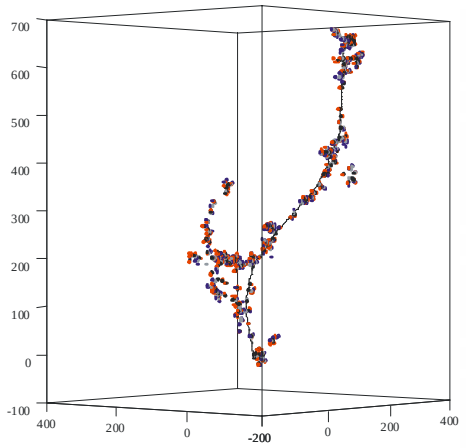
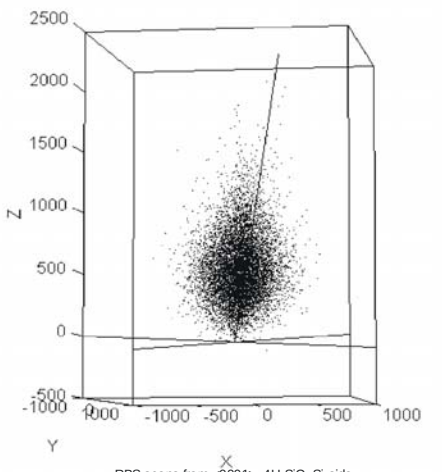
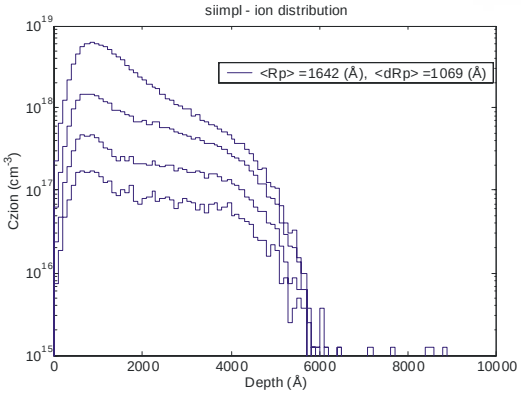
simpl

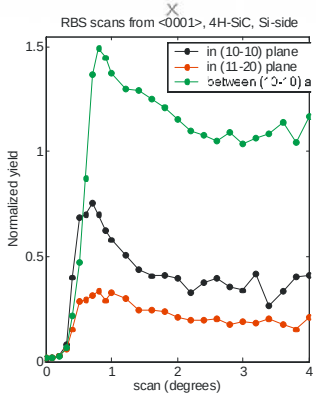
simulation of ion implantation

Users Manual









PREFACE

The SIIMP code (Simulation of Ion IMPLantation) came about from a need of the ion implantation sub-project in the now defunct Swedish silicon carbide program SiCEP. The need was to accurately predict ion and defects profiles, both for device design and for defect studies. This is the reason for why all the examples given at the end of this user manual are related to SiC. However, SIIMPL is not limited to simulation of implantations into SiC, but may in fact be used for any crystalline or amorphous target. The features of SIIMPL may be summarized as follows:

- Easy implementation of any type of crystal or amorphous target
- ZBL interatomic potential, where the scattering integral may be solved using the "Magic" formula or by numerical means
- Versatile electronic stopping model
- Possibility to import SRIM 2003 stopping (MATLAB only)
- New model for thermal vibrations
- Full cascade damage model
- Scattering of implanted ions at implantation induced damage
- Simulation of backscattering experiments through the calculation of Nuclear Scattering Probability distributions
- 3D distributions of ions and defects
- Rare event model for efficient simulations of deeply penetrating ions through channeling
- Sputtering yields

The aims for SIIMPL has been to be a universal and user-friendly tool, rather than focusing on the finer details of a specific ion-target combination.

This users' manual is organized as follows: Section 1 gives a brief general introduction to MC-BCA simulations, while Section 2 describes the models and algorithms, which are used in SIIMPL. Section 3 gives a detailed explanation of how to perform a simulation, and in Section 4, a series of simulation examples are given including the input files and the simulation results they produce. Finally, in Sec 5, some SIIMPL simulations are compared to corresponding experimental SIMS profiles.

To be on the safe side, I have to state that the author does not take any responsibility for any errors that may exist in the code, errors leading to erroneous simulations results. For this reason, the source C code is included with the program for free inspection. That said, it is my sincere hope that the SIIMPL code will be found useful, also outside the "old" SiCEP realm – please feel free to distribute the program to anyone that may have use for it. I further hope that this manual will give a useful description of how MC-BCA simulations in general and SIIMPL in particular functions. If any questions about the program and its functionality should arise, do not hesitate to contact me.

Martin Janson, Hamburg, November 2003

(present e-mail address; msjanson@gmail.com)

CONTENTS

1 INTRODUCTION

The first ion implantation Monte-Carlo simulation codes based on the binary collision approximation were presented already in the early 1950s, but most BCA codes of today stem from the algorithms described by Robinson *et al.* in the mid 1970s [1, 2], i.e., those implemented in the so-called MARLOWE code. A few years later Biersack and Haggmark presented the widely spread TRIM code [3], which was especially designed to decrease the computer execution time. Other reasons for the popularity of the TRIM package (or SRIM, as it is called today) are the inclusion of an extensive library of empirical stopping data for a large number of ion-target combinations [4], and the fact that it is extremely user friendly. However, one of the sacrifices that were made to increase the computer efficiency is that TRIM can only handle amorphous targets, a fact which, in some studies has lead to fatal underestimations of the damage range [5, 6]. Considering the enormous increase in computer power since TRIM was first developed, this rather severe limitation is no longer of need.

Today there exists many different MC-BCA codes for crystalline targets, for example MARLOWE, Crystal-TRIM [7], and KING-IV [8], but many of these are specialized for certain targets and are often not freely available.

1.1 BINARY COLLISION APPROXIMATION (BCA)

An implanted ion looses its energy by displacing the target atoms in elastic collisions and by exciting the target electrons, referred to as nuclear stopping (S_n) and electronic stopping (S_e), respectively. These two processes can in most applications be regarded as being independent of each other and the total stopping cross section is thus the sum of the two, $S = S_n + S_e$.

The detailed nuclear stopping depends on the interatomic potential between the ion *I* and all target atoms:

$$V_I = \sum_j V\left(r, Z_I, Z_j\right) \quad , \tag{1}$$

where $r = \left|\mathbf{r}_I - \mathbf{r}_j\right|$ is the interatomic distance. This means that the ion trajectory can be determined by evaluating the motion of all N atoms in a system, determined by N interatomic potentials as a function of time. This type of calculations, usually referred to as molecular dynamics (MD) simulations, are very time consuming and are today mostly used for studying phenomena related to individual ion-target interactions.

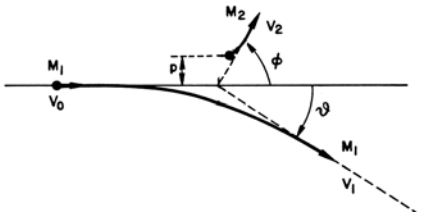


Figure 1 The geometry for a binary collision between an energetic ion M_1 and an initially motionless target atom M_2.

A great simplification is made by only considering the collision between two atoms at a time. This is the so-called binary collision approximation (BCA) and can be shown to be valid for ion energies typically above a few 100 eV [4]. The geometry for a collision between two atoms is shown in Fig. 1. An ion M_1 with velocity v_0 (energy E) is deflected by a target atom M_2, whose position relative the M_1 trajectory is defined by the so-called impact parameter p, which is the smallest distance between M_2 and the trajectory of M_1 before the collision. In the collision, M_1 and M_2 are deflected asymptotically towards trajectories with angles relative to the original M_1 trajectory, ϑ and ϕ, respectively, and the kinetic energy T is transferred from M_1 to M_2. The equations that relate ϑ, ϕ, and T to M_1, M_2, p, E, and $V(r)$ were originally derived by J.J. Thomson in his work on electrical conductivity [9], but are equally relevant to scattering of two atoms:

$$\phi = \int_{r_0}^{\infty} \frac{p\mathrm{d}r}{r^2 g^{1/2}}, \qquad g = 1 - \frac{V(r)}{E_c} - \left(\frac{p}{r}\right)^2, \tag{2a}$$

$$\tan\vartheta = \frac{M_2 \sin 2\phi}{M_1 - M_2 \cos 2\phi}, \tag{2b}$$

$$T = E\frac{4M_1M_2}{(M_1+M_2)^2}\cos^2\phi, \tag{2c}$$

where $E_C = E/(1 + M_1/M_2)$. r_0 is the so-called apsis of collision, which is the smallest interatomic distance r in the collision and can be determined from the equation:

$$g(r_0) = 0. \tag{2d}$$

1.2 MONTE-CARLO BCA SIMULATIONS

In Monte-Carlo binary collision approximation (MC-BCA) simulations, the ion implantation range and defect profiles are determined by simulating a large number of individual trajectories of pseudo ions (pseudo because each ion effectively represents a fraction of the total simulated ion fluence). Each ion is affected by a set of random numbers that determine the start position of the at the sample surface, and the displacements of the target atoms along the ion track due to thermal vibrations.

The MC-BCA algorithm is event driven, which in this case means that the ion trajectory is determined by consecutive binary collisions, as opposed to MD simulations where the ion path is discretized in time. The event driven propagation of a MC-BCA ion trajectory is demonstrated in Fig. 2 (a). An ion at position *I* is heading towards the target atoms *A*, *B*, and *C*. The target atom of the next binary collision is that which is closest to the ion (projected on the ion velocity vector and in front of it) and that has an impact parameter p which is smaller than a certain maximum value p_{max}. In Fig. 2 (a) this is atom *A* since $p > p_{max}$ for atom *C*, and *B* is further away from *I* than *A*. The value of p_{max} should be chosen large enough so that collisions with $p > p_{max}$ have a negligible affect on the ion trajectory, but not too large since the number of collisions per unit path length in a simulation increases as p_{max}^2. Typical values for p_{max} are 2-3 Å. Once the collision partner is found the ion is moved forward a distance Δr to the asymptotic point of the collision *I´* (see also Fig. 1) and the nuclear kinetic energy loss *T*, and scattering angles ϑ and ϕ, are calculated by solving Eq. 2. After subtraction of the electronic energy loss (see Sec. 2.4) the procedure is repeated – the next collision in Fig. 1 (a) would be with atom *B* at *I´´* – until the ion energy falls below some lower limit value E_{stop} where the ion is considered to be stopped. The value of E_{stop} is typically set to a value on the

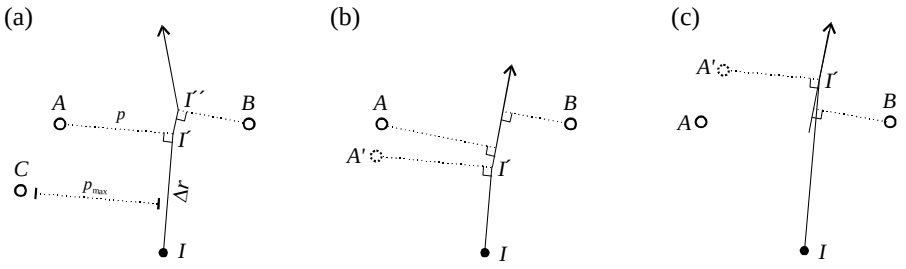


Figure 2 Geometry for the subsequent collisions between an ion *I* and the target atoms *A*, *B*, and *C* in a MC-BCA simulation with a maximum impact parameter of p_{max}. The target atoms remain at their respective lattice positions in (a) while a random displacement has been added to the lattice position of *A after* it has been chosen as a collision partner, bringing *A* to a position *A´* that is closer (b) or further away (c) from *I*.

order of the migration energy for atoms in the target, i.e. a few eV, but is usually not critical for the simulation result. When simulating range distributions, a large number of ions are followed from a randomly chosen position at the target surface until they are stopped. For a target with an ideal lattice, i.e. no thermal vibrations or damage, it is only the start position that involves any random numbers in a simulation.

The bottlenecks of MC-BCA codes are typically the algorithms that search for the next collision partner and the solution of the scattering integral, Eq. (2a).

