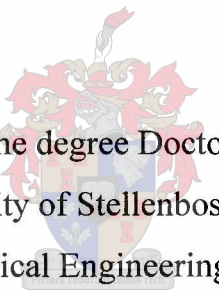
The Optimisation of mm-Wave GaAs Gunn diodes using a parallel implementation of the Monte Carlo particle simulation technique

Robert Ryk van Zyl



Dissertation presented for the degree Doctor of Philosophy at the University of Stellenbosch.
(Electrical Engineering)

Promoter:	Prof. W.J. Perold (University of Stellenbosch)
Co-promoter:	Dr. R. Botha (Nelson Mandela Metropolitan University)

December 2006

DECLARATION

I, the undersigned, hereby declare that the work contained in this dissertation is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Date: 30 November 2006

ABSTRACT

The principal research objective is the optimisation of the output power of GaAs Gunn diodes in the mm-wave spectrum. Specifically, the optimisation of Gunn diodes operating at 94GHz is investigated due to its relevance to current automotive and military precision radar applications.

A novel multi-domain Gunn diode with multiple hot-electron launchers is proposed and evaluated. This concept has been successfully applied to a double-domain Gunn diode. Further avenues of optimisation that have been incorporated into the design are notch doping and grading of the active layer doping profile. Output power in the region of 160mW at 2% efficiency can be expected from these diodes. This is far superior to current state-of-the-art GaAs Gunn diodes which are capable of around 90mW at 94GHz.

Although it has not been investigated, the optimised diode should benefit from the same advantages of a single domain hot-electron launcher diode. These advantages are reduced sensitivity to temperature and bias variations, improved turn-on characteristics and noise performance.

The design has been optimised using a novel parallel implementation of the Monte Carlo particle simulation technique. A cluster of personal computers, linked via a dedicated high-speed gigabit network, has been established. This renders a cost-effective super computer. The simulation model has been verified rigorously by comparing simulation results with real-life scenarios.

Thermal effects are incorporated into the overall Monte Carlo model. Temperature is determined with fine grid-resolution throughout the device and not assumed constant. This enables us to investigate the influence of graded doping profiles on device performance in more detail, and renders a more realistic model of high temperature Gunn diodes.

OPSOMMING

Die hoofnavorsingsdoel is die optimering van die uittree-drywing van GaAs Gunn diodes in die mm-golfspektrum. Die optimering van Gunn diodes by 94GHz word spesifiek ondersoek weens die relevansie tot huidige voertuig en militêre presisie radartoepassings.

'n Nuwe multi-domein Gunn diode met meervoudige warm-elektron injektors word voorgestel en geëvalueer. Hierdie konsep is suksesvol op 'n dubbeldomein diode toegepas. Verdere optimeringsopsies is ook geïnkorporeer. Dit sluit in kerf ("notch") dotering en die gradering van die doteringsprofiel in die aktiewe gebied. Uittree-drywing van ongeveer 160mW teen 2% effektiwiteit kan van hierdie diodes verwag word. Dit is aansienlik beter as huidige gespesialiseerde GaAs Gunn diodes wat in die omgewing van 90mW by 94GHz realiseer.

Alhoewel dit nie pertinent ondersoek is nie, behoort die geöptimeerde diode dieselfde voordelige kenmerke te vertoon as die enkeldomein eweknie. Dit sluit in minder sensitiwiteit tot temperatuur- en voorspanningsveranderinge en verbeterde aanskakel-en geraaskarakteristieke.

Die ontwerp is geöptimeer met 'n nuwe parallel-geïmplimenteerde Monte Carlo partikel simulasietegniek. 'n Groep persoonlike rekenaars, wat deur 'n hoë-spoed giga-bit netwerk verbind word, is opgerig. Hierdie implementering gee 'n koste-doeltreffende superrekenaar. Die simulasiemodel is noukeurig getoets teen praktiese eksperimente.

Temperatuur-effekte is as integrale deel van die Monte Carlo model geïnkorporeer. Temperatuurprofiele word met hoë rooster-resolusie regdeur die diode bepaal en nie as konstant aanvaar nie. Dit stel ons in staat om die invloed van gegradeerde doteringsprofiele in die aktiewe gebiede op die werkverrigting van die diode te bepaal. Dit gee ook 'n meer realistiese model van hoë-temperatuur Gunn diodes.

