current flowing through the device structure with unit [W] if *NDim*=3, [W/m]

if *NDim*=2, and [W/m^2] if *NDim*=1. The current is resolved at each slab (or unit cell) along the x-axis of the device structure. Due to current conservation the values in each unit cell should be the same.

- The transmission *YXX_TE_t_k_j.dat*, left-injected *YXX_ZEl_t_k_j.dat* and right-injected *YXX_ZEr_t_k_j.dat* density-of-states are defined as for electrons/holes, except for the indices t (temperature), k ($k_y - k_z$ configuration) and j (sample number) that have a different meaning.

- The files containing the left *YXX_Ekl_t_k.dat* and right *YXX_Ekr_t_k.dat* contact bandstructures are defined as for electrons/holes, except for the indices t (temperature) and k ($k_y - k_z$ configuration).

- The file containing information about the CPU distribution among the different momentum groups *YXX_CPU_mapping_t_j.dat* is defined as for electrons/holes, except for the indices t (temperature) and j (sample number) that have a different meaning.

- The file *condition_t_j.dat* monitors the convergence of the self-consistent Born iterations when anharmonic phonon-phonon scattering is present. It is defined as for electrons/holes, except that it has only two columns (thermal current in the first and last unit cell). Furthermore, the indices t (temperature) and j (sample number) have a different meaning.

Chapter 4

Simulation Examples

To help beginners start with OMEN, several examples accompany this manual, covering most of the capabilities of the simulator. They can be found in a directory called *EXAMPLE*. All the input parameters are summarized in Chapter 2 and the output file types in Chapter 3. The examples do not need to be modified before being launched, except eventually the line *directory*, which specifies where the results will be saved. It is recommended to use these examples as a starting point to create new device structures. Note that OMEN does not include any tool to visualize the output results. All the figures shown here were created with MATLAB. Therefore, the MATLAB files used to generate them are also stored in the *EXAMPLE* directory.

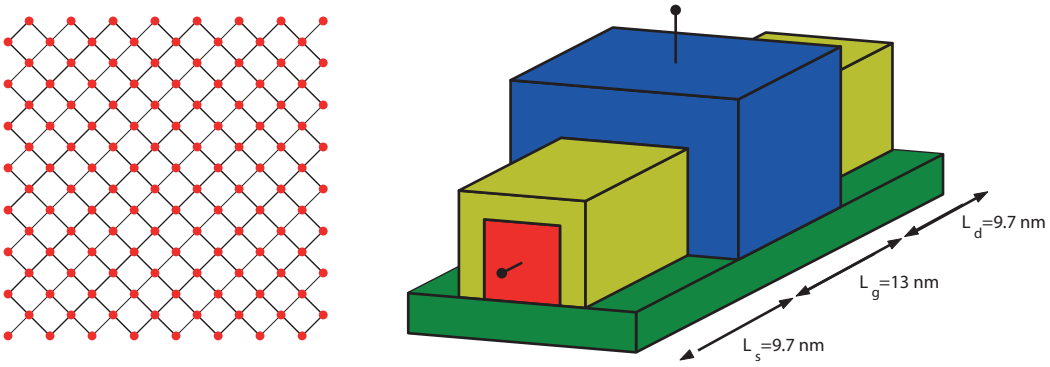


Figure 4.1: (Left) Cross section of a Si nanowire with a 2.1×2.1 nm^2 square cross section and transport along the <100> crystal axis. (Right) Schematic view of a 2.1×2.1 nm^2 square triple-gate nanowire field-effect transistor. The oxide layers have a thickness of 1 nm, transport occurs along the <100> crystal axis, y=(010) and z=(001) are directions of confinement, and the gate length L_g measures 13 nm.

4.1 Si Square Nanowire

4.1.1 Introduction

This example deals with a Si square nanowire with a 2.1×2.1 nm^2 cross section and transport along the <100> crystal axis, as shown in Fig. 4.1 (left). All the required command files can be found in the directory *EXAMPLE/Square_NW*.

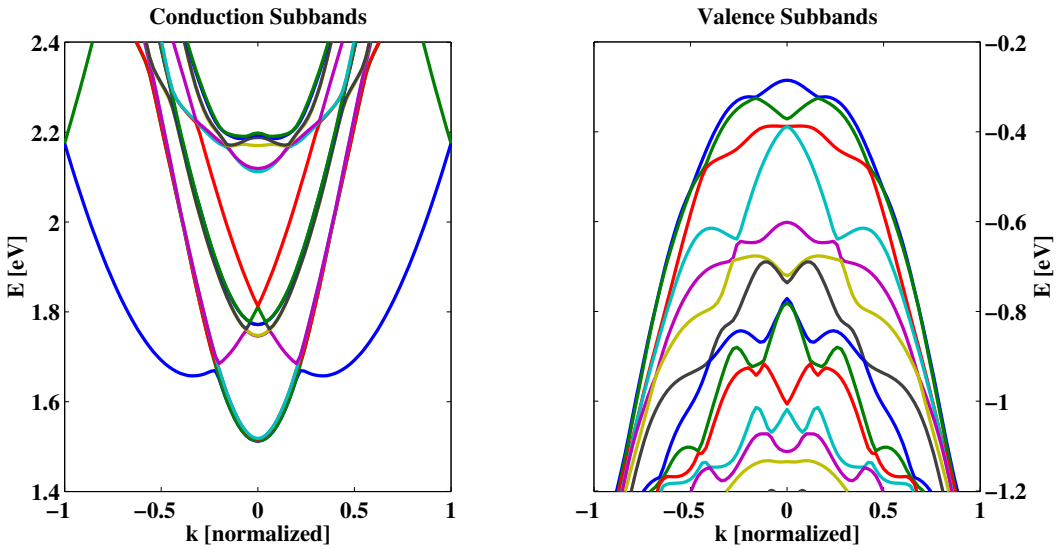


Figure 4.2: Electron (left) and hole (right) subbands of an infinite Si nanowire with a 2.1×2.1 nm^2 square cross section and transport along the <100> crystal axis.

4.1.2 Bandstructure

First, the electron and hole bandstructure of the left and right contacts of this square nanowire should be computed. The file *bandstructure_square.cmd* in the directory *EXAMPLE/Square_NW/BS* has been prepared for that purpose. The simulation should last less than 5 minutes on a single CPU. The ARPACK eigenvalue solver is employed to obtain *n_of_modes*=16 sub-bands at *Nk*=51 different k_x points. The results are plotted in Fig. 4.2. To visualize the data, the matlab functions *plot_cross_section.m* (atomic cross section) and *plot_bandstructure.m* can be used.