2 SIIMPL MODELS AND ALGORITHMS

2.1 SEARCH FOR TARGET ATOM ALGORITHM

The heart of any MC-BCA code is the algorithm that searches for the next collision partner. For the code to run be efficiently, the list of potential target atoms that are scanned in each collision event must be as short as possible, but without sacrificing the reliability of finding the correct target atom. In SIIMPL this is accomplished by first dividing the unit cell into smaller cells; from here on referred to as sub cells. In the search process the sub cell that contains the ion is first identified and the search for the next collision partner is then done with the target atoms belonging to that sub cell. The atoms belonging to a sub cell are the atoms that are inside the sub cell volume and the atoms at a distance of up to p_{max} from the sub cell walls. This volume can generally be defined by the sub cell itself [Fig. 3(a)], 6 parallel epideds with a thickness of p_{max} at each side of the subcell [Fig. 3(b)], 12 cylinders with a radius of p_{max}

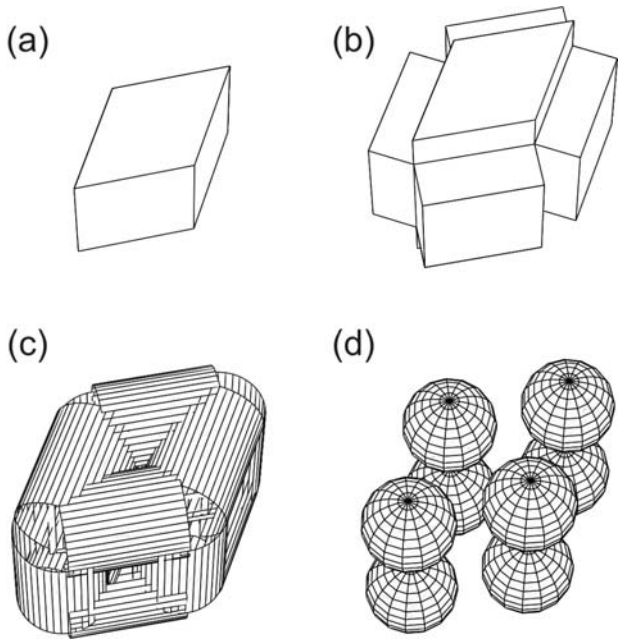


Figure 3 Hexagonal sub cell (a) and volumes (a + b + c + d) that defines which target atoms that belongs to the sub cell. p_{max}, is the radius of the spheres and the cylinders and is here much smaller compared to the sub cell dimensions than in a typical case.

centered on each edge of the sub cell [Fig. 3(c)], and eight spheres with a radius of p_{max} at each corner of the sub cell [Fig. 3(d)]. It should be noted that the p_{max} used for the volumes in Fig. 3, for clarity, is considerably smaller compared to the sub cell dimensions than it would be in a typical simulation. The set of atoms belonging to each sub cell are calculated prior to each simulation and the average number of atoms in each sub cell is written in the SIIMPL Log file.

The number of sub cells that the unit cell is divided into is controlled by the constant *SCsize*, which has the dimension of length. The unit cell is divided in each direction N_1, N_2, and N_3 times, respectively. N_i is calculated as

$$N_i = \text{ceil}(|\boldsymbol{a}_i|/\text{SCsize}), \tag{3}$$

where $|\boldsymbol{a}_i|$ is the length of the unit cell in direction *i*. For example, for a hexagonal unit cell with $|\boldsymbol{a}_1| = |\boldsymbol{a}_2| = 3.07$ Å, and $|\boldsymbol{a}_3| = 10.1$ Å, and with *SCsize* set to 2.0 Å, the unit cell will be divided into 2×2×6 sub cells. *SCsize* thus enables to control the number of atoms in each unit cell, but it should be noted that it is not meaningful to set *SCsize* to a value considerable smaller than p_{max}.

The default value for *SCsize* is 2.0 Å.

2.2 INTERATOMIC POTENTIAL

The current version of SIIMPL uses the so-called universal potential of Ziegler, Biersack and Littmark (ZBL) for all ion-atom, and atom–atom nuclear interactions:

$$V_U = \frac{Z_1 Z_2 e^2}{r} \Phi_U\left(\frac{r}{a_U}\right), \tag{4}$$

where the universal screening function $\Phi_U(x)$ and the screening length $a_U(Z_1, Z_2)$ are fitting formulas [4] given by

$$\Phi_U(x) = 0.18175\exp(-3.1998x) + 0.50986\exp(-0.94229\,x) + \\ + 0.28022\exp(-0.4029\,x) + 0.028171\exp(-0.20162x) \tag{5}$$

and

$$a_U = 0.4685\,/(\,Z_1^{0.23} + Z_1^{0.23}) \quad (\text{Å}). \tag{6}$$

2.3 SOLVING THE SCATTERING INTEGRAL

One of the more time demanding steps in MC-BCA simulations is the solving of the scattering integral, i.e. solving Eq. (2a). In SIIMPL the scattering integral can be solved by two methods: (i) by solving the integral numerically using the Gauss-Mehler quadrature method (with $N = 100$, see Ref. [10]), or (ii) by using the so-called Magic Formula developed by Biersack *et al.* [3, 4]. The Gauss-Mehler method is naturally considerably slower than the Magic-Formula, but is on the other hand more exact. By default, the Magic-Formula is used in SIIMPL.

2.4 ELECTRONIC STOPPING

2.4.1 MC-BCA electronic stopping

In MC-BCA simulations, the electronic energy loss is subtracted from the ion energy in each collision. In SIIMPL this energy loss, ΔE, can either be treated as a non-local stopping force, i.e. $\Delta E = NS_e\Delta r$, as a local energy loss depending on the impact parameter in the collision, or a combination thereof. The energy that is subtracted in each collision is given by

$$\Delta E = S_e\,[(1-f_l)\,N\Delta r + f_l\,A\,\exp(-s\,p/a_U)\,], \qquad (7)$$

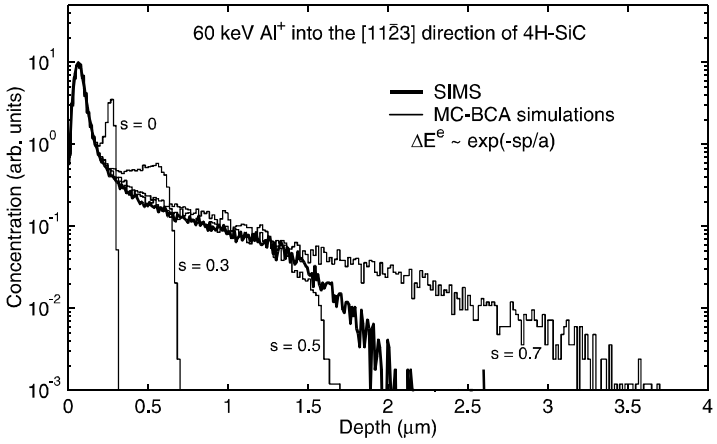


Figure 4 SIIMPL simulations of a 60 keV Al implantation into the [11$\bar{2}$3] direction of 4*H*-SiC. with the *s* parameter of the electronic stopping expression set to 0, 0.3, 0.5, and 0.7 (histograms). The corresponding experimental SIMS profile is included for comparison (solid line).

where f_l is a weighing factor set to a value between 0 and 1. The expression for the local stopping has the form suggested by Oen and Robinson [2] where s is a constant originally set to 0.3, but is here treated as a fitting parameter. A is a normalizing constant, which ensures that the average stopping cross section of Eq. (8) for an ion with a random trajectory is equal to S_e independent on the values of f_l and s:

$$A = s^2/(2\pi a_U^2 [1 - (1 + s\, p_{max}/ a_U) \exp(-s\, p_{max}/a_U)]) . \qquad (8)$$

The default values of f_l and s are 1 and 0.3, respectively. The total average stopping anf therefore the penetration depth of ions with a channeling trajectory is very sensitive on the value of s. This is illustrated in Fig. 4, which depicts the simulated range profiles of a 60 keV Al implantation in the [11$\bar{2}$3] direction of 4*H*-SiC together with the corresponding experimental SIMS profile. One simulation is performed with no impact parameter dependence (s = 0) , and three with s set to 0.3, 0.5, and 0.7, respectively. All simulations give the same projected range for ions that never entered a channel at the surface (the random peak at 0.07 μm), which shows that the stopping for random trajectories is the same independent on s, i.e., the normalization of the parameter A in Eq. (8) has been accurately determined. The differences in penetration depth for the deepest channeled ions are, on the other hand, substantial.

2.4.2 Mean electronic stopping, S_e

The mean electronic stopping S_e is composed of a low ion velocity (S_e^{lo}) and a high velocity (S_e^{hi}) contribution. The total stopping is calculated using the interpolation formula proposed by by Biersack and Haggmark [3]

$$S_e = \left(S_e^{lo\,-1} + S_e^{hi\,-1} \right)^{-1} . \qquad (9)$$

The low velocity stopping is modeled according to

$$S_e^{lo} = A_1 E^{\,p + f\,p_2}$$
$$f = \exp\left(-3 \left(\frac{\ln E_F}{\ln E} \right)^4 \right), \qquad (10)$$

where E is the ion energy, and E_F the Fermi energy, both in keV. (Both E and E_F should strictly speaking be divided by (1 keV) in Eq. (10) to yield a dimension of A_1 that is independent of ($p + f\,p_2$) .) The inclusion of $f\,p_2$ in the exponent of S_e^{lo} has been introduced to account for the increased stopping at energies above E_F due to the stripping of the electrons off the ion. f has been chosen to give a relatively fast but smooth transition between $S_e^{lo} \propto E^p$ below, and $S_e^{lo} \propto E^{p+p_2}$, above E_F.

The default values are $A_1 = k_{LS}$, $p = 0.5$, $p_2 = 0$, and $E_F/M_1 = 25$ keV/amu, where k_{LS} is the Lindhard-Scharff prefactor given by:

$$k_{LS} = 3.8454 \frac{Z_1^{7/6} Z_2}{M_1^{1/2} (Z_1^{2/3} + Z_2^{2/3})^{3/2}} \quad \left(\text{eV}/(10^{15}\ \text{cm}^{-2})\right) \tag{11}$$

NOTE that Eq. (11) is only valid when E is given in keV in Eq. (10). For compound materials the default value for A_1 is determined according to Bragg's rule:

$$k_{LS} = \sum_i w_i k_{LS,i} \ . \tag{12}$$

where w_i is the fraction of atom i.

The high velocity electronic stopping uses the parameterization proposed by Ziegler [11]:

$$S_e^{hi} = A_3/(E/1000) \log[\ 1 + A_4/(E/1000) + A_5\ (E/1000)\ ] \tag{13}$$

where E is given in keV.

The default values of the A_3–A_5 parameters are those obtained from the Bethe-Bloch theory as described in Ref. [3], and are here rewritten to the form of Eq. (13)

$$A_3 = 0.238\ Z_1^2\ Z_2^{eff}\ M_1 \quad \text{keV eV}/(10^{15}\text{cm}^{-2}) \tag{14a}$$

$$A_4 = 4.55\times10^{-4}\ C\ I\ M_1 \qquad \text{keV} \tag{14b}$$

$$A_5 = 2.20\times10^{3}\ /\ I\ M_1 \qquad \text{keV}^{-1} \tag{14c}$$

where Z_2^{eff} is defined as the weighted average Z_2 of all atoms in the unit cell:

$$Z_2^{eff} = \sum_i w_i Z_{2i} \tag{15}$$

I is the mean ionization energy and has been proposed to be given by [3]

$$\begin{aligned} I &= Z_2^{eff}\left[12 + 7\left(Z_2^{eff}\right)^{-1}\right] \ \text{eV}, && for \quad Z_2^{eff} < 13 \\ I &= Z_2^{eff}\left[9.76 + 58.5\left(Z_2^{eff}\right)^{-1.19}\right] \ \text{eV}, && for \quad Z_2^{eff} \geq 13 \end{aligned} \tag{14}$$

I should be given in eV in Eqs. (14b) and (14c). Finally C is a dimensionless empirical correction parameter set to [3]:

$$\begin{aligned} C &= 100 Z_1 / Z_2^{eff}, && for \quad Z_1 < 3 \\ C &= 5, && for \quad Z_1 \geq 3 \end{aligned} \tag{15}$$

M_1 is given in units of amu in all above equations.

2.4.3 Importing SRIM 2003 electronic stopping

A special MATLAB function has been written that imports the electronic stopping for any target used by the SRIM 2003 package (via the SRIM program "*SRmodule.exe*") and translates it to the parameters used in SIIMPL. The function is called "*Se_srim2siimpl.m*" and is located in the directory *Se_SRIM* of the SIIMPL package. The following example (MATLAB commands) demonstrates how to import the S_e for He in SiC as predicted by SRIM:

```
Z1 = 2;
M1 = 4.0;
Z2 = [6 14];
M2 = [12 28];
N = [1 1];
Ef = 25 * M1;
```

Note that the target atom numbers Z2, weight M2, and stochiometry N are given as row vectors. The following MATLAB command

```
[A1, p, p2, A3, A4, A5] = Se_srim2siimpl(Z1, M1, Z2, M2, N, Ef);
```

then generates the SIIMPL stopping parameters that gives the best fit of the SIIMPL S_e model to the SRIM 2003 data. In the above case, the following output should be obtained:

```
A1 =  3.1186
p =   0.5096
p2 =  0.0405
A3 =  37.8877
A4 =  2.4099
A5 =  4.5551
```

2.5 THERMAL VIBRATIONS

Thermal vibrations of the target atoms are a major factor for the dechanneling of channeled ions and should therefore be included in MC-BCA simulations of crystalline targets. The most commonly used model for thermal vibrations in MC-BCA simulations is that of the MARLOWE code. In that model the vibrations of the atoms are treated in such a way that the collision partner is first located with the target atoms at their equilibrium positions. A random 3D-displacement with a Gaussian distributed amplitude and a standard deviation of $\sqrt{3}\,u_1$ is thereafter added to the lattice position, after which the impact parameter and point of collision are recalculated. (u_1 is the one dimensional mean vibration amplitude: $u_1^2 = \langle x^2 \rangle$.) However, this treatment brings problems that may be difficult to circumvent. Figure 2 (b) illustrates a situation where the thermal displacement has brought the atom from lattice position *A*

to a position *A*', closer to the ion. This means that if the scattering angle is small, the ion, now at *I´*, may collide with atom *A* a second time. This effect was, however, taken into account and could be avoided [1]. What was not considered is the situation where the thermal displacement brings the atom further away from the ion [Fig. 2 (c)], which also gives rise to severe problems. In this case, when the ion is moved to the position *I´*, there is a probability that the collision with atom *B* never occurs, irrespective of the impact parameters for the collisions. For a channeled ion, i.e. with a trajectory almost parallel to the atomic rows, the probability for this to happen is substantial since the projections on the ion path of the lattice positions of two atoms at opposing sides of the channel then can be very close. This is for example the case for the atoms *A* and *B* in Fig. 2. The reason why this has not been previously noted is most likely due to the fact that this artifact is – more or less – cancelled by the simultaneous collision model, further elaborated in Sec. 2.6.