CONTENTS

Chapter 3: MC-PVM: A PARALLEL MONTE CARLO SIMULATOR

Chapter 1

INTRODUCTION

1.1 RESEARCH OBJECTIVE AND METHODOLOGY

The principal research objective is the optimisation of output power of GaAs Gunn diodes in W band (75-111GHz). Specifically, the optimisation of Gunn diodes operating at 94GHz is investigated due to its relevance to current automotive and military precision radar applications. However, the findings are not confined to 94GHz, but could be applied to the full mm-wave spectrum.

As will be pointed out in subsequent sections, the maximum output power obtained from commercially available Gunn oscillators is of the order 80mW at 90GHz and 70mW at 94GHz. A goal of 100mW at 94GHz has therefore been set to achieve high power Gunn diode oscillators. (This is in agreement with a similar venture under the auspices of the South African Council for Scientific and Industrial Research (CSIR) [1].)

The exorbitant infrastructure costs involved in developing these specialised diodes have put the local manufacturing of these diodes out of reach. It has therefore been decided to mimic these diodes in reliable software "experiments" to characterise their performance. A Monte Carlo particle simulator has been developed for this purpose.

The verification of simulation results presented in this work is an integral part of sound research methodology. Especially in the absence of real experiments, these software "experiments" must be subjected to extensive verification. This has indeed been implemented, as will be pointed out in Chapters 2 and 5, where simulation results have been meticulously compared with real world bulk material and discrete device experiments.

1.2 OVERVIEW

1.2.1 Monte Carlo Particle Simulation Technique

The Monte Carlo method has been dealt with in various degrees of complexity by numerous authors. Some excellent references are available in the literature, [2] - [6]. The author has also

written on the subject [7], [8], aimed at the local research establishment. A summary of the model employed in this work is given in Chapter 2.

Two novel implementations should be mentioned here. Firstly, the Monte Carlo algorithm is computationally extremely intensive and single desktop computer implementations are unrealistic. In the absence of funding for a supercomputer, the author has developed a parallel version of the Monte Carlo algorithm which is implemented on a cluster of (affordable) personal computers [9] - [11]. The parallel implementation of the Monte Carlo algorithm is dealt with extensively in Chapter 3.

Secondly, as will be pointed out in this work, thermal effects play a crucial role in Gunn diode performance. This is even more pertinent to the double-domain diodes proposed in this work. Unlike conventional Monte Carlo simulators where a constant, and sometimes unrealistic, lattice temperature throughout the device is assumed, the temperature profile has been determined consistently with the time-evolution of the particle current and internal electric field distribution.

1.2.2 The Gunn diode

The American Heritage dictionary of the English Language [12] defines the Gunn effect as "the production of microwave oscillations when a constant voltage in excess of a critical level is applied to opposite faces of a semiconductor." JB (Ian) Gunn discovered the Gunn effect on 19 February 1962 [13]. He observed random noise-like oscillations when biasing n-type GaAs samples above a certain threshold. He also found that the resistance of the samples dropped at even higher biasing conditions, indicating a region of negative differential resistance. (It should also be noted here for historical value, that B.K. Ridley and T. B. Watkins in 1961 [14] and C. Hilsum in 1962 [15] independently predicted that the transferred electron effect would occur in GaAs.)

Figure 1.1 shows part of the famous page from one of Gunn's laboratory notebooks with the entry "noisy" on the line for 704V. Describing it as the "most important single word" he ever wrote, it laid the foundation for what was to become a major mode of a.c. power generation.