4.1.3 Transmission and density-of-states

The transmission and density-of-states of the square Si in Fig. 4.1 can be calculated with the function *transmission_square.cmd* that can be found in the directory *EXAMPLE/Square_NW/Transmission*. A total *NEmax*=100 energy points are calculated with the basis compression algorithm [33] implemented in OMEN. The simulation time T_0 should not exceed 10 minutes on a single CPU, while it

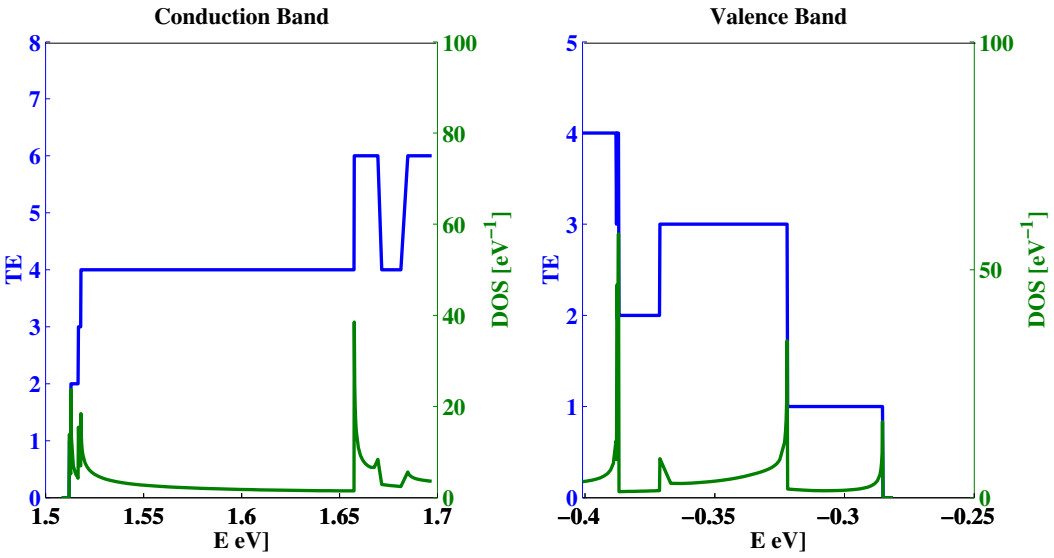


Figure 4.3: Electron (left) and hole (right) transmission (blue) and density-of-states (green) of a Si nanowire with a 2.1×2.1 nm^2 square cross section, a perfect structure, no applied bias, and transport along the <100> crystal axis.

should be reduced to T_0/N on N CPUs. The results for the 2.1×2.1 nm^2 square Si nanowire are plotted in Fig. 4.3, which has been created with the matlab function *plot_transmission_dos.m*.

4.1.4 Ballistic self-consistent simulation

The square nanowire can be used as the channel of a triple-gate field-effect transistor, as illustrated in Fig. 4.1 (right) and then simulated in the ballistic limit of transport for different applied gate-to-source voltages. The file *n_wire_square.cmd* in the directory *EXAMPLE/Square_NW/SC/nFET* describes a triple-gate n-doped Si nanowire transistor with a 2.1×2.1 nm^2 cross section. The source, the channel, and the drain measure 9.7 nm, 13 nm, and 9.7 nm, respectively. They are surrounded by 1 nm thick oxide layers. The source and drain regions are doped with a donor concentration N_D=10^{20} cm^{-3}. The file *p_wire_square.cmd* in the directory *EXAMPLE/Square_NW/SC/pFET* contains exactly the same structure, but the source and drain are p-doped with an acceptor concentration of N_A=10^{20} cm^{-3}, and the gate work function is increased to ϕ_m=5.0 eV instead of ϕ_m=4.25 eV. The simulation time is about 4 (nFET) to 8 (pFET) hours on 32 CPUs.

The transfer characteristics at V_{ds}=±0.6 V of the n- and p-type triple-gate transistors are shown in Fig. 4.4. On the left, the p-FET $I_d - V_{gs}$ is given, on the right the n-FET characteristic. The matlab function *plot_id_vg.m* can be used to generate such a plot. Note that the current in all the unit cells and for all the bias points is represented. Since it is conserved, only one single line is visible.

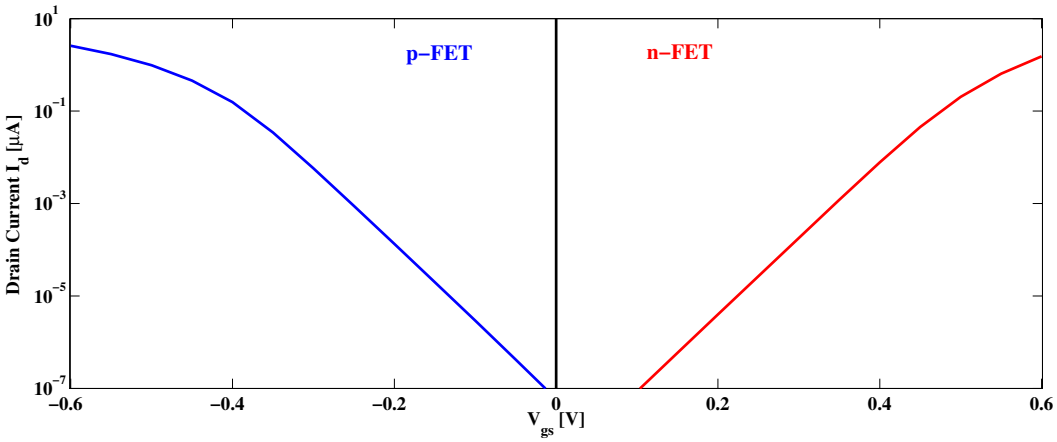


Figure 4.4: Transfer characteristics of the square nanowire transistor in Fig. 4.1 (right) at V_{ds}=±0.6 V. A p-FET (blue) and a n-FET (red) were simulated.

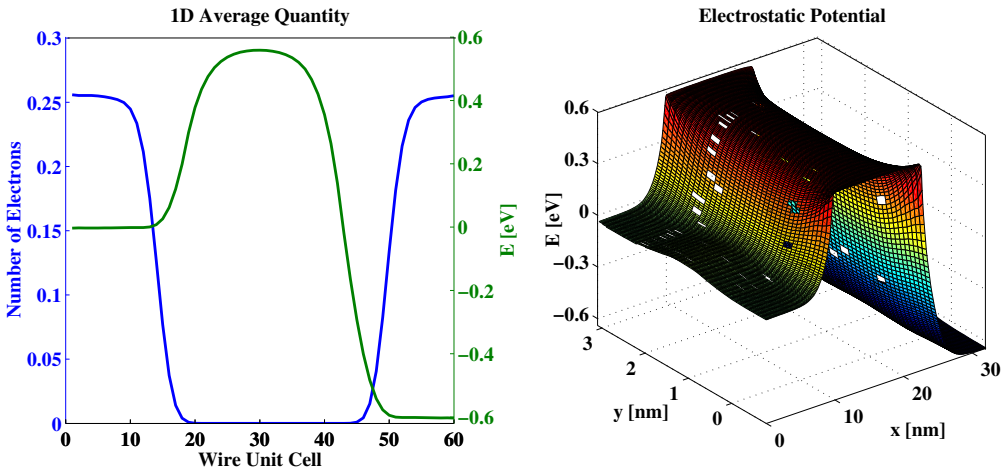


Figure 4.5: Internal quantities of the square nanowire transistor in Fig. 4.1 (right) at V_{gs}=0.0 V and V_{ds}=0.6 V. On the left, the number of electrons per wire unit cell (blue) and the averaged electrostatic potential (over the number of atoms) are given. On the right, a cut of the 3D electrostatic potential at z=1.3 nm is shown along the x and y axis.

Some users may be interested to see what happens within the transistor structure, for example how the carrier density or the electrostatic potential look like. Unfortunately, these quantities cannot be directly plotted since the carrier density is given at the position of the atoms only and the electrostatic potential on a non-uniform finite element grid. However, there are two possibilities to circumvent this problem. The first one consists in averaging the charge density and the electrostatic potential over one unit cell, i.e. evaluating the following expressions

$$n_i = \int_{\Omega_i} \mathrm{d}x\mathrm{d}y\mathrm{d}z\, n(x,y,z) \tag{4.1}$$

$$V_i = \frac{1}{n_i}\int_{\Omega_i} \mathrm{d}x\mathrm{d}y\mathrm{d}z\, V(x,y,z)\cdot n(x,y,z) \tag{4.2}$$

where Ω_i is the volume of the i^{th} wire unit cell and $n(x,y,z)$ and $V(x,y,z)$ are the position-dependent electron density and electrostatic potential, respectively. The variables n_i and V_i are shown in Fig. 4.5 (left) for the square n-doped transistor at V_{gs}=0.0 V and $V_{ds} = 0.6V$. The figure was generated using the first part of the MATLAB function *plot_internal_quantities*.