Unfortunately, there is no easy way to solve this latter problem. Instead, SIIMPL uses a new algorithm for thermal vibrations where the displacement is added to the lattice positions first, prior to the search for the collision partners. Naturally, the displacements can then not be randomly chosen in each collision, since the atom positions have to be the same for the same atom position each time it is evaluated, to avoid the above mentioned problems. However, by replacing the random numbers by an oscillating function that is a function of the laboratory lattice position $|\mathbf{r}|$ only, seemingly uncorrelated lattice vibrations can be accomplished. The wave length of this function has to be very short relative to the distance between neighboring atoms. In this way the problems of the MARLOWE model are avoided since the positions of the atoms are not changed after a collision partner has been found. Appendix A presents the algorithm that SIIMPL uses to produces these "quasi random" numbers as a function of $|\mathbf{r}|$.

Besides from this new algorithm for thermal vibration, it is also possible to use the MARLOWE algorithm in SIIMPL. The default setting is the 'SIIMPL' algorithm however.

By default, u_1 is set to 0.05 Å for all target atoms.

2.6 SIMULATANEOUS COLLISIONS MODEL

This simultaneous collision algorithm was introduced in the MARLOWE code since Robinson and Torrens suggested that the standard BCA treatment may introduce instability for ions with a channeling trajectory, if the distance between subsequent binary collisions is too small. Instead they proposed a quasi many-body treatment of the problem where the momentum transfer in each of these binary collisions should be linearly subtracted from the ion momentum and the ion moved directly to the most distant of these

"simultaneous" collisions [1]. However, this point has never been properly investigated, although it would be possible with today's computational resources using MD-simulations.

The simultaneous collision (SC) algorithm has been included as an option in SIIMPL, both for the SIIMPL and the MARLOWE thermal vibration models. The free parameter in the SC model is the maximum distance ΔR_{SC} between two consecutive collisions for the collisions to be treated as simultaneous. It should be noted that all collision within ΔR_{SC} are treated as simultaneous.

NOTE In the current version of SIIMPL, the recoiling energy will not be correct using the SC algorithm. This is due to the "unphysical" treatment of a many body collision, which can be compensated for although this is not implemented yet. The effect is relatively small, but it is strongly advisable NOT to use the SC model when the damage profile is of interest. The magnitude of this error can be viewed by the error given in the energy check, written in the SiiMP Log file.

The SC model is turned off by default. The default value for ΔR_{SC} is 0.2 Å.

2.6.1 Test of thermal vibration and simultaneous collision models

One consequence of the simultaneous collision model is the cancellation of the above mentioned neglect of collisions due to the treatment of the thermal vibrations. The effect of the missing collisions in the MARLOWE thermal vibration algorithm and the role of the simultaneous collision algorithm are illustrated in Fig. 5, which shows the simulated range profiles of a 100 keV implantation into the [001] direction of a single cubic test target with a lattice parameter of 2 Å. Both the ion and target atoms are of a test species X, which has been defined with atomic number 15 and mass 30 amu. The simulations

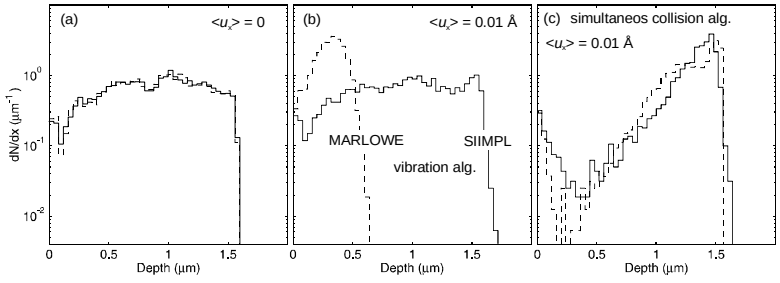


Figure 5 Two sets of simulations of 100 keV X ions (Z = 15 and M = 30) into the <001> directions of a single cubic test target X, using the MARLOWE (broken lines) and SIIMPL (solid lines) algorithms for thermal vibrations. No vibrations are included in (a) while the mean atomic 1D-displacement $<u_x>$ is set to 0.01 Å in (b) and (c). For the simulations in (c) the simultaneous collision algorithm has been used. Both sets of simulations were executed using the SIIMPL code.

using the two different thermal vibration algorithms are both executed using the SIIMPL code. In Fig. 5(a) simulations are performed with a frozen, non-vibrating lattice and the profiles of the MARLOWE and SIIMPL algorithms are practically identical. However, when a very small vibration is included, u_1 = 0.01 Å, [Fig. 5(b)] the simulated profile using the MARLOWE vibration algorithm becomes considerably more shallow, while the SIIMPL profile does not differ significantly from the simulations of a frozen lattice. The SIIMPL result seems to make more sense considering the small amplitude of the vibrations. Figure 5 (c) shows the simulated profiles with the same small lattice vibration as in Fig. 5(b), but now also including the simultaneous collision algorithm in both simulations. Again both algorithms produce very similar profiles, but with even less dechanneling compared to the frozen lattice simulations in Fig. 5(a). These results clearly demonstrate that the MARLOWE treatment of thermal vibrations leads to a significantly overestimated dechanneling rate and that this artifact is cancelled by the inclusion of the simultaneous collision algorithm. As a consequence, these results also put some major question marks around the actual role of the simultaneous collisions algorithm in MC-BCA simulations.

2.7 AMOURPHOUS TARGETS AND SURFACE LAYERS

Although SIIMPL was primarily written for crystalline targets, it is also possible to treat the entire target, or a surface layer as being amorphous.

For an amorphous target, the position of the next collision partner is randomly chosen in a cylinder in front of the ion, which has a radius p_{max} and a length

$$L_{cyl} = 2/(\pi p_{max}^2 natomamorph), \qquad (16)$$

where *natomamorph* is the atomic density of the amorphous material. The type of atom is randomly chosen among the atoms defined in the unit cell, weighted by their relative abundance, w_i. This means that it in the present version of SIIMPL is only possible to include an amorphous surface layer of the same material as the crystalline target. It is, however, possible to define a different atomic density for the amorphous material compared to that of the crystalline target.

natomamorph is by default set to the same density as the crystalline target.

2.8 DAMAGE

2.8.1 Defect distributions

The calculation of ballistic damage profiles, i.e. of mono vacancies and interstitial Frenkel pairs produced in the binary collisions is a relatively straight forward procedure in MC-BCA simulations. If the energy transferred from the ion to the target atom in a collision T exceeds the displacement energy E_d for that species, the ion trajectory of the recoiling secondary atom is treated in the same way as the implanted, primary ion. Since there is no time dependence in the MC-BCA scheme and hence no correlation between the recoils, the trajectories of the secondary ions and the subsequent generations of recoils that they may produce, can be performed one at a time after the primary ion trajectory has been completed. This method is usually referred to as the full damage cascade algorithm.

A simpler way to determine damage profiles is the Kinchin-Pease (KP) model, which states that the interstitial and vacancy concentration profiles are proportional by a factor of $0.4/E_d$ to the distribution of energy deposited in nuclear collisions by the primary ion [4]. This is obviously a much faster method since the recoiling trajectories are not simulated, but it is less accurate especially for compound targets, where the KP model only registers defects for the same type of target atoms as the one taking part in the binary collision.

The default setting for SIIMPL is the Kinchin-Pease model, but it should be noted that when detailed damage profiles are the primary interest then it is advisable to use full damage cascade calculations. This is especially true for compound targets.

The default settings are E_d = 15 eV for all target atoms.

2.8.2 Damage build-up, dose effects

If the crystal damage reaches high enough concentrations during an implantation, the trajectories of the subsequently implanted ions may be seriously affected. This dose-effect can be seen as a saturation of channeling tails in ion range profiles since atoms displaced by the implantation (in the following called interstitials) gradually "fill" the channels of the crystal and hence enhance the dechanneling rate. Another example where implantation induced damage affect ion trajectories is of course in RBS analysis in the channeling mode.

A simplistic way to include scattering at defects in the crytsal in MC-BCA simulations is to evaluate the probability that the next collision will be a collision with an implantation induced interstitial atom. This probability P is often [7, 12] evaluated as

$$P = c_a N_I / N \,, \tag{17}$$

where N_I is the local density of interstitials provided by the simulation, c_a is a fitting constant, and N the atomic density of the crystal. A random number then decides if the collision should be with an interstitial instead of a lattice atom. If it is with an interstitial atom then the collision is treated as a collision in an amorphous target (see Sec. 2.7), but with the atomic density set to the density of the crystalline material, N.

A careful examination of Eq. (17) reveals that it is, strictly speaking, only valid for ions with a non-channeled trajectory. This is a rather severe limitation since the model was introduced to account for the dechanneling of channeled ions. SIIMPL introduces a correction term to Eq. (17) that annihilates this problem, described in detail in Appendix B.

This version of SIIMPL makes no difference between different types of damage when evaluating P, and N_I at a certain depth z below the surface is simply evaluated as

$$N_I(z) = \sum_i C_{Ii}(z) \tag{18}$$

where $C_{I\,i}$ is the concentration of interstitials of target atom type i at z.

The default setting for the damage scattering model is $c_a = 0$.

2.9 SPUTTERING

In the full cascade damage mode, SIIMPL keeps track of the number of recoiling target atoms that leave the sample, i.e. are sputtered. When a recoiling atom reaches the surface it will only be considered as sputtered if it has a kinetic energy that surmounts the surface barrier, U_a.

By default U_a is set to 5 eV. The fluence of sputtered target atoms is given in the SIIMPL Log file (see Sec. 3.3.3).

2.10 RARE EVENT, RARE DEPTH ALGORITHM

When implanting medium to heavy ions in crystals using a low symmetry, "random" direction, a small amount of the implanted ions will be scattered into some major channel and may penetrate deep into the material. In the ion range profiles this is typically observed as deep, low concentration tails. To accurately simulate these implantations, a very large number of pseudo ions need to be calculated since the probability for this deep penetration is small. One way to overcome exhausting execution times in this case is the use of the

rare event, rare depth algorithm [13]. In this model, one or several depths are defined at the deeper end of the expected mean projected range of the ion distribution. When a simulated ion passes one of these (rare event) depths it will be multiplied, or cloned to several ions. All parameters carried by the initial ion (type, energy, direction) are copied to the cloned ions, except for the pseudo dose carried by the ions, which is divided by a factor equal to the cloning number to ensure a correct normalization. (It should be noted that all distributions recorded by SIIMPL are internally normalized by the ion pseudo dose in order to give a correct result also with the rare depth algorithm.) Each of the cloned ions is then simulated from the cloning point, and the thermal vibration model will ensure that they do not obtain identical ion trajectories. This implies that the rare event algorithm **does not work** when no thermal vibrations are considered.

NOTE The rare event algorithm can be very effective in saving execution time, but it should be used with some care since it can be difficult to determine if the simulated profile is statistically correct. Without the rare event algorithm this is not the case since the trustable part of the simulated distribution usually can be determined from the "background" noise signal.

To obtain a good result with the rare depth algorithm, the first (most shallow) depth have to be chosen at a depth where the profile would have been statistically correct without the rare event algorithm. It is usually not meaningful to define more than 2-3 rare depths.

The rare event algorithm is by default turned off, and the default value for the rare event multiple factor is *rareeventmult* = 10.

2.11 NUCLEAR ENCOUNTER PROBABILITY (NEP)

SIIMPL can also be used to simulate RBS experiments, although it does not directly provide RBS spectra. Instead the distribution of the so-called Nuclear Encounter Probability (NEP) factor is evaluated from which RBS spectra can be extracted [14]. Note that the NEP is only evaluated for the target atoms, which means that this function is mainly of interest for simulating RBS-channeling experiments. The NEP for target atom type i is registered as a function of depth according to

$$NEP_i(z) = \frac{A}{2\pi u_{1i}^{\,2} dz} \left\langle \exp\left(-\frac{p_0^{\,2}}{2u_{1i}^{\,2}} \right) \right\rangle \tag{19}$$

where p_0 is the impact parameter in the collisions relative the *equilibrium position* of the target atom (i.e. the position without thermal vibrations). NOTE that one NEP profile is generated for each type of target atom type. $NEP(z)$ is the

mean value for all collisions in the target in the depth interval ($z \pm dz/2$). A is the mean atomic surface density of the target, i.e. $A = N^{2/3}$ for crystalline material and $A = natomamorph^{2/3}$ for amorphous targets.

NOTE that *NEP* is normalized by dz in SIIMPL.

3 RUNNING SIIMPL

The simulation code SIIMPL is an executable Windows file *siimpl.exe* that has to be executed from a Windows command window or some higher level platform that allows passing commands to the operating system, e.g. MATLAB. All information to and from SIIMPL are in the form of ASCII files. This chapter describes how SIIMP is executed and gives a detailed description of the input and output data files. At the present state no graphical post processing of the simulation results are included at a Windows command level. However, several MATLAB functions that import and plot SIIMPL results have been written and are briefly described at the end of the chapter.