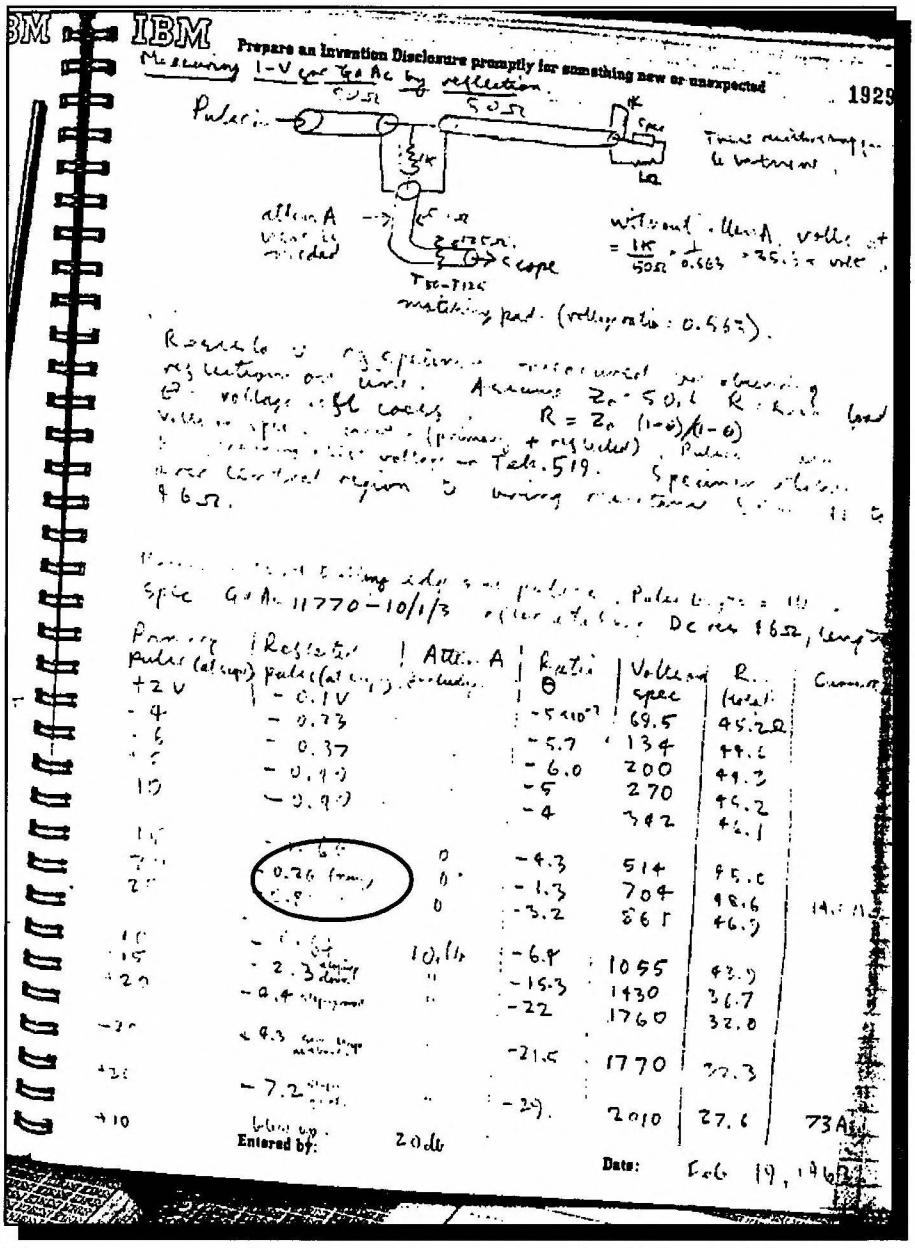


Figure 1.1: A page from Gunn's laboratory notebooks in which he noted the discovery of the Gunn effect (taken from [16]). The famous entry "noisy" is encircled.

An overview of the Gunn effect, and how it is exploited in Gunn diodes, is given in Chapter 4.

The boundaries of Gunn operation have since been greatly expanded through optimisation. Commercially available diodes today are capable of producing from 40mW [17] up to 80mW at about 90GHz [18]. Diode optimisation is discussed Chapter 5. An overview of optimisation efforts is given in the next section.

1.3 OPTIMISATION OF GUNN DIODES

1.3.1 Harmonic mode operation

Implicit in the optimisation of Gunn diodes operating above about 65GHz, is the fact that they rely on harmonic operation [19]. Second-harmonic mode operation for W-band applications has been investigated since the mid-seventies, with useful power reported at frequencies of up to 110GHz [20], [21], [22].

Fundamental mode operation at these frequencies is very inefficient. The reason for this is two-fold, and both relate to the length of the active region. Fundamental mode operation at about 100GHz requires the active region to be of the order <1μm long (see data presented in Figure 4.3). The active region length of a corresponding second-harmonic mode diode would be double this value.

The shorter length of the active region (in fundamental mode) would require a bias voltage of about half that of the second-harmonic mode. These lower voltage levels reduce the available output power. A second effect that the shorter active region has on the efficiency of the diode, is the prominence of the "dead zone" near the cathode. The dead zone is a region where no Gunn domain formation takes place due to the finite distance that electrons require to transfer to the satellite valleys. It is therefore essentially a parasitic series resistance that reduces efficiency. These dead zones are more appreciable in shorter devices.