This gives a good insight into the distribution of the electrons and the shape of the potential barrier. There is a second possibility to visualize internal quantities and obtain more information by interpolating $n(x,y,z)$ and $V(x,y,z)$ to the desired regular mesh. A cut through the three-dimensional electrostatic potential of the n-doped square nanowire is shown in Fig. 4.5 (right) at V_{gs}=0.0 V and $V_{ds} = 0.6V$. The variable $V(x,y,z)$ is shown along the x and y axis at z=1.3 nm. This kind of 3-D objects can be generated with the second part of the MATLAB function *plot_internal_quantities.m*.

4.2 Si Circular Nanowire

4.2.1 Introduction

The second application is about a circular nanowire with a diameter of d=2.5 nm and transport along the <111> crystal orientation, as shown in Fig. 4.6. All the command files that are needed to simulate the bandstructure, transmission and $I-V$ characteristics of this device can be found in the directory *EXAMPLE/Circular_NW/*.

4.2.2 Bandstructure

The electron and hole bandstructure of the left and right contacts of this circular nanowire can be computed with the file *bandstructure_circle.cmd* in the directory *EXAMPLE/Circular_NW/BS*. The simulation should last less than 2 minutes on a single CPU. The ARPACK eigenvalue solver is employed to obtain *n_of_modes*=16 sub-bands at *Nk*=51 different k_x points. Results are shown in Fig. 4.7. To visualize the data, the matlab functions *plot_cross_section.m* (atomic cross section) and *plot_bandstructure.m* can be used.

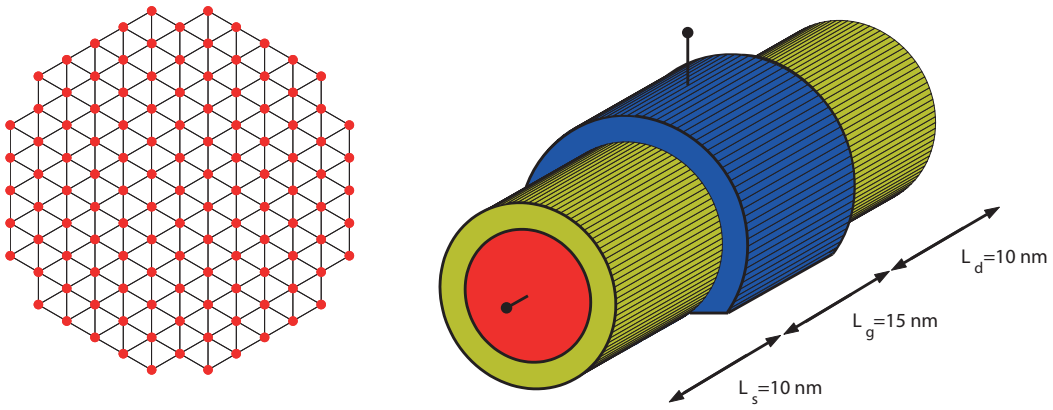


Figure 4.6: (Left) Cross section of a Si nanowire with a circular cross section (diameter d=2.5 nm) and transport along the <111> crystal axis. (Right) Schematic view of a Ω-gated Si nanowire field-effect transistor with a diameter of 2.5 nm, a 15 nm gate length, 10 nm long source and drain extensions (donor doping concentration N_D=1e20 cm^{-3}), and 1 nm thick oxide layers. Transport occurs along the <111> crystal axis, y=($\bar{1}$10) and z=(11$\bar{2}$) are directions of confinement.

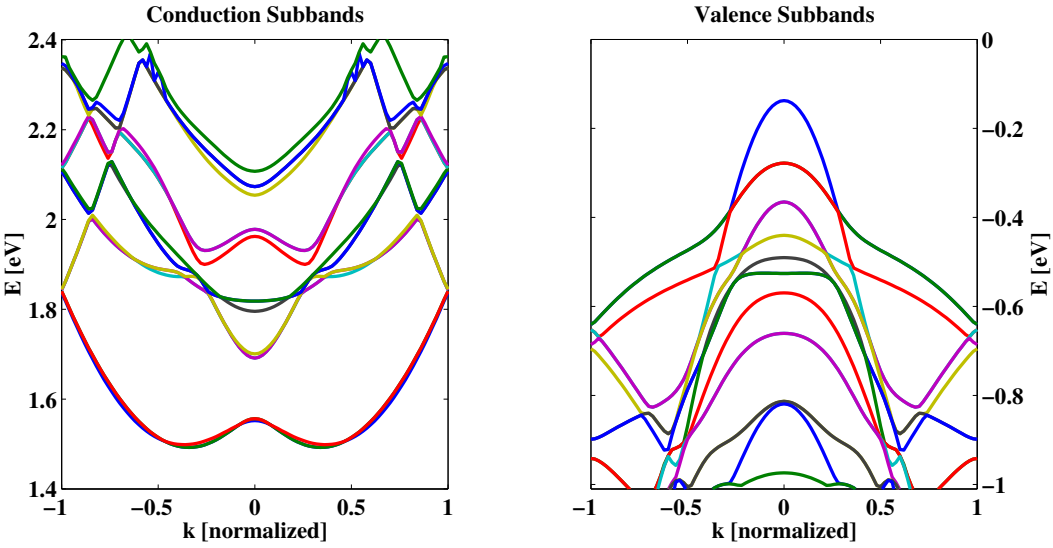


Figure 4.7: Electron (left) and hole (right) subbands of an infinite circular Si nanowire with a diameter d=2.5 nm and transport along the <111> crystal axis.

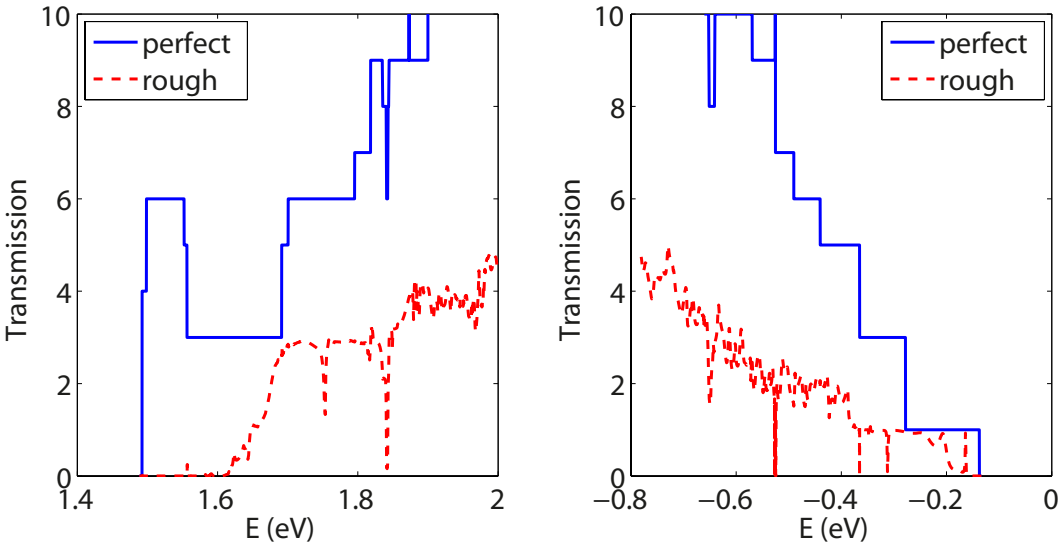


Figure 4.8: Electron (left) and hole (right) transmission through a circular Si nanowire with a diameter of 1.25 nm, transport along the <111> crystal axis. A perfect (blue lines) and rough (dashed red lines) nanowire surface are considered.