3.1 EXECUTION

To start a SIIMPL simulation the program *siimpl.exe* is executed from a Windows Command window, or some higher platform that allows invoking operating system commands. (In Windows XP a command window can be opened by Start -> Accessories -> Command Prompt.) The simulation ASCII input file is passed to SIIMPL by typing its name directly after the *siimpl* command. The following example runs SIIMPL from a Windows command window in the directory *D:\simulations*, which contains the SIIMPL input file *sim1.txt*:

```
D:\simulations\>D:\siimpl\siimpl  sim1.txt
```

(the extension.*exe* need not be included). In this example it is assumed that the program *siimpl.exe* is placed in the directory *D:\siimpl*. From MATLAB the corresponding command would read:

```
!D:\siimpl\siimpl  sim1.txt;
```

3.2 INPUT DATA

3.2.1 General

All information that specifies a simulation is provided in the form of ASCII text files. These files contain SIIMPL commands and instructions described in detail below. The instructions can be written in any order (with one exception, see Sec. 3.2.4.1) and can be repeated several times where the last statement overwrites the previous ones. In general one instruction is composed by one command, often followed by on or more numerical values, on a single line. The exception to this rule is the unit cell definitions, which requires several lines for each instruction (see Sec. 3.2.4.1).

An important feature of the input file structure is that all instructions do not need to be included explicitly in the file that is passed to SIIMPL. By using the *readsimfile* command it is possible to include other ASCII files into the present simulation. This is especially useful for defining the crystal structure and for defining the electronic stopping data, which usually only has to be specified once for each target and ion-target combination, respectively. For example, the SIIMPL instruction:

```
readsimfile D:\siimpl\targets\SiC\4HSiC.txt
```

includes the text file *4HSiC.txt* from the *D:\siimpl\targets\SiC* directory to the current simulation. The included file can in turn contain one or several *readsimfile* commands, and so on.

For commands that are followed by numerical input values, the line has to be ended by ";". Strictly speaking this is only necessary when SIIMPL can not know how many numbers follow the command, but it is a good habit to write a ";" after the final numerical number on each line. Those instructions which do not require numerical numbers should *not* be ended by ";".

Lines that begin with either "%" or "$" are ignored by the program in order to enable commenting text in the input files. Empty lines are allowed.

NOTE that the instructions are case sensitive and should be written exactly as defined below.

The SIIMPL commands and instructions described in this chapter will be explained by various examples. These examples sometimes include commenting text (after a % sign). This text is naturally not needed for the execution of SIIMPL, but is included here to help the reader.

To further clarify how the instructions should be used, several examples of full simulation input files are given in Sec. 4.

3.2.2 Units

The units for various entities in SIIMPL should always be given according to

- Length Å
- Mass Atomic Mass Units (amu)
- Energy keV
- Stopping Power[1] eV/(10^{15} atoms cm^{-2})
- Concentration cm^{-3}
- Fluence (or dose) cm^{-2}
- Rotation angles °, degrees (360° = 2π radians)

3.2.3 Coordinate systems

SIIMPL uses one single Euclidean coordinate system (x,y,z) to which all positions and directions are related. This coordinate system will also be referred to as the laboratory system. The exception to this rule is the definition of the atom positions in the unit cell, which are only given using the unit cell base (see Sec. 3.2.4.1). Some directions may also be defined using the unit cell base. If nothing else is specified points, vectors, and directions are given in the laboratory coordinate system.

3.2.4 Target Properties

3.2.4.1 Unit Cell

The unit cell is described by the unit cell base and the unit cell atoms, position and type. The following example defines the unit cell base, i.e. its three lattice vectors (a_1–a_3), for 4H-SiC

```
UCbase
3.0730      0           0;          %  = a1
-1.5365     2.6613      0;          %  = a2
0           0           10.0518;    %  = a3
```

[1] The SIIMPL internal unit for stopping power is keVÅ^2 , but this is only of interested for those aiming to exam the SIIMPL source code.

NOTE that the definition of the unit cell base is the only case in SIIMPL where the order of the instructions is of importance. The unit cell base has to be defined prior to the definition of the unit cell atom positions and also prior to any direction declarations that are related to the unit cell base.

The types of atoms that are included in the unit cell are defined separately and are assigned a number related to the sequence that they are given. The following example defines the target atomic numbers (Z2) and mass (M2) of the atoms for SiC:

```
Z2    6     14;
M2    12.0  28.0;
```

In this example carbon is assigned the number 1 and silicon 2. The number of atom types is simply given by the number of statements following Z2 and M2, which naturally have to be of the same length.

The position of the atoms in the unit cell is given by the *UCatoms* command. The type of atom and its position is then given by four numbers on a single line, one for each atom. The first number is an integer related to the type of atom defined above, and the following three real numbers define the atomic position.

NOTE that the positions of the atoms in the unit cell are always given relative the unit cell base (a_1, a_2, a_3) and not in the Euclidean, laboratory space coordinates.

After the last atom is defined the command *end* should be given on a separate line. The following example defines the atoms of 4H-SiC with the unit cell base given above

```
UCatoms
2    0.6667    0.3333         0;
1         0         0    0.0625;
2         0         0    0.2500;
1    0.3333    0.6667    0.3125;
2    0.3333    0.6667    0.5000;
1         0         0    0.5625;
2         0         0    0.7500;
1    0.6667    0.3333    0.8125;
end
```

The unit cell described above will in the following examples be related to as the 4H-SiC unit cell.

3.2.4.2 Target surface and thickness

The surface of the target is defined by a point belonging to the surface R_{surf} and the surface normal vector. The following instruction defines the surface to cross (0,0,0)

```
Rsurf 0 0 0;
```

, which represents the default value. For a typical simulation R_{surf} does not have to be explicitly given.

The surface normal may be defined either as a vector in the laboratory system:

```
surfnorm  1.5365  -2.6613     0;
```

or in the unit cell base

```
surfnormUC 0 -1 0;
```

Both examples above give the same result for the 4H-SiC unit cell base. (The coordinates in *surfnormUC* do not need to be integers.)

The surface normal may also be defined by a special instruction that relates the surface normal to a rotation of a unit vector $\mathbf{e}_z$, initially parallel to the laboratory +z direction, first around the laboratory y-axis, and then around the laboratory z-axis, according to the right hand rule. The following instruction gives the surface normal as 8° off the [000-1] direction towards a <11-20> direction for the 4H-SiC definitions above:

```
surfnormrotyrotz 188 0;
```

(The hexagonal four digit direction definitions used above are *not* allowed in SIIMPL.)

The thickness of the target is normally defined for the sake of the discretization of the depth distribution. There are, however other reason to define a certain thickness of the target. For example when the target in fact is very thin (compared to the range of the implanted ions) and in RBS simulation where only NEP information from the surface layer, is of interest. A target thickness of 1 μm is defined according to

```
layerwidth 10000;  % (Å)
```

3.2.4.3 Amorphous surface layer and target

The thickness and atomic density of an amorphous surface layer (described in Sec. 2.7) are given by the instructions *randsurflayer* and *natomamorph*, respectively.

To define an amorphous surface layer of 10 Å with a density of 8.4×10^{22} cm^{-3}, the following instructions are given

```
randsurflayer 10; % (Å)
natomamorph 8.4e22;  % (cm-3)
```

In the current version of SIIMPL it is only possible to use an amorphous layer with the same atoms and stochiometry as for the crystalline target.

To define the entire target as amorphous either define the surface layer as the same thickness as the target or simply use the following instruction

```
randomtarget
```

3.2.4.4 Thermal vibration amplitudes and surface energy

The 1D thermal vibration amplitudes of the target atoms (*vibu1*) have to be given for each type of atom in the unit cell. For the case of the 4H-SiC unit cell defined above, the following statement would assign a 0.057 Å and 0.051 Å amplitude for the C and Si atoms, respectively:

```
vibu1       0.057       0.051;      % (Å)
```

The surface energy used in the sputtering model is defined according to:

```
surfaceenergy  5e-3; % (keV)
```

which sets the surface barrier for sputtering to 5 eV.

3.2.5 Implantation Properties

The implantation is defined by the implanted species (atomic number Z_1 and mass M_1), the implantation energy (E_0), dose, and direction. To the implantation properties also belongs the implanted surface area, which is mainly of interest when 3D distributions are recorded. To define a 100 keV Al implantation with a dose (or fluence) of 1.5×10^{14} cm^{-2} the following four instructions would be used (not necessarily in this order):

```
Z1    13;
M1    27;          % (Å)
E0    100;         % (keV)
dose  1.5e14;      % (cm-2)
```

The implanted direction is defined in the same way as the surface normal, see Sec. 3.2.4.2. The implantation direction instructions related to the laboratory system, the unit cell base, and the rotation of $\mathbf{e}_z$ are *impdir*, *imdirUC*, and

impdirrotyrotz. The following three examples give implantation directions normal to the corresponding sample surface definitions in Sec. 3.2.4.2:

```
impdir            -1.5365  2.6613  0;     (Å)
impdirUC          0  1  0;
impdirrotyrotz    8  0; (degrees)
```

SIIMPL allows some control of the surface area that is being implanted. This area is defined by a point at the surface $R_{implant}$ and two perpendicular vectors in the plane of the surface. The implanted area is the rectangle formed by the two vectors originating from $R_{implant}$. In the current version of SIIMPL it is only possible to define the length of these vectors while SIIMPL determines their directions. The exact 3D coordinates of the implanted rectangle are given in the SIIMPL output log file. To define a squared implanted area with a side of 100 nm and originating from (0,0,0) the following instructions would be given:

```
Rimplant 0 0 0; % (Å)
implantarea 1000 1000; % (Å*Å)
```

which also are the SIIMPL default values. For most cases where only 1D distributions are of interest, the defined surface area need not to be explicitly defined. It should be stated that the size of the implanted area has no connection to the specified implantation dose used for the 1D distributions.

NOTE that the definition of a too small implanted area (in the order of one unit cell) can result statistical problems. Furthermore, the implanted area and $R_{implant}$ should not be defined too large (larger than the ~1 mm) since this could lead to problems for the sequence of quasi random numbers in the thermal vibration model (Sec. 2.5), which relies on the difference in absolute ion position **R** between consecutive binary collisions.

3.2.6 Electronic Stopping Power, S_e

The parameters in the electronic stopping model described in Sec. 2.4 should be explicitly given for each ion-target combination, since the SIIMPL default values given by the "classical" linear models may result in large errors (>50%) in the predicted ion range. It is important to understand that SIIMPL in itself, in contrast to the TRIM package, do not include any empirical correction terms for the stopping powers. However, the empirical S_e parameters for many ions in SiC have been included in the form of SIIMPL input files. Furthermore, a special MATLAB program has been written that import the electronic stopping used in the SRIM 2003 code, to the parameters used by the SIIMPL S_e model.

3.2.6.1 Defining ESP parameters

The electronic stopping power parameters may be defined for the implanted ion, and optionally also for the atoms in the target. This latter option, which is used for the recoil trajectories in the full cascade damage model, could be considered to be somewhat unnecessary since the energy of the recoiling atoms is such that the nuclear stopping is considerably greater than the electronic stopping power for a clear majority of the recoils. However, for the calculation of shifted interstitial and vacancy profiles it may be of some importance since this shift only depends on the few recoils that travel a long distance from the point where the Frenkel pair was created. When the S_e parameters for the target atoms are not explicitly defined, S_e for the recoiling atoms is evaluated using the default values given in Sec. 2.4. For most simulation conditions, S_e does not have to be explicitly given for the target atoms.

The following example shows how the S_e parameters are defined for the primary ion only:

```
% Low velocity Se parameters
Efermi 35; % (keV/amu)

esp.A1      13.4;       % (eV/(10^15 cm-2)
esp.p       0.327;
esp.p2      0.11;

% Hi velocity Se parameters
esp.A3      9000;       %  (keVeV/(10^15 cm-2))
esp.A4      7;          %  (keV^-1)
esp.A5      0.5;        %  (keV)

% Loacal fraction fl and local stopping parameter s
esp.fl      1.0;
esp.s       0.45;
```

The low velocity parameter p may also be defined using:

```
esp.A2      0.327;
```

With this notation the SIIMPL S_e stopping model, with corresponding parameters (A_1-A_5), is identical to that of Ziegler et al., often used to describe S_e for light atoms (see also example 4.7). **NOTE,** that p_2 is automatically set to 0 when the esp.A2 statement is used (unless the *esp.p2* command is used after *esp.A2*).

There is also an alternative way to define the low velocity S_e pre-factor, that gives A_1 relative to the Lindhard-Scharff pre-factor of Eq. (11): $A_1 = c\, k_{LS}$. For a 20% increased stopping relative S_e^{LS}, this correction factor is defined according to:

esp.c 1.2;

The stopping power parameters (all above except E_{fermi}) for the target atoms are given after the primary ion parameters on the same line and in the order as that defining the target atoms (Sec. 3.2.4.1). The following example defines the low velocity stopping parameters for P in SiC, including the target atoms:

```
Efermi 35; % (keV/amu)

%           ion         C           Si
% ------------------------------------
esp.A1      3.9         5.5         4.0;
esp.p       0.52        0.45        0.52;
esp.p2      0.11        -0.05       0;
```

where the C and Si S_e data in this case where taken to be identical to B and Al , respectively. In this example the hi-velocity stopping parameters are the default values given in Sec.2.4.2.