We can therefore conclude that fundamental mode operation will not be pursued in this work. The more efficient second-harmonic operation will be assumed.

1.3.2 Hot-injection launcher

The effect of high-energy electron injection at the cathode has been discussed by Greenwald [23], [24]. Hot-injection entails the launching of electrons from the cathode at elevated energies, close to the threshold for significant electron transfer from the central to satellite valleys.

Neylon *et al* [25] points out numerous advantages of using hot-electron injectors in Gunn diodes:

- In the first place it reduces the dead zone, since domain formation and nucleation are greatly enhanced close to the cathode. As already mentioned, this enhances efficiency.

- Hot-injector Gunn diodes also display much improved turn-on characteristics, compared to conventional diodes. This allows coherent oscillations around peak power over the full military specification temperature range.
- The diode's performance is much more independent of operating temperature than its conventional counterpart. This is due to the fact that hot-injected electrons have temperatures of the order 2000K, much more than the nominal operating temperature of Gunn diodes (about 450K at ambient room temperature).
- The diode's output power and position of domain nucleation can be made much less sensitive to bias variation with an appropriate launcher structure (see also [26]). The latter would imply greater frequency stability.
- Seen in totality, the hot-injector diode's noise performance will be superior to that of a conventional diode.

An appropriate launcher is proposed in [27] and [28]. The heterostructure injector consists of an undoped linearly graded $Al_xGa_{1-x}As$ layer, followed by a very narrow n^+ doping spike. This spike serves as a non-equilibrium connector to prevent depletion, set up by the forward biased injector, from extending into the active region. The structure is discussed in more detail in Chapter 5.

A disadvantage of employing hot-electron launchers is the creation of a high electric field domain on the anode side of the active layer (see Section 5.3.5). These electric fields cause excessive heating. This would explain why, initially, these diodes have not delivered on the promise of much higher output power and efficiency [29]. Grading of the active region doping profile can be employed to counter this effect (see Section 1.3.4).

1.3.3 Notch doping

The incorporation of a nominally undoped notch at the cathode, preceding the active layer, also reduces the dead zone. The doping notch encourages high electric fields, above the critical value for Gunn domain formation, at the cathode. An early mentioning of a doping notch is by Tully [30]. Doping notches are frequently used in conjunction with hot-electron launchers, e.g [31].

1.3.4 Grading of the active region doping profile

For optimum device efficiency, a uniform resistivity profile across the active layer is required [32], [33]. This has erroneously been interpreted by Hasegawa [34] to imply a uniform doping profile, and many has followed suit. However, it was subsequently shown that increasing the doping density from cathode to anode enhances diode efficiency and output power [29], [35], [36], [37]. This can only be explained by taking the varying temperature profile across the active layer into account (Hasegawa assumed a constant temperature profile) [33].

In the work presented here, the temperature profile is determined throughout the device, and not assumed constant. The implicit effect it has on the resistivity profile can therefore be investigated in great detail. This is discussed further in Chapter 5.

1.3.5 Multi-domain operation

Intuitively, the output power of a Gunn diode can be enhanced by merely increasing the cross-sectional area of the diode. This will increase the current for a given terminal voltage. This is, however, unrealistic because the admittance that the external circuit must present to the diode for optimal matching purposes, also decreases. Circuit losses will increase as a result [38]. This, together with an increase in operating temperature due to enhanced d.c. power dissipation, will place a limit on the minimum cross-sectional area of the diode.

A solution to this problem has been the combination of series-connected diodes to improve output power, as originally proposed Thim [39] in 1968. Thim stacked several wafers on top of

each other. In subsequent work by Slater and Harrison [40], a *horizontal* diode was implemented where multi-domain nucleation centres were forced by literally scratching the surface of the active layer. Subsequently, in 1979, Talwar [41] reported on a dual-diode 73GHz Gunn oscillator that produced double the output power of a single diode. (Two discrete diodes were used.)

The basis of these approaches is that, with series connected Gunn diodes, the impedance of the grouped diodes (whether discrete or integrated into a single device) increases. Therefore, the area of each individual diode may be increased while the group will still present a favourable impedance level to the external circuitry.