4.2.3 Transmission through a perfect and rough structure

The transmission probability through the circular Si nanowire in Fig. 4.6 can be calculated assuming a perfect surface (file *transmission_circle.cmd* in the directory *EXAMPLE/Circular_NW/Transmission/perfect*) or a rough surface (file *transmission_circle_rough.cmd* in the directory *EXAMPLE/Circular_NW/ Transmission/roughness*). In the latter case, the correlation length of the rough surface is set to 2 nm and its root mean square to 0.15 nm. A total $NE \approx 1000$ energy points are calculated with the basis compression algorithm [33]. The simulation time T_0 amounts to roughly 40 minutes on 32 CPUs. Results are shown in the Fig. 4.8 for both the conduction (left) and valence (right) band. Note that the rough surface is determined through a random process so that each new execution of the file *transmission_circle_rough.cmd* will produce different results. The matlab function *plot_transmission.m* can be used to produce a plot similar to Fig. 4.8.

4.2.4 Ballistic self-consistent simulation

The command file *wire_circle.cmd* in the directory *EXAMPLE/Circular_NW/SC/Ballistic* describes a circular Si nanowire transistor (diameter d=2.5 nm) with transport along the <111> crystal axis and controlled by an Ω gate configuration, i. e. the gate covers only the angles comprised between -45 and 225 degrees of the nanowire contour. The structure is composed of two 10 nm long n-doped (10^{20} cm^{-3}) source and drain regions, and of a 15 nm long gate, all surrounded by a 1nm thick oxide layer. Some ballistic results (transfer characteristics $I_d - V_{gs}$ at V_{ds}=0.6 V, electron

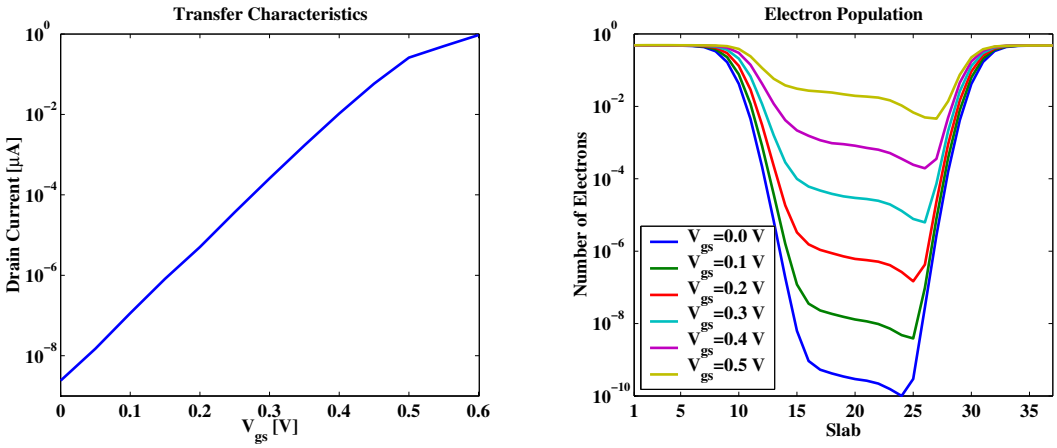


Figure 4.9: Transfer characteristics of the circular nanowire transistor in Fig. 4.6 at V_{ds}=0.6 V (left) and number of electrons per wire unit cell at different gate voltages (right).

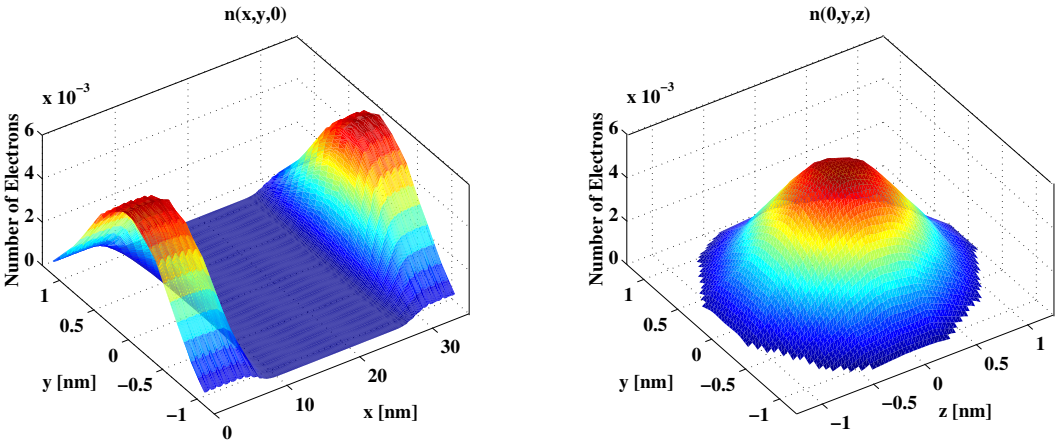


Figure 4.10: Number of electrons in the circular nanowire in Fig. 4.6 on the x-y plane (z=0 nm, left) and on the y-z plane (x=0 nm, right) at V_{gs}=0.0 V and V_{ds}=0.6 V.

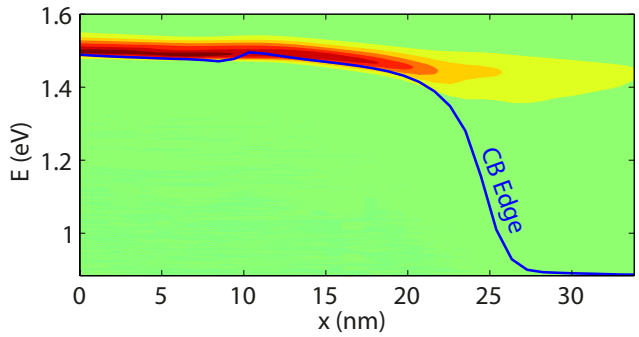


Figure 4.11: Spectral distribution of the current in the circular Si nanowire transistor in Fig. 4.6 at V_{gs}=V_{ds}=0.6 V. The results are shown after 5 Poisson iterations.

population per wire unit cell, and the 2-D surface plots $n(x, y, 0)$ and $n(0, y, z)$) are shown in Fig. 4.9 and 4.10, respectively. The MATLAB function *plot_nw_data.m* can be used to generate similar figures. Note that the calculation of the entire ballistic transfer characteristics shown in Fig. 4.9 takes about 2 hours on 512 CPUs (CRAY XE6).

4.2.5 Simulation with electron-phonon scattering

The same nanowire as in Fig. 4.6 can be simulated in the presence of electron-phonon scattering. Since such simulations are computationally very intensive, it is recommended to calculate only 1-2 bias points with electron-phonon scattering and to use the electrostatic potential from a ballistic simulation as an initial guess. Here, the file *wire_circle_e_ph_scatt.cmd* in the directory *EXAMPLE/Circular_NW/SC/Scattering* shows how to simulate the bias point V_{gs}=V_{ds}=0.6 V starting from the corresponding ballistic electrostatic potential stored in the file *vact_dat*. This simulation should last 12-16 hours on 5200 cores of a CRAY-XE6 system. The spectral current flowing through this nanowire structure at V_{gs}=V_{ds}=0.6 V is given in Fig. 4.11 that can be reproduced with the MATLAB function *plot_data.m*.