3.2.7 Damage Model

The primary physical entity in the damage model is the displacement energy *Edispl*, which has to be given for each type of atom in the unit cell. The following example shows how to assign 20 eV and 35 eV displacement energies for the C and Si atoms, respectively, of the 4H-SiC unit cell defined above:

```
Edispl      20e-3       35e-3;      % (keV)
```

The default damage model in SIIMPL is that of Kinchin and Pease. The example below illustrates how to switch on the full cascade damage model

```
damage.cascade    1;
```

The instruction

```
damage.cascade    0;
```

overrides any previous statement, and makes sure that the Kinchin-Pease model is used in the simulation. The damage scattering parameter c_a is defined by the *damage.c_a* statement, and is here assigned a value of 0.5:

```
damage.c_a        0.5;
```

3.2.8 MC-BCA parameters

3.2.8.1 BCA parameters

A fundamental parameter in the BCA is the maximum impact parameter, p_{max}, which by default is set to 2.0 Å. To change this value to for example 1.5 Å the following instruction is used:

```
pmax  1.5;  % (Å)
```

The E_{stop} parameter defines the energy at which the ion is considered to have stopped. The default value is E_{stop} = 2 eV. The following instruction sets E_{stop} to 10 eV:

```
Estop 10e-3;       % (Å)
```

3.2.8.2 Sub-Cell size

Another parameter that may be important for the execution time of the simulations is the *SCsize* parameter, which controls the size of the sub cells described in Sec. 2.1. The size of the sub cells can of course not be the exact size given by *SCsize* , since an integer number of sub cells precisely have to fill the unit cell. The size of the sub cells is thus chosen to be the greatest one *up to SCsize* that just fill the unit cell. The defaul value is *SCsize* = 2 Å. To set *SCsize* to 1.5 Å, write the instruction

```
SCsize       1.5;  % (Å)
```

3.2.8.3 Method for solving the scattering integral

SIIMPL provides two methods for solving the scattering integral: the Magic formula (default) or the Gauss-Mehler method, as described in Sec. 2.3. The method used is controlled by the *IAPfunction* instruction according to:

```
IAPfunction 0; % = Magic formula
IAPfunction 1; % = Gauss-Mehler
```

3.2.8.4 Simultaneous collision model

By default SIIMPL does not consider so-called simultaneous collisions (SC), for reasons described in Sec. 2.6. However, the SC algorithm has been implemented in SIIMPL as an option and is activated by the command:

```
simultcollision
```

The SC parameter ΔR_{SC} is controlled by:

```
dRsc 0.2; % (Å)
```

which also is the SIIMPL default value.

3.2.8.5 Monte-Carlo parameters

An important parameter in MC simulations is the number of simulated pseudo ions, N_{ions}. This number is set by the *Nions* command and is here assigned a value of 100;

```
Nions 100; % number of pseudo ions
```

The default value is N_{ions} = 10000.

Normally, the sequence of random numbers generated in SIIMPL is identical in each simulation (described in Appendix C), which means that two simulations with identical input data will yield identical results. However, in order to get a different random number sequence in different runs the *randomize* instruction should be given:

```
randomize
```

3.2.8.6 Rare depth model

The random depth model described in Sec. 2.10 is switched on by defining the position(s) of the rare depths using the *raredepth* instruction. SIIMPL allows the definition of up to 5 rare depths, which should be given in increasing order after the *raredepth* command. To define two rare depths at 0.1 and 0.5 μm type:

```
raredepth  1000 5000;  % Å
```

The number of splitting ions at a rare depth event is defined by the *rareeventmult* command. The following example sets the number of splits to 10:

```
rareeventmult 10;
```

which is the default value.

3.2.9 Importing ion and defect profiles from previous simulations

Generally the start condition for a simulation is with the ion and defect profiles set to 0. It is, however, possible to use a different start condition by importing ion and/ or defect profiles from a previous simulation. With this feature it is possible to simulate multiple implantations as well as a c-RBS spectra of

implanted samples. SIIMPL only handles one type of impurity at a time, which means that if different ions are implanted only the defect profiles should be imported to the simulation. The commands are *importCzion*, *importCzI*, and *importCzV*, which should be followed by the full path of the directory containing the corresponding *Czion.txt*, *CzI.txt*, and *CzV.txt* files. The following example imports the interstitial and vacancy distribution from a simulation stored in the directory *D:\siimpl\simulations\example1*

```
importCzV D:\siimpl\simulations\example1
importCzI D:\siimpl\simulations\example1
```

It goes without saying that the depth scale discretization (layerwidth and Ndistz) of the imported simulation has to be equal to that of the simulation to be launched.

SIIMPL has a special feature for importing profiles from the last simulation are located in the present directory. The following commands import the ion and defect profiles from the previous simulation:

```
importCzion current
importCzI current
importCzV current
```

With the import commands it is also possible to import arbitrary profiles to the simulation. One example would be a start condition with a homogenous concentration of interstitials.

3.2.10 Output control

3.2.10.1 1-D distributions

SIIMPL generates several 1-D distributions in form of histograms (see Sec. 3.3.1). The depth resolution dz of these distributions are determined by the target thickness L and the number of bins N_{distz} according to $dz = L/ N_{distz}$.. where $N_{distz} = 200$ is defined according to:

```
Ndistz 100;
```

The default value for is $N_{distz} = 200$.

When the damage scatter model is used it can sometimes be interesting to monitor the dose dependence in the simulation. The *savatdose* command allows for saving the profile(s) when a certain dose is reached in the simulation. The following example demonstrated the *saveatdose* command when the distributions of a 1×10^{15} cm^{-2} implant also are saved at doses of 1×10^{14} and 3×10^{14} cm^{-2}:

```
saveatdose 1e14 3e14; % cm-2
```

Note that the value of the final dose is not included since it is always saved.

3.2.10.2 3-D distributions

SIIMPL can also save the 3-D positions of each simulated ion as well as of the produced vacancies and interstitials (only with the full damage recoil model). To save the 3-D distribution of the ions, type:

```
saveionsR3
```

The corresponding instruction for the vacancies and recoils reads:

```
saveIVR3
```

NOTE that saverecoilsR3 may produce very large output files when many pseudo ions are simulated.

3.3 OUTPUT DATA

3.3.1 1-D distributions

Each SIIMPL simulation produces several 1-D depth distributions in the form of ASCII files. These text files are produced in the same directory from which *siimpl.exe* is executed and always have the same names:

- Czion.txt Ion concentration
- CzI.txt Interstitial concentration
- CzV.txt Vacancy concentration
- QTz.txt Deposited electronic and nuclear energy density
- NEP.txt Nuclear Encounter Probability

The files contain two or more columns of length N_{distz} (see Sec. 3.2.10.1) where the first column in all files is the depth (Å) normal to the surface. The depth variables represent the center of the bins in the histograms. For example, with a target thickness of 1000 Å and with N_{distz} = 100, the first column would be 5, 15, 25,…, 995 Å.

In the *Czion.txt* file, the second column represents the concentration of the implanted ion (cm^{-3}), while *CzI.txt* and *CzV.txt* contain a number of columns corresponding to the number of atom species in the target. The interstitial and vacancy distributions for each species are given in the same order as they are defined in the unit cell (see Sec. 3.2.4.1).

The *QTz.txt* file first gives the energy ($keV/Å/cm^2$) deposited in the electronic system *Q* then the energy deposited to the target atom nuclei *T*. When the Kinchin-Pease damage model is used, *Q* and *T* represent the total electronic and nuclear energy loss, respectively, of the primary ion.

The *NEP.txt* file gives the Nuclear Encounter Probability profile. Similarly to the defect files, one column is given for each target atom species.

In case that the output data are saved at different doses (*saveatdose* command), the profiles are written sequentially in each file.

3.3.2 3-D distributions

Three 3-D distribution files are optionally created in SIIMPL (see Sec. 3.2.10.2)

- ionR3.txt Ion distribution

- IR3.txt Interstitial distribution
- VR3.txt Vacancy distribution

In the *ionR3.txt* file, the first three columns represents the position (x, y, z) of the starting point at the surface for each simulated ion, while the last three columns represents the corresponding rest positions. In the defect files, four columns are given. The first column represents the identity of the defect according to the number given in the definition of the target atom species, i.e. in the Z2 command. The final three columns give the position (x,y,z) of the corresponding interstitials (or vacancies).

3.3.3 Log file

SIIMPL generates a log-file in each simulation, *siimplLog.txt*. Besides from information given in the input file, the log-file contains the following information:

- The number of Sub-Cells and the total and average number of atoms in the Sub-Cells.
- The explicit parameters of the S_e model for the primary ion.
- The 3D-coordinates of the implanted rectangular area.
- The total number of binary collisions, including and excluding any recoil trajectories.
- The total number of events when no collision partner was found in the present Sub-Cell.[2]
- The mean projected range of the primary ions R_p and corresponding mean path, R_{path}.
- The mean distance between collisions for the primary ion $<dR_0>$, and the corresponding expected value for a random target, *Random* dR_0 (see Sec. 2.8.2).
- The doses of backscattered, and transmitted primary ions, and sputtered target atoms.
- An energy test: $E_{out}/E_{in} - 1$, where E_{out} is the sum of the deposited nuclear (phonon) and electrinic energies, and the kinetic energies of the

[2] This can happen for small values of p_{max} and SC_{size}. When this happen the ion is forwarded a small distance until a collision partner is found. As long as the total number of no-found events are small this should not significantly affect the simulation result.

backscattered, transmitted and sputtered ions. E_{in} is the product of the implantation energy and dose.

For an example of a SIIMPL log-file, see Sec. X.

3.3.4 MATLAB plot routines

SIIMPL includes several MATLAB scripts for visualization of the simulations results. These scripts, which are located in the *\siimpl\plot* directory of the SIIMPL program package are the following:

- plotCzionFile 1-D histogram of ion distribution
- plotCzIVFile 1-D histograms of interstitials and vacancies
- plotQTzFile 1-D histograms of deposited electronic and Nuclear energy density
- plotNepFile 1-D histograms of NEP
- plotIonR3File 3-D ion distribution (absolute position including implanted surface rectangle)
- plotionR3_0_File 3D ion distribution (positions relative start position including the implantation direction)
- plotIVR3File 3D vacancy and interstitial distributions

In order to use the plot files efficiently it is a good idea to add the *\siimpl\plot* directory to the MATLAB path. This can be accomplished by running the *siimpl_init.m* script in the *\siimpl* directory.

4 EXAMPLES

This chapter gives a few examples of various SIIMPL input files and, in some cases the simulation results they produce. If not explicitly given, all examples may be found in the directories *\siimpl\examples*. The file names are given on the first line of each input file. Some of these SIIMPL input files have the extension ".m", which is the extension for the MATLAB script and function files. The reason for this is that if the MATLAB editor is used, then will the commenting text (starting with a "%") be colored in green for easier reading. From MATLAB, the simulations can be executed using the *siimpl.m* script included in each directory.

4.1 DEFINITION OF 4H-SIC CRYSTAL

The following file defines the 4H-SiC crystal so that the Si-side surface normal = [0 0 -1] = $-\mathbf{e}_z$. The surface of the crystal is defined to be 8 ° tilted off the (Si-side) c-axis towards a <11-20> direction, in accordance with the 4H-SiC wafers currently provided by Cree Inc. Thermal vibration values (300 K) as well as typical displacement energies are also defined here.

```
% File Name: UC_4HSiC.txt

% --- Unit cell base:
UCbase
3.0730      0           0;          %  = a1
-1.5365     2.6613      0;          %  = a2
0           0           10.0518;    %  = a3


% --- Unit Cell atoms

% atom      1           2
% ---------------------------
Z2          6           14;
M2          12.000000   28.000000;


% Hexagonal positions in (0001) plane
% A=[0.6667    0.3333 ];
% B=[0         0];
% C=[0.3333    0.6667];
%
% 4H-SiC = A B C B
```

```
UCatoms
2    0.6667    0.3333         0
1         0         0    0.0625
2         0         0    0.2500
1    0.3333    0.6667    0.3125
2    0.3333    0.6667    0.5000
1         0         0    0.5625
2         0         0    0.7500
1    0.6667    0.3333    0.8125
end

% With these definitions, and assuming that the
% hexagonal c-axis is parallel to the Si-side
% surface normal:
% ex || [100] UCbase    (eq. <11-20>hex)
% ey || [120] UCbase    (eq. <10-10>hex)
% ez || [001] UCbase    (eq. [000-1]hex)

% Amorphous atomic density
natomamorph 8.4e22; % (cm-3)

% --- Define surface normal 8 deg off c-axistowards
% a <11-20> direction (for example || ex)
surfnormrotyrotz 188 0; % (rot, tilt) (degrees)

% Displacement energies and thermal vibrations
% atom      1           2
% ----------------------------------------
Edispl      20e-3       35e-3       % (keV)
vibu1       0.057       0.051 % (Å)
```

4.2 AL IMPLANTATION IN 4H-SIC

This example demonstrates a 60 keV 1×10^{14} cm^{-2} Al implantation into 4H-SiC with the implantation direction anti parallel to the (Cree) wafer surface defined above. The simulation is performed with the full cascade damage model.