Tsay *et al* [42] incorporated this approach into a single diode with multiple domain nucleation and quenching regions. Their diode essentially consists of multiple active layers, separated by highly doped regions where the electric field is quenched. They showed that, with an N-domain diode, an increase of N^2 in output power could be obtained in favourable frequency ranges. Teo and Dunn [43] verified this approach with Monte Carlo simulations. They have found no obvious limit to the number of domains that can successfully be incorporated, which is unrealistic if thermal effects are properly accounted for. They have, however, alluded to the fact that significant heat generation with more transit regions will affect the operation of these diodes. This is indeed the case, as will be pointed out in Chapter 5.

1.3.6 Multi-domain operation with multiple hot-electron launchers

The two main optimisation routes mentioned in the foregoing discussions, namely multi-domain operation and the incorporation of hot-electron launchers, is proposed by the author [44], [45]. These multi-domain Gunn diodes with multiple hot-electron launchers will, intuitively, benefit from the advantages associated with both hot-injection and multi-domain operation. This premise forms the basis of the work presented here. Furthermore, the incorporation of notch doping and graded doping profiles are investigated and incorporated in the overall optimisation approach.

Previous work on multi-domain diodes mentioned above ([42], [43]) ignored the very important thermal aspects of Gunn operation, by assuming unrealistic nominal operational temperatures, as well as constant temperature profiles in the active layers. Multi-domain diodes operate at highly elevated temperatures, due to the fact that the power dissipation also increases with the square of the number of domains. In this work, the temperature is determined throughout the device with high grid resolution, and not assumed constant. Thermal effects are therefore addressed accurately in the optimisation of the Gunn diode. It is worth mentioning here that this treatment points to the necessity of using diamond heat sinks for efficient multiple-domain operation.

The optimisation of these multiple-domain, multiple-launcher Gunn diodes is discussed at length in Chapter 5. The main conclusion of the work is that this "cocktail" approach, incorporating various optimisation strategies, yields diodes that are far superior to those that are commercially available. This is discussed in more detail in Chapter 6 where conclusions are drawn on the work presented.

1.3.7 Scope and limitations

This work is limited to the optimisation of the output power of Gunn diodes. Noise characterisation and frequency drift due to temperature and bias conditions have not been investigated. It can, however, be assumed that the optimised diode will display superior performance in terms of noise and drift, compared to conventional Gunn diodes, because of the hot-injection launchers.

The investigation is limited to a double-domain Gunn diode. It is the opinion of the author that the incorporation of more domains will not yield appreciably more power, if any, compared to the double-domain diode due to excessive power dissipation. Furthermore, variations in layer characteristics (doping and dimensions) will play an increasingly important role in higher order domain diodes.

It has been decided to include a somewhat extensive overview of the Monte Carlo algorithm to

serve as reference to local researchers interested in this method. (The Monte Carlo technique is still much avoided in the local research fraternity.)

1.3.8 Dissertation Layout

Chapter 2 deals with the Monte Carlo particle simulation algorithm. Special attention is given to the incorporation of the thermal model into the algorithm, because the author is unaware of any such treatment in literature. The chapter concludes with the very important aspect of model verification. Verification of both bulk material and device simulations is done.

In Chapter 3 the parallel implementation of the Monte Carlo simulation on a cluster of personal computers is discussed. This novel and cost-effective approach has led to a vast increase in computer power, without which the work presented here would not have been possible. Again, verification of the parallel implementation is carried out.

An overview of the Gunn effect and its incorporation into Gunn diodes and oscillators is given in Chapter 4. The Gunn effect in the strict sense is discussed as well as typical characteristics of Gunn oscillators.

The optimisation of Gunn diodes is treated in Chapter 5. A novel multi-domain diode with multiple hot-launchers is investigated. A very efficient double-domain with a diamond heat sink is established. The Chapter concludes with a yield analysis of this diode by investigating the effect of variances in active layer doping and length on the diode's performance.

The dissertation concludes with Chapter 6 which critically discusses the main achievements of the thesis.

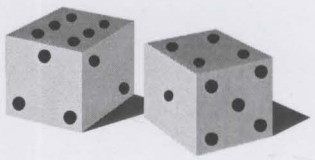
The code for the parallel implementation of the Monte Carlo algorithm is presented in Appendix A.

Chapter 2

THE MONTE CARLO DEVICE SIMULATION ALGORITHM

Sequences of truly patternless, unpredictable digits are a valuable commodity.