4.3 Complex bandstructure

Calculating the complex bandstructure of 1-D, 2-D, and 3-D nanostructures is relevant in applications where intra- or inter-band tunneling plays an important role. In these cases, the WKB approximation can be used to estimate the tunneling probability based on the complex band dispersion of the device under consideration. The file *wire_complex_bandstructure.cmd* in the directory *EXAMPLE/CBS* demonstrates how to compute the complex bandstructure of a Si nanowire with transport along the <100> crystal axis. The simulation should take about 3 minutes on 128 cores. OMEN proceeds in three steps: first, it computes the valence and conduction real bandstructure and extracts the minimum of the conduction band

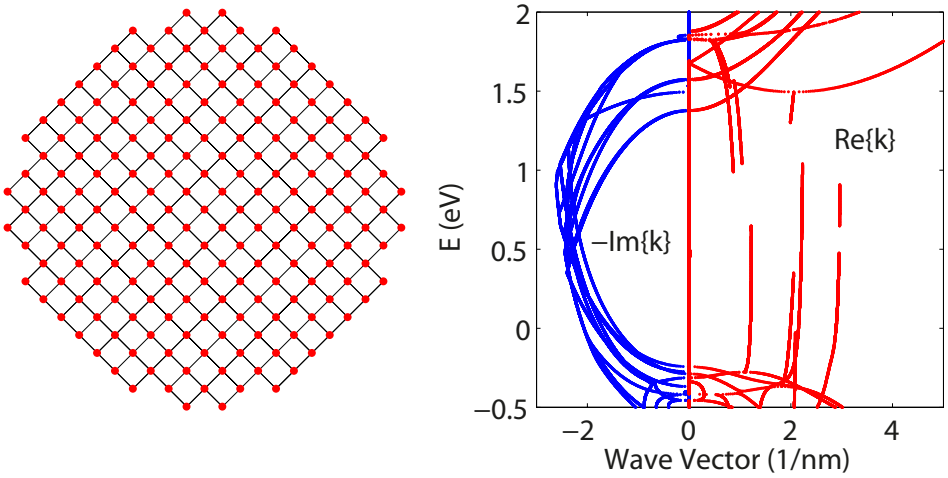


Figure 4.12: (Left) Cross section of a Si nanowire with a circular cross section (diameter d=3 nm) and transport along the <100> crystal axis. (Right) Corresponding complex bandstructure. Blue (red) curves refer to the imaginary (real) components.

and the maximum of the valence band from these data. Then, it creates an energy vector starting at the maximum of VB minus an energy offset defined by the parameter *EOffset* up to the minimum of CB plus an energy offset equal to *EOffset*. The distance between two energy points is determined by the parameter *dE_f*. Finally, for each energy point, OMEN calculates the corresponding wave vector, exactly as it does for the open boundary conditions. The structure simulated in the file *wire_complex_bandstructure.cmd* as well as the resulting complex bandstructure are shown in Fig. 4.12. The two sub-plots were generated by the MATLAB functions *plot_cross_section.m* and *plot_cbs.m*.

4.4 Carbon nanotube FET

OMEN can simulate carbon nanotube field-effect transistors (CNT FETs), as indicated in Fig. 4.13. In the directory *EXAMPLE/CNT*, the files *cnt_1.cmd* and *cnt_2.cmd* can be found to calculate the transfer characteristics I_d-V_{gs} at V_{ds}=0.5 V of a (13,0) carbon nanotube FET whose gate length L_g measures 15 nm, its drain and source extensions L_s=L_d=15 nm as well with a donor doping concentration of 0.54 atom per nm, and whose channel is covered by a 1 nm SiO_2 layer. The file *cnt_1.cmd* is used to generate the stiffness matrix that is required by Poisson equation. It should be run on a single CPU. The parameters to calculate the transfer characteristics are defined in the file *cnt_2.cmd*. As orbital basis, the single p_z orbital model is selected. Hence, this simulation should last about 2.5 hours on 128 CPUs if the basis compression algorithm is employed to solve the Schrödinger equation. Figure 4.13 (right) can be reproduced with the MATLAB file *plot_cnt_data.m*.

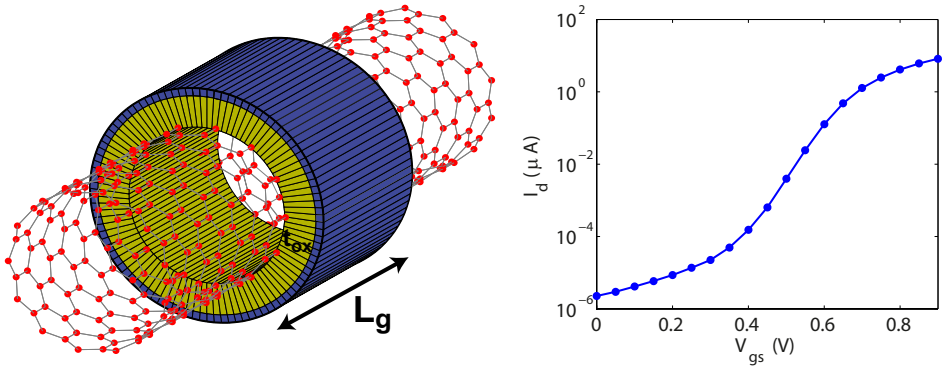


Figure 4.13: (Left) Schematic view of a (13,0) carbon nanotube field-effect transistor (CNT FET) with a gate length L_g=15 nm, source and drain extensions L_s=L_d=15 nm, and surrounded by a 1 nm SiO_2 layer. The doping concentration in the source and drain extension amounts to 0.54 donor per nm. (Right) Transfer characteristics I_d-V_{gs} at V_{ds}=0.5 V of the CNT FET shown on the left.

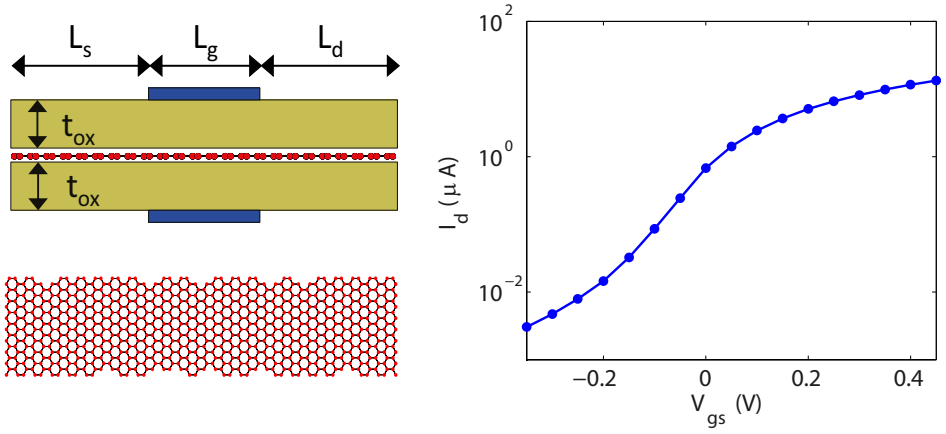


Figure 4.14: (Left) Schematic view of a double-gate armchair graphene nanoribbon field-effect transistor (AGNR FET) with a width w=1.37 nm, a gate length L_g=10 nm, source and drain extensions of length L_s=L_d=10 nm (donor doping concentration N_D=0.28 atom per nm), and 1 nm thick SiO_2 oxide layers. (Right) Transfer characteristics I_d-V_{gs} at V_{ds}=0.4 V of the AGNR FET shown on the left.