First the electronic stopping is defined separately in its own file (in the *\siimpl\Se\SiC* folder)

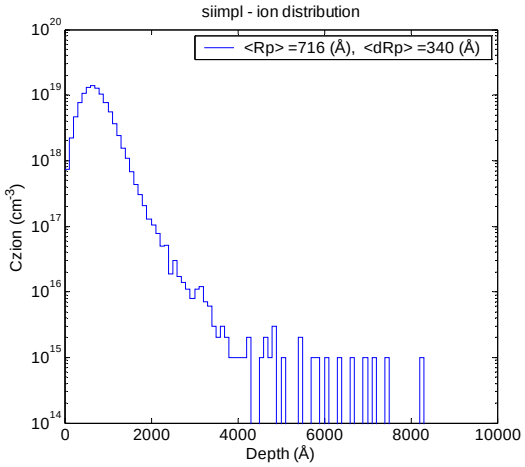
```
% File name: Se_Al_SiC.m
% Experimental stopping data from M.S. Janson et al. to be published
%
% C and Si stopping are the ones for B and Al, respectively
% Only low-velocity parameters are defined here
```

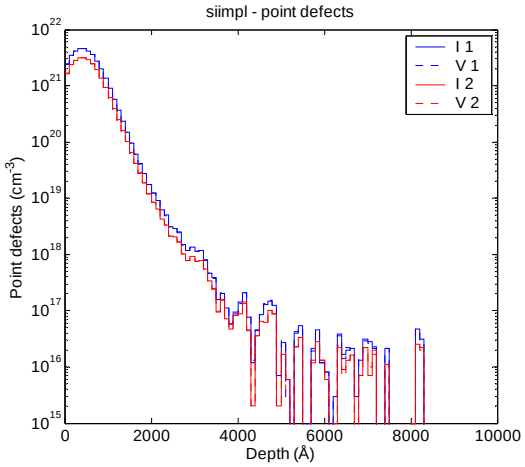
```
% ------- Electronic Stopping Data -------
Efermi 35; % (keV/amu)

%           ion         C           Si
% ---------------------------------------
esp.A1      4.0   5.5   4.0;
esp.p       0.52  0.45  0.52;
esp.p2      0     -0.05 0;

esp.fl      1.0   1.0   1.0;
esp.s       0.52  0.4   0.5;

% NOTE. The carbon atoms must earlier have been defined as
% atom #1, and Silicon as atom #2 (in Z2 instruction)
```

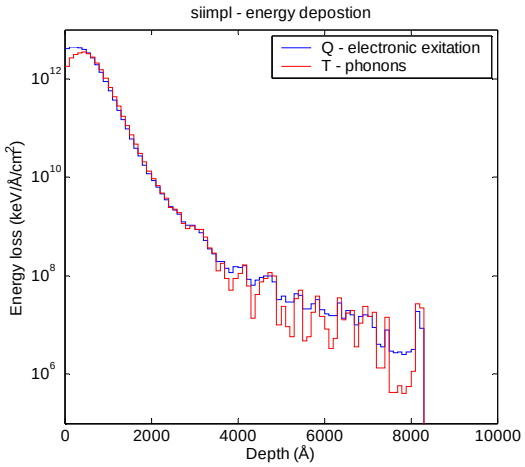
The simulation file can then be written as follows:

```
% File name: 60keV_Al_in_SiC.m

% Crystal, target data
readsimfile D:\siimpl\crystals\UC_4HSiC.txt

% Electronic stopping
readsimfile D:\siimpl\Se\SiC\Se_Al_SiC.m

% --- Implanted ion data  --------
Z1    13;
M1    27; % (amu)
E0    60; % (keV)

dose    1e14; % (cm-2)

% Implantation direction
impdirrotyrotz    8 0; %  (degrees)

% --- Damage model ------
damage.cascade 1;

% --- MC statistics ---------
Nions       100000; % number of pseudo ions

layerwidth  10000; % (Å)
Ndistz      50;
```

By using the MATLAB plot scripts *plotCzionFile, plotCzIVFile, and plotQTzFile* the following graphs are produced from this simulation







The SIIMPL log file produced from this simulation is given in Appendix D.

If this implantation instead would have been performed normal to the surface of a (11-20) 4H-SiC wafer, the surface normal of the crystal and the implantation direction would have had to be modified. In this case the simplest way is by using the UCbase directions:

```
surfnormUC  -3 -3 0;
impdirUC     3  3 0;
```

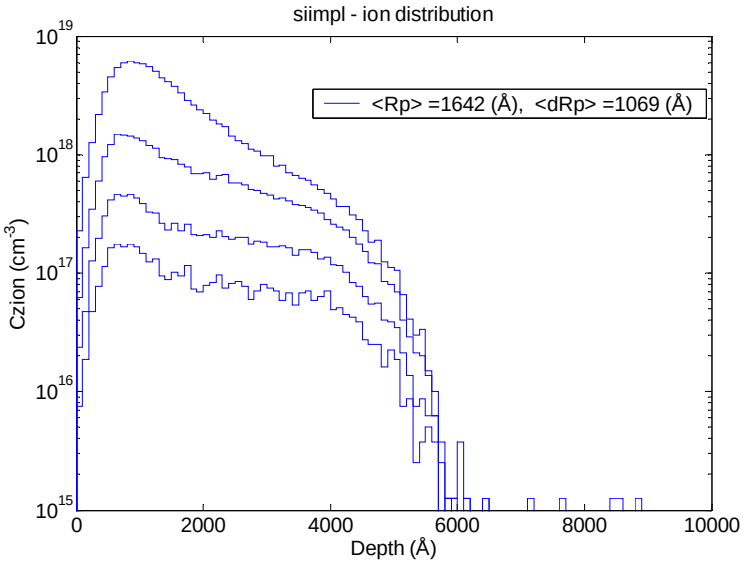
remembering that the hexagonal direction [00-20] is identical to [1100]. (The above statements are of course identical to the UCbase directions -1-10 and 110, respectively.) Note that the surface redefinition does not have to be made in the *UC_4HSiC.txt* but can be written in the above input file as long as it is done *after* any previous surface definitions.

4.3 DOSE DEPENDENCE OF [0001] AL IMPLANTATION

The following example demonstrates the use of the dose dependence model in SIIMPL. The simulation is of a 60 keV 1×10^{14} cm^{-2} in the [0001] direction of a (Cree) 4H-SiC crystal. An amorphous surface layer of 8 Å is used here and the damage scattering constant c_a is set to 1.2. To monitor the dose dependence, the 1-D distributions are saved at doses of 4×10^{12}, 1×10^{13}, and 3×10^{13} cm^{-2}.

```
% File name: 60keV_Al_0001.m

% Crystal, target data
readsimfile D:\siimpl\crystals\UC_4HSiC.txt

randsurflayer 8;  % (Å)

% Electronic stopping
readsimfile D:\siimpl\Se\SiC\Se_Al_SiC.m

% --- Implanted ion data  --------
Z1        13;
M1    27; % (amu)
E0        60; % (keV)

dose    1e14; % (cm-2)

% Implantation direction
impdirUC 0 0 1;

% --- Damage model ------
damage.cascade 1;
damage.c_a  1.2;
```

```
% --- MC statistics ---------
Nions 80000; % number of pseudo ions

layerwidth 10000; % (Å)
Ndistz 100; % division of layer width

saveatdose  4e12  1e13 3e13;
```

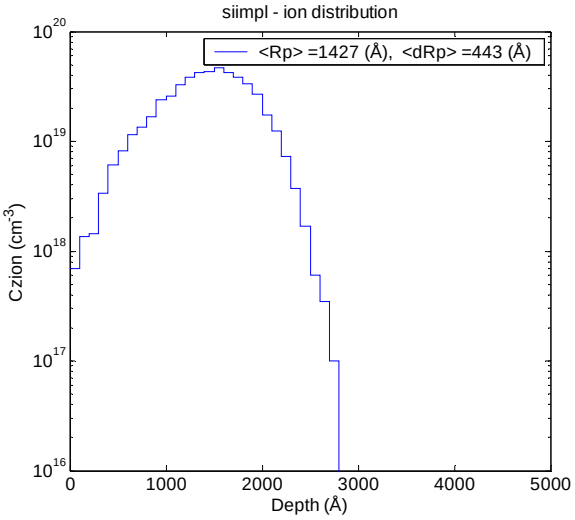


4.4 AMORHPOUS TARGET

The following example demonstrates the input file and the produced profiles of a 115 keV, 5×10^{14} cm^{-2} Al implantation into amorphous SiC. The corresponding experimental profile is included for comparison.

```
% File name: 115keV_Al_in_SiC.m

% Crystal, target data
readsimfile D:\siimpl\crystals\UC_4HSiC.txt

randomtarget

% Electronic stopping
readsimfile D:\siimpl\Se\SiC\Se_Al_SiC.m

% --- Implanted ion data  --------
```

```
Z1     13;
M1     27; % (amu)
E0     115; % (keV)

dose     5e14; % (cm-2)

% Implantation direction
impdirrotyrotz     8 0; %  (degrees)

% --- Damage model ------
damage.cascade 0;

% --- MC statistics ---------
Nions 5000; % number of pseudo ions

layerwidth 10000; % (Å)
Ndistz 50; % division of layer width
```

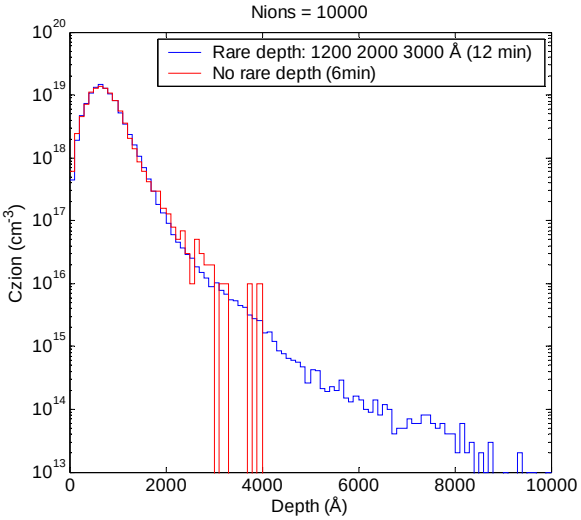


4.5 RARE DEPTH ALGORITHM

The following example demonstrates the usefulness of the rare depth algorithm. The implantation is the same as in Example 4.2, but with 10000 (originally) pseudo ions and with the Kinchin-Pease damage model. One simulation is made without any rare depths and one (file below) with rare depths defined at 1200, 2000, and 3000 Å. Note that the input simulation file makes use of the input file already written for Example 4.2. The execution time for the rare depth simulation was in this case only twice that of the standard

one. (Also compare to ion profile in Example 4.2., which used 100000 pseudo ions.)

```
% File name:  Rare_depth.m

readsimfile D:\siimpl\examples\60keV_Al_4H_SiC\60keV_Al_in_SiC.m

raredepth   1200    2000    3000; % (Å)
rareeventmult 10;

damage.cascade  0;

Nions 10000;
```



(The extension of the tail of the Rare depth simulation may a first glance seem too long by comparing to the [0001] simulation above. However, the deep tail originates from scattering into the [11-23] channel, see Ref. [15].)

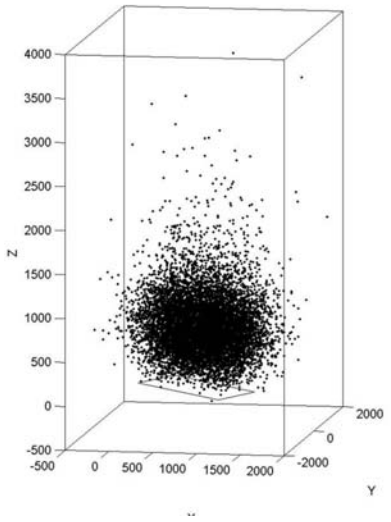
4.6 3D DISTRIBUTIONS OF AL IMPLANTATION

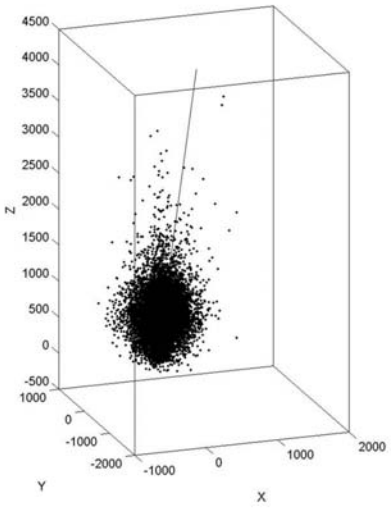
The following example produces the 3D positions of 10000 60 keV Al ions implanted over a surface 1000×1000 Å^2. Note that the specified implantation dose and the "dose" obtained from the number of (pseudo) ions and the implanted area are usually not the same.

```
% File name: 3D_ion.m
```

```
% Crystal, target data
readsimfile D:\siimpl\crystals\UC_4HSiC.txt

% Electronic stopping
readsimfile D:\siimpl\Se\SiC\Se_Al_SiC.m

% --- Implanted ion data  --------
Z1        13;
M1    27; % (amu)
E0        60; % (keV)

dose    1; % (cm-2);

% Implantation direction
impdirrotyrotz    8 0; %  (degrees)

Rimplant 0 0 0; % (Å)
implantarea 1000 1000; % (Å*Å)


% --- Damage model ------
damage.cascade 0;

% --- MC statistics ---------
Nions 10000; % number of pseudo ions

layerwidth 10000; % (Å)
Ndistz 100; % division of layer width

saveionsR3
```

The two graphs below were generated by the SIIMPL MATLAB plot-functions *plotionR3File* (absolute positions of ions including implanted rectangle) and *plotionR3_0_File* (positions relative start position including the implantation direction)