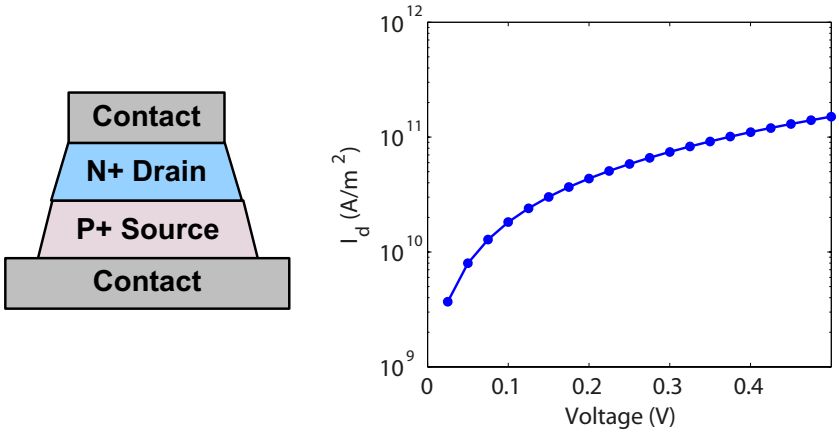


Figure 4.15: (Left) Schematic view of a bulk-like $In_{0.75}Ga_{0.25}As$ band-to-band tunneling (BTBT) diode composed of a heavily p- and n-doped region (N_D=N_A=5e19 cm^{-3}). (Right) Current vs. voltage of the BTBT diode shown on the left.

4.5 Graphene nanoribbon FET

Field-effect transistors based on armchair graphene nanoribbons can also be treated by OMEN. An example *graphene.cmd* is provided in the directory *EXAMPLE/GNR*: the transfer characteristics I_d-V_{gs} at V_{ds}=0.4 V of a 1.37 nm wide graphene nanoribbon FET with a double-gate configuration, as schematized in Fig. 4.14, is computed in the ballistic limit of transport. The gate length is set to 10 nm, while the source and drain extensions measure 10 nm each with an homogeneous donor doping concentration N_D=0.28 atom per nm. The sparse linear solver Umfpack and a $p_z d_{xz} d_{yz}$ tight-binding model with explicit hydrogen passivation of the nanoribbon edges [55] are used. This simulation should roughly last 10 minutes on 12 CPUs. The I-V characteristics reported in Fig. 4.14 can be extracted from the OMEN simulation results with the MATLAB function *plot_gnr_data.m*.

4.6 InGaAs tunneling diode

Band-to-band tunneling (BTBT) diodes also belong to the kind of devices that can be simulated by OMEN. The file *tunnel_diode_ingaas.cmd.cmd* in the directory *EXAMPLE/TD* describes the simulation of a 30 nm long, bulk-like, $In_{0.75}Ga_{0.25}As$ BTBT diode composed of a heavily p- and n-doped region (15 nm each with N_D=N_A=5e19 cm^{-3}), as shown in Fig. 4.15. The directions assumed periodic (y and z) are modeled with N_{k_y}=N_{k_z}=20 momentum points. Electron transport occurs along the <111> crystal axis. The $sp^3d^5s^*$ tight-binding model with spin-orbit coupling is employed as orbital basis. This simulation should last about 4 hours on 640 CPUs using the recursive Green's Function (RGF) algorithm to solve the quantum transport problem. The current vs. voltage characteristics in Fig. 4.15 can

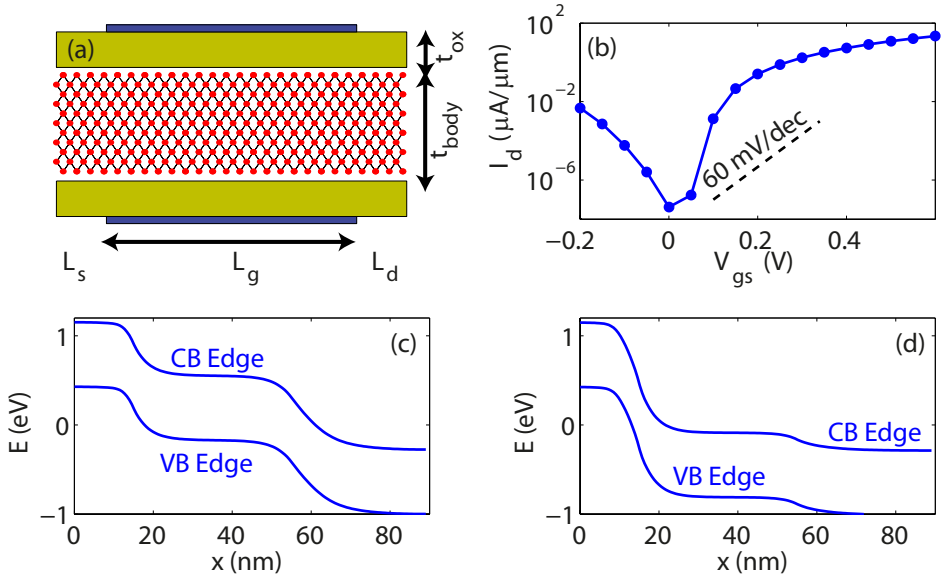


Figure 4.16: (a) Schematic view of a double-gate ultra-thin-body (UTB) InAs band-to-band tunneling field-effect transistor (TFET) with t_{body}=5 nm, L_g=40 nm, L_s=15 nm (N_A=4e19 cm^{-3}), L_d=35 nm (N_D=1e19 cm^{-3}), t_{ox}=4 nm (ϵ_R=20), and transport along the <100> crystal axis. (b) Transfer characteristics I_d-V_{gs} at V_{ds}=0.5 V. (c) Band diagram of the TFET at V_{gs}=0.0 V and V_{ds}=0.5 V. (d) Same as (c), but for V_{gs}=0.6 V and V_{ds}=0.5 V.

be generated with the MATLAB function *plot_td_data.m*.

4.7 InAs ultra-thin-body tunneling FET

Over the years, the capabilities of OMEN have been extended to treat 2-D and 3-D band-to-band tunneling field-effect transistors (TFETs). In the directory *EXAMPLE/TFET*, the files *inas_utb_tfet_1.cmd* and *inas_utb_tfet_2.cmd* allow for the simulation of a double-gate ultra-thin-body InAs TFET. Detailed specifications about the transistor structure can be found in Fig. 4.16. The file *inas_utb_tfet_1.cmd* should be run on a single CPU to create a stiffness matrix for Poisson equation while execution of the file *inas_utb_tfet_2.cmd* should take approximately 2.5 hours on 480 CPUs to compute the InAs TFET transfer characteristics at V_{ds}=0.5 V with the basis compression algorithm, as shown in Fig. 4.16(b). The MATLAB function *plot_tfet_data.m* can be used to generate the band diagrams given in Fig. 4.16(c)-(d).

4.8 CdSe/CdS core/shell quantum dot

OMEN was initially designed to deal with quantum transport situations with open boundary conditions in various types of nanostructures. However, it can also com-

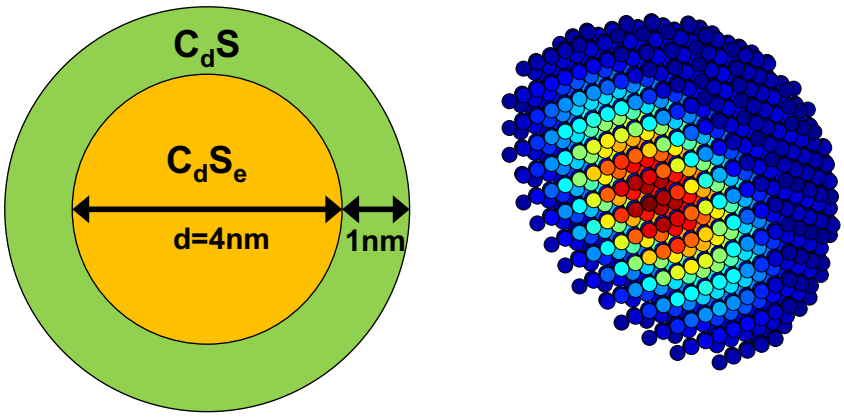


Figure 4.17: (Left) Schematic view of a CdSe-CdS core-shell quantum dot structure. The core material CdSe has a diameter d=4 nm and is surrounded by 1 nm of CdS. (Right) Spatial distribution of the first conduction band mode within the CdSe-CdS quantum dot.

pute the eigenstates of closed systems such as 3-D quantum dots. The file *qdot.cmd* in the directory *EXAMPLE/QDot* describes a CdSe-CdS core-shell quantum dot structure with a core diameter d=4 nm and 1 nm thick layer of CdS all around. Computing 16 conduction and valence sub-bands should take less than 1 minute on a single CPU. To plot the spatial distribution of the different eigenmodes, as in Fig. 4.17, the MATLAB file *plot_qdot_data.m* can be utilized.

4.9 Phonon transport in Si nanowire

As mentioned in Chapter 1, OMEN cannot only treat electron/hole transport, but also thermal (phonon) transport through 1-D, 2-D, and 3-D nanostructures. Here, as an example, thermal transport through a Si nanowire with a diameter d=3 nm, a length L=40 nm, and phonon propagation along the <110> crystal axis is considered, as illustrated in the left part of Fig. 4.18. A temperature difference ΔT=0.1 K is applied between the left (TL=300 K) and right (TR=300.1) contacts of the nanowire.

4.9.1 Ballistic simulation

To simulate thermal transport in its ballistic limit for the nanowire structure in Fig. 4.18, the command file *wire_circle_phonon.cmd* in the directory *EXAMPLE/Phonon/Ballistic* can be used. The execution of this file should take about 3 minutes on 256 CPUs using MUMPs to solve the phonon dynamical equation with open boundary conditions. The position-resolved current is given in the right part of Fig. 4.18. Such a plot can be obtained with the MATLAB file *plot_phonon_data.m*.

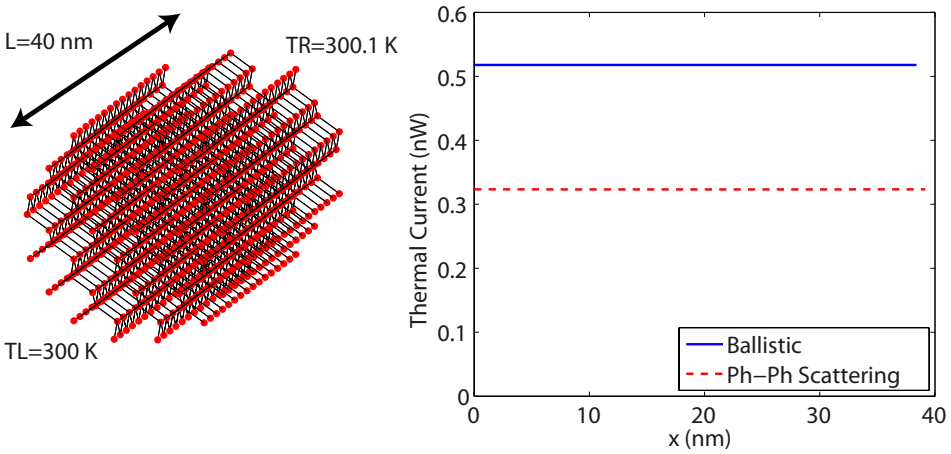


Figure 4.18: (Left) Schematic view of a Si nanowire of length L=40 nm and diameter d=3 nm with transport along the <110> crystal axis. A temperature TL=300 K is applied to the left contact and TR=300.1 K to the right one. (Right) Position-resolve thermal current flowing through the Si nanowire shown on the left, in the ballistic limit of transport (solid blue line) and in the presence of anharmonic phonon-phonon scattering (dashed red line).

4.9.2 Simulation with anharmonic phonon-phonon scattering

The same nanowire structure can be simulated in the presence of anharmonic phonon-phonon scattering using the file *wire_circle_ph_ph_scatt.cmd* in the directory *EXAMPLE/Phonon/Scattering*. Due to the additional self-consistent loop between the phonon Green's Functions and scattering self-energies, such a simulation takes nearly 2.5 hours on 1344 CPUs of a CRAY-XE6 machine. As can be seen in Fig. 4.18, anharmonic phonon-phonon scattering causes a substantial decrease of the thermal current.

Bibliography

[1] W. Fichtner, D. J. Rose, and R. E. Blank, "Semiconductor device simulation", IEEE Trans. on Elec. Dev. **30**, pp. 1018-1030 (1983).

[2] C. S. Rafferty, M. R. Pinto, and R. W. Dutton, "Iterative methods in semiconductor device simulation", IEEE Trans. on Elec. Devices **32**, pp. 2018-2027 (1985).

[3] S. Selberherr, A. Schutz, and H. W. Potzl, "MINIMOS-A two-dimensional MOS transistor analyzer", IEEE Trans. on Elec. Devices **27**, pp. 1540-1550 (1980).

[4] Mathieu Luisier. *Quantum Transport Beyond the Effective Mass Approximation.* PhD thesis, ETHZ, 2007.

[5] M. Luisier, G. Klimeck, A. Schenk, and W. Fichtner, "Atomistic Simulation of Nanowires in the $sp^3d^5s^*$ Tight-Binding Formalism: from Boundary Conditions to Strain Calculations, Phys. Rev. B, **74**, 205323 (2006).

[6] M. Luisier and A. Schenk, "Atomistic Simulation of Nanowire Transistors", J. of Computational and Theoretical Nanoscience **5**, 1031-1045 (2008).

[7] M. Luisier, "Full-Band Quantum Transport in Nanowire Transistors", J. of Comp. Electronics **7**, 309-314 (2008).

[8] M. Luisier and G. Klimeck, "A multi-level parallel simulation approach to electron transport in nano-scale transistors", Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, article 12 (2008).

[9] M. Luisier, A. Schenk, and W. Fichtner, "Three-Dimensional Full-Band Simulations of Si Nanowire Transistors", IEDM Tech. Digest **2006**, 811 (2006).

[10] M. Luisier, A. Schenk, and W. Fichtner, "Atomistic treatment of interface roughness in Si nanowire transistors with different channel orientations", Appl. Phys. Lett. **90**, 102103 (2007).

[11] M. Luisier, A. Schenk, and W. Fichtner, "Full-band atomistic study of source-to-drain tunneling in Si nanowire transistors", Int. Conf. on Simulation of Semiconductor Processes and Devices (SISPAD) Vienna, Austria (2007).

[12] M. Luisier and G. Klimeck, "Full-band and atomistic simulation of n- and p-doped double-gate MOSFETs for the 22nm technology node", Int. Conf. on Simulation of Semiconductor Processes and Devices (SISPAD) Hakone, Japan (2008).