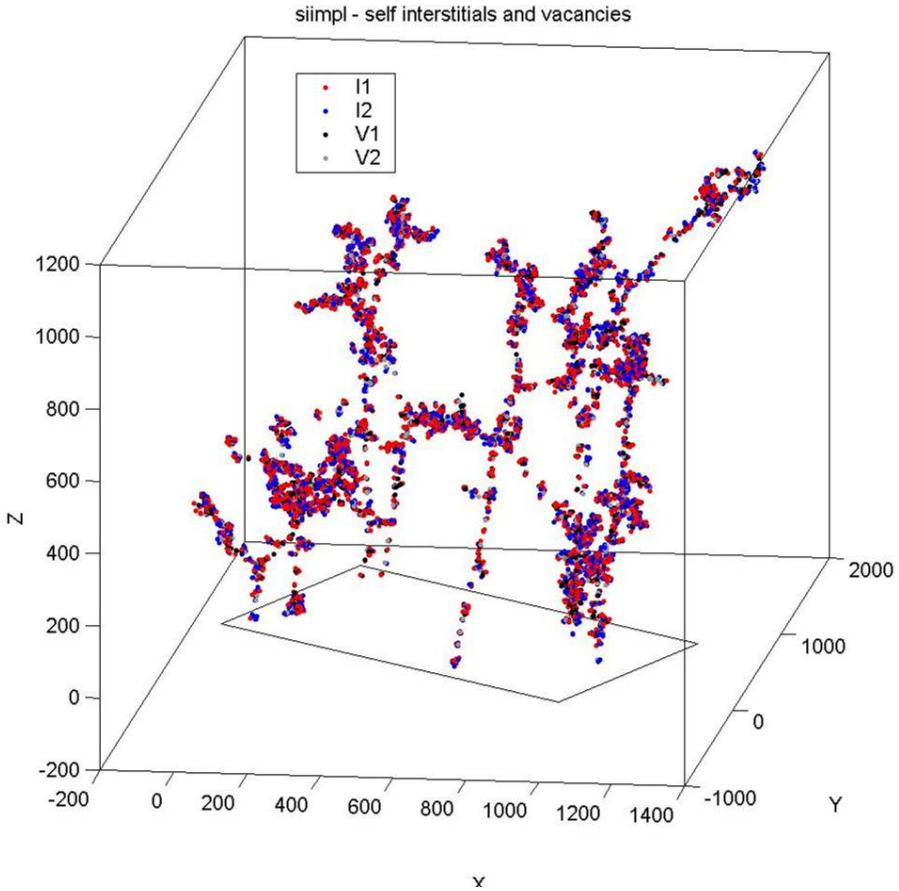




In order to study the 3D defect distributions generated by 10 ions (as above) the following commands should be given in the file (see *3D_defects* directory for complete file):

```
damage.cascade 1;

Nions 10; % number of pseudo ions

saveIVR3
```

The plot function *plotIVR3File* then gives the simulation output

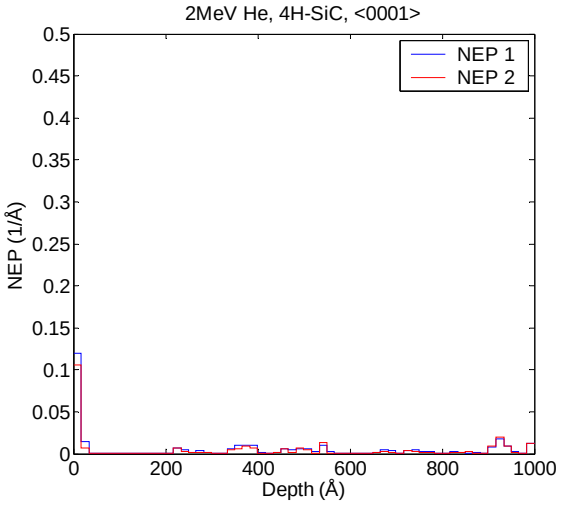


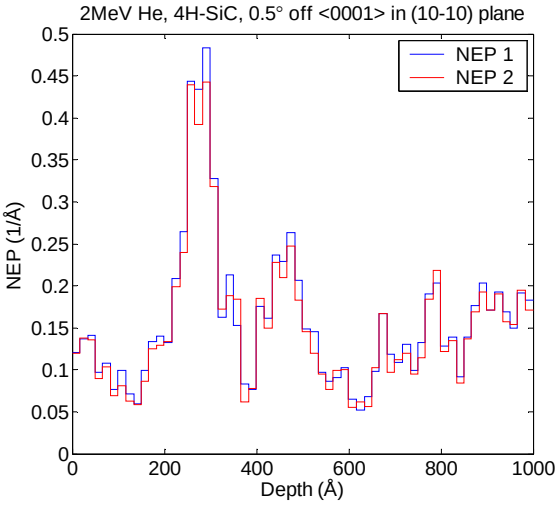
4.7 RBS SCANS FROM NEP CALCULATIONS

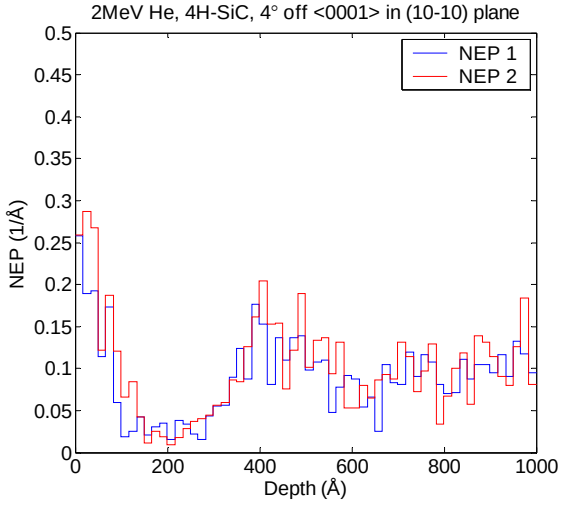
The following example shows how the NEP function may be used to calculate the RBS line spectra in (virgin) 4H-SiC of a 2 MeV He beam.

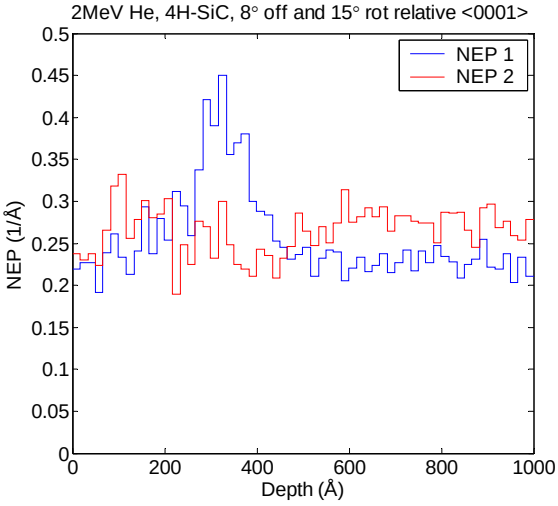
```
% File name: 2MeV_He_RBS.m

% Crystal, target data
readsimfile D:\siimpl\crystals\UC_4HSiC.txt

layerwidth 1000; % (Å)

% --- Implanted ion data  --------
Z1        2;
M1        4; % (amu)
E0        2e3; % (keV)

dose    1e0; % (cm-2)

% First rot around y-axis, the rot around z-axis
impdirrotyrotz 0.5 0; % (degrees)

% ------- Electronic Stopping Data -------
%
%                 ion   atom1  atom2
% ----------------------------
esp.A1            1.7;
esp.A2            0.602;
esp.A3            50.5;
esp.A4            2.68;
esp.A5            2.1;

esp.fl            1.0         1.0       1.0;
esp.s       0.4         0.3       0.5;


% --- MC statistics ---------
Nions 1000; % number of pseudo ions

Ndistz 60; % division of layer width
```

The figures below shows the NEP profiles obtained using the *plotNepFile* with tilts off <0001> in a (10-10) plane of 0, 0.5 and 4°. The profile from a low symmetry direction is also shown.









4.8 BATCH SIMULATIONS OF RBS SCANS

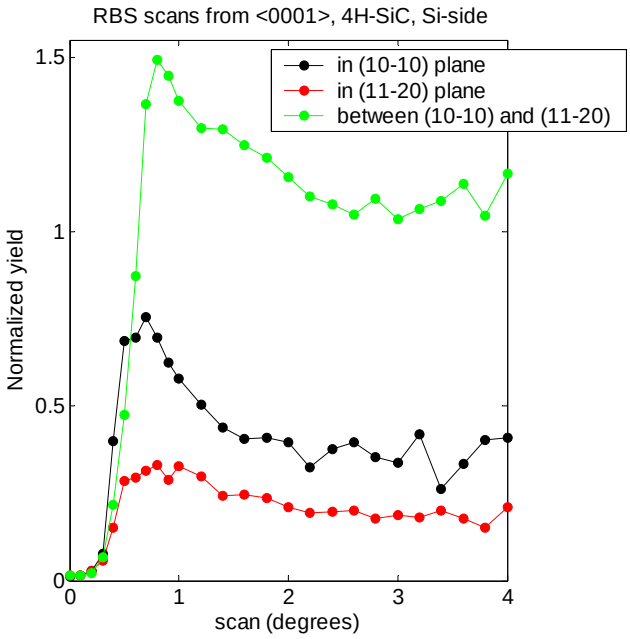
The calculations of full line scans require several input definition files. The example below shows how this can be accomplished automatically by letting MATLAB create and run the SIIMPL input files. In this case, the "RBS yield" is approximated by integrating the Si related NEP signal (*NEP2* below) between depths 250 and 950 Å (see4.7). These yields are normalized to the corresponding NEP signals from a low symmetry direction. In a more detailed calculation it would be better to first extract the RBS spectra from the NEP profiles, from which the yield may be integrated between typical energies used in an experiment.

NOTE that this is not a SIIMPL input file, but a MATLAB program:

```
% File name: RBS_scan.m
rotz = 0;
scan = [0:0.1:1 1.2:0.2:4];

RBSnormal = 0.270;

a = 250;
b = 950;

yield = [];

for i = 1:length(scan);
      % ---- create siimpl init file;

      fid = fopen('init.txt','w');

    fprintf(fid,'readsimfile
d:\\siimpl\\examples\\RBS\\2MeV_He_rbsimpdef_rbs.m \n\n');

    roty = scan(i);
    fprintf(fid,'impdirrotyrotz %e %e;\n\n', [roty rotz]);

    fclose(fid);

      % --- start siimpl simulation
      tic
      !d:\siimpl\siimpl init.txt
      time=toc;
      disp(['Simulation time: ',time2str(toc)]);disp('');

    [z, NEP1, NEP2] = import_mj('NEP.txt');
```

```
    yield = [yield, intvect(z, NEP2, a, b) / ( RBSnormal*(b - a))];
end

save('RBSscan.mat','scan','yield');

axes;
xlabel('scan (degrees)');
ylabel('Normalized yield');
title('4H-SiC, Si-side: from <0001> in (10-10) plane');

plot(scan, yield, 'k-');
```

The figure below demonstrated the scans obtained with the above program in three different planes of 4H-SiC.



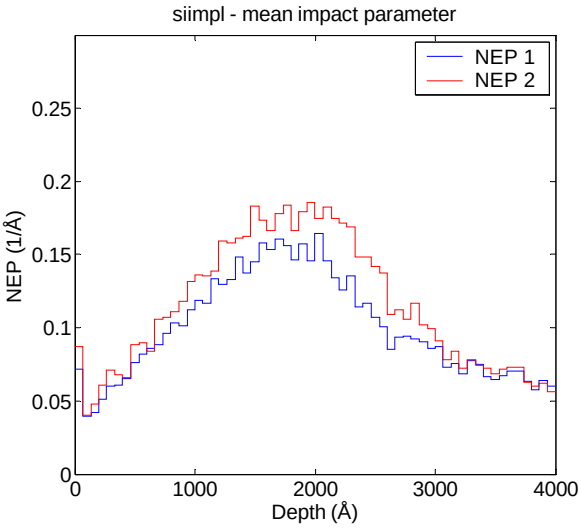
4.9 NEP PROFILE OF AL IMPLANTED 4H-SIC

Finally, an example is given demonstrating the simulation of a c-RBS experiment of an implanted sample. In this case a previous SIIMPL simulation of a 5×10^{14} cm^{-2}, 100 keV Al implantation in 4H-SiC has been executed from the directory *d:\SIIMPL\Examples\rbs_spectra\Al_implant*. The simulation of the c-RBS simulation could then be as follows (note the definition of the

scattering efficiency of the simulated interstitials on the 2 MeV He RBS beam, i.e. the *damage.c_a* instruction):

```
% Crystal, target data
readsimfile D:\siimpl\crystals\UC_4HSiC.txt

importCzI d:\SIIMPL\Examples\rbs_spectra\Al_implant

damage.c_a 2.0;

% --- Implanted ion data  --------
Z1        2;
M1        4; % (amu)
E0        2e3; % (keV)

dose    1e0; % (cm-2)

% First rot around y-axis, the rot around z-axis
impdirrotyrotz 0 0; % (degrees)

% ------- Electronic Stopping Data -------
%
%                 ion   atom1  atom2
% ----------------------------
esp.A1            1.7;
esp.A2            0.602;
esp.A3            50.5;
esp.A4            2.68;
esp.A5            2.1;

esp.fl      1.0         1.0       1.0;
esp.s       0.4         0.3       0.5;


% --- MC statistics ---------
Nions 10000; % number of pseudo ions

layerwidth 4000; % (Å)
Ndistz 60; % division of layer width
```

This simulation generates the following NEP profile (Compare to the 0° tilt NEP profile of a virgin crystal in Example 4.7)


siimpl - mean impact parameter
NEP 1
NEP 2
NEP (1/Å)
0
0.05
0.1
0.15
0.2
0.25
0
1000
2000
3000
4000
Depth (Å)