[13] M. Luisier and G. Klimeck, "Full-Band and Atomistic Simulation of Realistic 40 nm InAs HEMT", IEDM Tech. Digest **2008**, 887-890 (2008).

[14] N. Kharche, G. Klimeck, D.-H. Kim, J. A. del Alamo, and M. Luisier, "Performance Analysis of Ultra-Scaled InAs HEMTs", IEDM 2009, Baltimore MD, USA (2009).

[15] M. Luisier and G. Klimeck, "Performance limitations of graphene nanoribbon tunneling FETS due to line edge roughness" 66^{th} Device Research Conference, Penn State University, PA (2009).

[16] M. Luisier and G. Klimeck, "Performance analysis of statistical samples of graphene nanoribbon tunneling transistors with line edge roughness", App. Phys. Lett. **94**, 223505 (2009).

[17] M. Luisier and G. Klimeck, "Atomistic, Full-Band Design Study of InAs Band-to-Band Tunneling Field-Effect Transistors", IEEE Elec. Dev. Lett. **30**, 602-604 (2009).

[18] M. Luisier and G. Klimeck, "Investigation of $In_xGa_{1-x}As$ Ultra-Thin-Body Tunneling FETs using a Full-Band and Atomistic Approach", International Conference on Simulation of Semiconductor Processes and Devices, SISPAD 2009, San Diego CA, USA (2009).

[19] M. Luisier and G. Klimeck, "Performance Comparisons of Tunneling Field-Effect Transistors made of InSb, Carbon, and GaSb-InAs Broken Gap Heterostructures", IEDM 2009, Baltimore MD, USA, (2009).

[20] S. Agarwal, G. Klimeck, and M. Luisier, "Leakage Reduction Design Concepts for Low Power Vertical Tunneling Field-Effect Transistors", IEEE Elec. Dev. Lett. **31**, 621-623 (2010).

[21] M. Luisier and G. Klimeck, "Atomistic full-band simulations of silicon nanowire transistors: Effects of electron-phonon scattering", Phys. Rev. B **80**, 155430 (2009).

[22] M. Luisier and G. Klimeck, "Simulations of Nanowire Tunneling Transistors: from the WKB Approximation to Full-Band Phonon-Assisted Tunneling", J. of App. Phys. **107**, 084507 (2010).

[23] M. Luisier, "Phonon-limited and effective low-field mobility in n- and p-type [100]-, [110]-, and [111]-oriented Si nanowire transistors", App. Phys. Lett. **98**, 032111 (2011).

[24] M. Luisier, "Investigation of thermal transport degradation in rough Si nanowires", J. of App. Phys. **110**, 074510 (2011).

[25] J. C. Slater and G. F. Koster, "Simplified LCAO Method for the Periodic Potential Problem", Phys. Rev. **94**, 1498-1524 (1954).

[26] Z. Sui and I. P. Herman, "Effect of strain on phonons in Si, Ge, and Si/Ge heterostructures", PhysRevB **48**, 17938 (1993).

[27] S. Datta, "Electronic Transport in Mesoscopic Systems", Cambridge University Press (1995).

[28] R. Lake, G. Klimeck, R. C. Bowen, and D. Jovanovic, "Single and multiband modeling of quantum electron transport through layered semiconductor devices", J. of Appl. Phys. **81**, 7845 (1997).

[29] T. A. Davis, "A column pre-ordering strategy for the unsymmetric-pattern multifrontal method", ACM Trans. on Math. Software **30**, 165 (2004).

[30] P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent, "Multifrontal parallel distributed symmetric and unsymmetric solvers" Comput. Methods in Appl. Mech. Eng. **184**, 501 (2000).

[31] X. S. Li and J. W. Demmel "SuperLU_DIST: A Scalable Distributed Memory Sparse Direct Solver for Unsymmetric Linear Systems", ACM Trans. on Math. Software **29**, 110 (2003).

[32] O. Schenk and K. Gärtner, "Solving Unsymmetric Sparse Systems of Linear Equations with PARDISO, Journal of Future Generation Computer Systems", J. of Future Generation Computer Systems **20**, 475 (2004).

[33] T. B. Boykin, M. Luisier, and G. Klimeck, "Multi-band transmission calculations for nanowires using an optimized renormalization method", Phys. Rev. B **77**, 165318 (2008).

[34] M. S. Lundstrom, "On the mobility versus drain current relation for a nanoscale MOSFET", IEEE Elec. Dev. Lett. **22**, 293 (2001).

[35] M. Luisier, "A parallel implementation of electron-phonon scattering in nanoelectronic devices up to 95k cores", Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, (2010).

[36] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard", Parallel Computing **22**, 789 (1996).

[37] M. Luisier and G. Klimeck, "Numerical strategies towards peta-scale simulations of nanoelectronics devices", Parallel Computing **36**, 117-128 (2010).

[38] M. Luisier, T. B. Boykin, G. Klimeck, and W. Fichtner, "Atomistic nanoelectronic device engineering with sustained performances up to 1.44 PFlop/s", Proceedings of the 2011 ACM/IEEE Conference on Supercomputing, (2011).

[39] http://www.cise.ufl.edu/research/sparse/amd/

[40] http://www.netlib.org/

[41] http://www.cs.sandia.gov/CRF/aztec1.html

[42] http://people.sc.fsu.edu/ jburkardt/c_src/metis/metis.html

[43] http://graal.ens-lyon.fr/MUMPS/

[44] http://www.pardiso-project.org/

[45] http://www.qhull.org/

[46] http://crd-legacy.lbl.gov/ xiaoye/SuperLU/

[47] http://www.cise.ufl.edu/research/sparse/UFconfig/

[48] http://www.cise.ufl.edu/research/sparse/umfpack/

[49] S. Lee, F. Oyafuso, P. von Allmen, and G. Klimeck, "Boundary conditions for the electronic structure of finite-extent embedded semiconductor nanostructures", Phys. Rev. B **6**9 045316 (2004).

[50] S. M. Goodnick, D. K. Ferry, C. W. Wilmsen, Z. Liliental, D. Fathy, and O. L. Krivanek, "Surface roughness at the Si(100)-SiO_2 interface", Phys. Rev. B **32**, 8171 (1985).

[51] T. B. Boykin, M. Luisier, M. Salmani-Jelodar, and G. Klimeck, "Strain-induced, off-diagonal, same-atom parameters in empirical tight-binding theory suitable for [110] uniaxial strain applied to a silicon parameterization", Phys. Rev. B **81**, 125202 (2010).

[52] T. B. Boykin, G. Klimeck, R. C. Bowen, and F. Oyafuso, "Diagonal parameter shifts due to nearest-neighbor displacements in empirical tight-binding theory", Phys. Rev. B **66** 125207 (2002).

[53] S. Cauley, V. Balakrishnan, G. Klimeck, and C.-K. Koh, "A two-dimensional domain decomposition technique for the simulation of quantum-scale devices", J. Comput. Phys. **231**, 4 (2012).

[54] S. Cauley, M. Luisier, V. Balakrishnan, G. Klimeck, and C.-K. Koh, "Distributed NEGF Algorithms for the Simulation of Nanoelectronic Devices with Scattering", J. Appl. Phys. **110**, 043713 (2011).

[55] T. B. Boykin, M. Luisier, G. Klimeck, X. Jiang, N. Kharche, Y. Zhou, and S. K. Nayak, "Accurate six-band nearest-neighbor tight-binding model for the π-bands of bulk graphene and graphene nanoribbons", J. Appl. Phys. **109**, 104304 (2011).