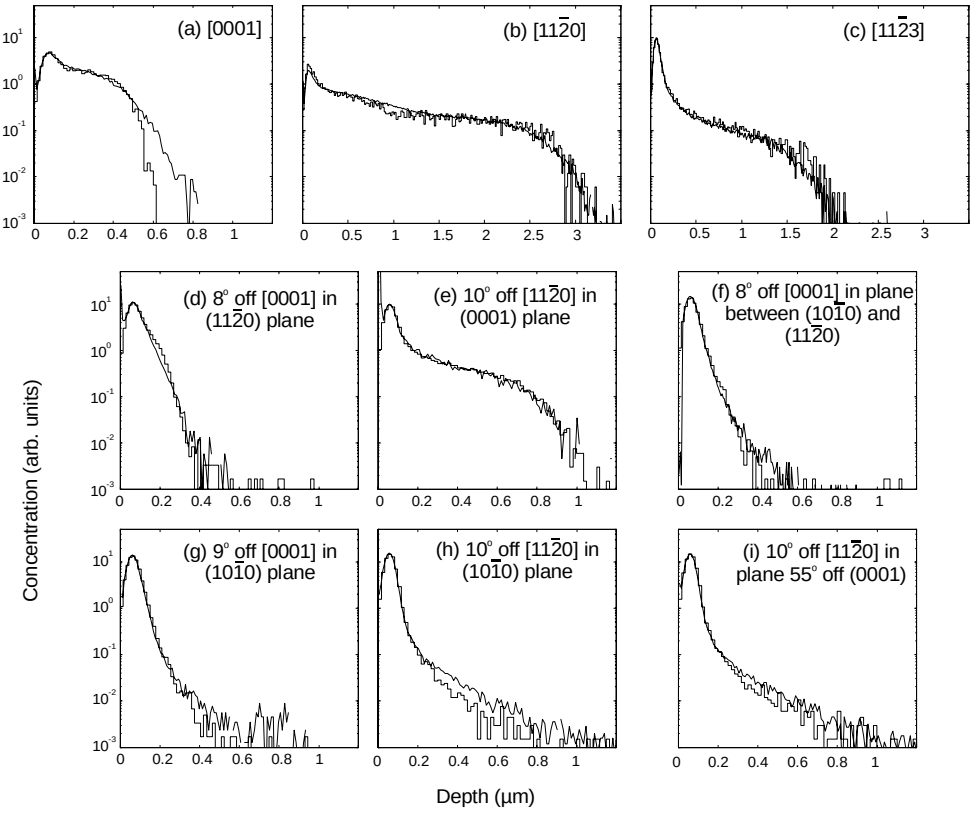
5 SIIMPL VS. EXPERIMENT

This section gives a few examples of SIIMPL simulations together with corresponding experimental SIMS profiles.

5.1 IMPLANTATION DIRECTION, AL IN 4H-SIC

See Ref. [15] for experimental details.

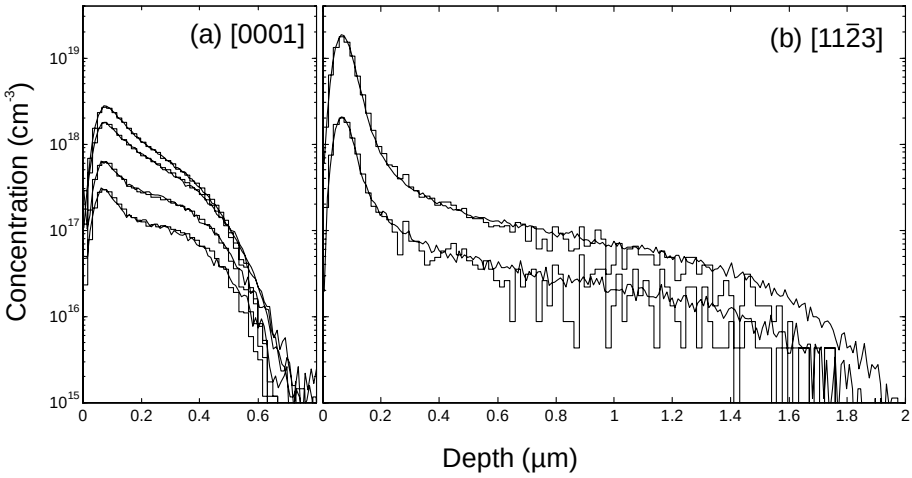
- 4H-SiC
- 60 keV, Al
- ~1×10^{13} cm^{-2}
- 300 K
- *esp.s* = 0.52, all profiles
- Amorphous surface layer = 8 Å, all profiles



5.2 DOSE DEPENDENCE, AL IN 4H-SIC

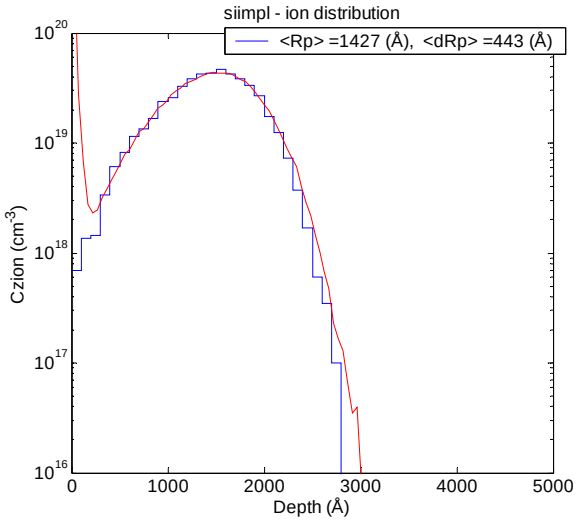
See Ref. [15] for experimental details.

- 4H-SiC
- 60 keV, Al
- 300 K
- *damage.c_a* = 1.2 and 0.50 for [0001] and [11-23] implants, respectively.



5.3 AMORPOUS TARGET, AL IN SIC

- Amorphous SiC
- 115 keV, Al
- 5×10^{14} cm^{-2}
- Simulation, the same as in Example 4.4



SIMS profile in red and SIIMPL simulation in blue.

APPENDIX A

Quasi random numbers

This appendix outlines the algorithm used by SIIMPL to generate a random-like number sequence for neighboring lattice sites of a crystal and that is a function of the lattice position **R** only. The algorithm is designed to be as computer efficient as possible considering that it will be evaluated a large number of times for each collision of the simulation. This is accomplished by creating a 64 bit variable R that expresses the distance from **R** to some reference point in the laboratory system, for example [0,0,0]. The last 10 bits of R will have a random-like behavior when evaluated for neighboring **R** and still give enough information for the current purpose.

Below follows an example of how this algorithm can be implemented in the programming language C. The above described variable R is assumed to have already been calculated. The last ten bits of R are shifted to become the highest bits of the mantissa in the union variable X.QR. The exponent of X.QR is set to 0, which means that X.QR becomes a number between 1 and $(2 - 2^{-10}) \approx 1.999$.

```
#define BITSHIFT 10
// BITSHIFT = (32-1-11 - N) for IEEE standard 754
// N = number of lowest bits that are shiftest to the
// highest bits, here N = 10


union QR_union    {
      double QR;              // 64 bit double
      unsigned int i[2];            // 32 bit integer
      // i[0] == lo byte of QR
      // i[1] == hi byte of QR
};
union QR_union X;

X.QR = R;
X.i[1] = (0x3ff00000 | ((X.i[0]<<BITSHIFT) & 0x000fffff));
```

APPENDIX B

Random scattering probability *P*

P can be expressed as the ratio of the average path lengths between collisions with target atoms, $<\Delta r>$, and that between interstitial atoms, $<\Delta r_I>$, respectively:

$$P = <\Delta r> / <\Delta r_I>, \tag{B-1}$$

where $<\Delta r>$ for a random ion trajectory becomes

$$<\Delta r_{rand}> = (\pi p_{max}^2 N)^{-1}. \tag{B-2}$$

$<\Delta r_I>$ is similarly written as

$$<\Delta r_I> = (\pi p_{max}^2 c_a N_I)^{-1}, \tag{B-3}$$

which must be true for all ion trajectories since the model assumes randomly distributed interstitials. Assuming a random trajectory, i.e. $<\Delta r> = <\Delta r_{rand}>$ and with *P* evaluated according to Eq. (X), Eq. (B-1) shows that $<\Delta r_I>$ equals that specified by Eq. (B-3). However, for an ion with a channeling trajectory, Eq. (B-2) is no longer valid due to the ordered movement of the ion relative the lattice, and $<\Delta r_I>$ will thus differ from the value stipulated by Eq. (B-3). Simulations have shown that depending on the value of p_{max}, $<\Delta r>$ for a channeled ion can differ up to about 30% compared to $<\Delta r_{rand}>$. This has the effect that the dechanneling rate caused by implantation induced damage is over- or underestimated by the same amount. One way to overcome this problem is to instead evaluate *P* as:

$$P = c_a N_I / N \times <\Delta r_{loc}> / <\Delta r_{rand}> \tag{B-4}$$

where $<\Delta r_{loc}>$ is a local value, determined during the simulation and updated typically every 50 collisions. In this way, Eq. (B-3) will always hold.

APPENDIX C

Random number generator

C-code random number function *randmj* used in SIIMPL.

```
double randmj(void)
// This is a "Minimal Standard" random number generator
// Descripbed by  Edgar H. Sibley in Communications of the
// ACM, Volume 31 Number 10, October 1988
{
   #define m  2147483647L
   #define q  127773L
   #define a  16807
   #define r  2836

   randomz = a*(long)(randomz % q) - r*(long)(randomz / q);
   if (randomz <= 0) randomz += m;
   return (double) randomz / m;
}
```

APPENDIX D

SIIMPL Log file

SIIMPL Log-file *siimpllog.txt* produced in the simulation example of Sec. 4.2.

```
siimpl, simulation of ion implantation, (c) M.S. Janson
Log file:

vibu1  0.057000  0.051000  (Å)
Edisp  0.020000  0.035000  (keV)

Z2  6  14
M2  12.000000  28.000000  (amu)

Number of sub cells (SC) = 24
Totat number of atoms in all SC = 326
Avarege number of atoms per SC = 13.583333

Z1 13
M1 27.000000 /(amu)
E0 60.000000 (keV)
dose 1.000000e+14(cm^-2)

Simulated pseudo ions        : 100000
  (including rare event ions): 100000

% Primary ion ESP data:
esp.A1 4.000000 (eV/(10^15cm^-2))
esp.A2 0.520000   (= esp.p)
esp.p2 0.000000
esp.A3 10859.940000 (keVeV/(10^15cm^-2))
esp.A4 7.801400 (keV^-1)
esp.A5 0.640705 (keV)

esp.fl 1.000000
esp.s  0.520000

IAPfunction: 0
pmax = 2.000000 (Å)
Estop = 2.000000e-03 (keV)
vibalg: 1

Implanted direction:
1.391731e-01      0.000000e+00      9.902681e-01
```

```
Surface Normal:
-1.391731e-01     0.000000e+00      -9.902681e-01

Implanted rectangle area (Å):
0.000000e+00      0.000000e+00      0.000000e+00
9.575820e+02      -2.548040e+02     -1.345794e+02
1.209906e+03      7.121888e+02      -1.700412e+02
2.523242e+02      9.669927e+02      -3.546186e+01

surfaceEnergy = 0.005000 (keV)

Damage model: Full cascade

Random surface layer:0.000000 Å

rndscatfac: 0.000000e+00
Total number of collisions: 86623787
Total number of collisions for primary ion: 10346949
total number of nofound: 224580

Rp 711.964180
Rpath 844.822800

<dRv0> (ion) =  0.816495
Random dRv0 =  0.817712

Backscattered ions: 4.000000e+10 (cm^-2)
Transmitted ions: 0.000000e+00 (cm^-2)
Sputtered target atoms: 4.604000e+13 (cm^-2)

Energy test: Eout/Ein - 1 = 2.637686e-09
```

REFERENCES

[1] M. T. Robinson and I. M. Torrens, Phys. Rev. B **9**, 5008 (1974).
[2] O. S. Oen and M. T. Robinson, Nucl. Instr. and Methods **132**, 647 (1976).
[3] J. P. Biersack and L. G. Haggmark, Nucl. Instr. Meth. **174**, 257 (1980).
[4] J. F. Ziegler, J. P. Biersack, and Y. Littmark, *The stopping and range of ions in solids* (Pergamon press, 1985).
[5] M. Fujinami and N. B. Chilton, J. Appl. Phys. **73**, 3242 (1993).
[6] A. Nylandsted-Larsen, C. Christensen, and J. W. Petersen, J. Appl. Phys. **86**, 4861 (1999).
[7] M. Posselt, B. Schmidt, T. Feudel, and N. Strecker, Mat. Sci. and Eng. B **71**, 128 (2000).
[8] G. Lulli, E. Albertazzi, M. Bianconi, and R. Nipoti, Nucl. Instr. and Meth. in Phys. Res. B **148**, 573 (1999).
[9] J. J. Thomson, *Conduction of Electricity Through Gases* (Cambridge Univ. Press, 1903).
[10] B. Yuan, F. C. Yu, and S. M. Tang, Nucl. Instr. Meth. Phys. Res. B **83** (1993).
[11] J. F. Ziegler, *Helium Stopping Powers and Ranges in All Elemental Matter* (Pergamon Press, New York, 1977).
[12] G. Lulli, E. Albertazzi, M. Biancola, R. Nipoti, M. Cervera, A. Carnera, and C. Cellini, J. Appl . Phys. **82**, 5958 (1997).
[13] S.-H. Yang, D. Lim, S. J. Morris, and A. F. T. Jr., Nuclear Instrum. Methods B **102**, 242 (1995).
[14] J. H. Barret, Phys. Rev. B **3**, 1527 (1971).
[15] J. Wong-Leung, M. S. Janson, and B. G. Svensson, J. Appl. Phys. **93**, 8914 (2003).