- Ivars Peterson



2.1 INTRODUCTION

Peterson's comment could be no closer to the truth in the context of Monte Carlo simulations. Randomness unlocks the awesome power intrinsic to the Monte Carlo simulation method. Monte Carlo simulations are statistical numerical integration methods used in the simulation of physical processes that are governed by probability distributions. The behaviour of such systems can be simulated by sampling from appropriate random distributions. It is possible to generate arbitrary distribution functions through the generation of uniformly distributed random numbers by means of the direct method [3] which is easily implemented on a computer.

The Monte Carlo method is applicable to a variety of fields, from analysing phenomena in quantum physics to the investigation of the formation and development of galaxies. The application of the Monte Carlo method to semiconductor material and device simulations dates back to the mid sixties when Kurosawa [46] reported on the simulation of hot electron problems. Monte Carlo methods are employed in semiconductor simulations for the implicit solution of Boltzmann's transport equation [2]. This enables the characterisation of materials used in device simulations. In its application to semiconductor simulations, the Monte Carlo method tracks the movement of electrons (or holes) through the crystal. It is evident that the Monte Carlo simulation imitates, and in fact closely resembles, a *physical experiment.*

Herein lies its remarkable power as a versatile simulator. It permits the simulation of certain physical phenomena which are unattainable, or rarely observed, in experiments and even the investigation of hypothetical materials to deepen our understanding of the underlying physics.

An important consideration is the Monte Carlo's applicability to the simulation of small scale and/or hot electron devices. It is of particular importance to microwave and high speed digital circuits. Classic simulations are based on drift-diffusion models [47] under the unrealistic assumption that the charge carriers assume steady-state immediately after a field variation. Steady-state is only reached after a finite time and associated distance [48]. Drift-diffusion models can therefore not be used in simulations where the physical size of the device or the

frequency of operation prohibits steady-state.

In the following section the Monte Carlo simulation algorithm will be discussed in brief. This will be followed by a more in-depth treatment of certain pertinent topics raised in the foregoing overview. The chapter is concluded by a discussion on the GaAs material parameters used in the simulation model of this work and a verification of the model by comparing it to experimental mobility curves as well as a benchmark diode simulation.

2.2 THE MONTE CARLO DEVICE SIMULATION ALGORITHM

Monte Carlo simulations of semiconductors track the time evolution of ensembles of electrons (or holes) through the crystal in both real- and $\boldsymbol{k}$-space (the three-dimensional vector space defined by all possible wave vectors). The algorithm follows closely the sequence of events as experienced by the charge carriers in the confines of the device.

Before discussing the algorithm in more detail, it is necessary to ponder upon the feasibility of simulating individual charge carriers. In real devices the actual number of charge carriers is in the order of $N = 10^7$. To follow such a large number of particles is impractical with current computational resources. It is therefore feasible to select N_s ($<N$) particles from the full ensemble of charge carriers. These particles, or "super particles", are representative of the actual charge particles in the device. If an appropriate random selection is made, the simulation of super particles will give a meaningful, and in fact reliable, picture of the overall charge transport within the device [2]. The relatively limited number of super particles employed in the simulation will undoubtably have an adverse effect on the validity of the results. A compromise has to be made between the statistical accountability of the results and the simulation time needed to simulate the super particles. As a rule of thumb, it is recommended that at least 10 super particles be incorporated for each grid point in the simulation area [49], which, for the purposes of numerically solving relevant internal distributions, is spatially divided into a mesh of discrete regions. Typical values for N_s lie between 10 000 and 50 000. In the high temperature simulations presented in this work, 80 000 super particles are used to minimise noise in the

terminal currents. In order to determine the internal field distribution correctly, the effective charge of each super particle has to be weighted to compensate for the shortage of total charge. The super particle charge q_s is given by the volume integral of the doping density n_d over the entire simulation volume V

$$q_s = \frac{q}{N_s}\left(\int_V n_d dv\right) \tag{2.1}$$

with q the unit electronic charge.

It is very important to note that the super particle has to be treated as a *single* electron with unit charge when describing the particle's dynamics. Throughout this work, where only *n*-type Gunn devices are considered, the simulated super particles will simply be referred to as "electrons" with the understanding that the reader will make the necessary distinction in the context of the discussion above.

The Monte Carlo algorithm follows closely the sequence of events as experienced by the electron in the confines of the device.

The initiation procedure in essence comprises the distribution of the ensemble of electrons in real and $\boldsymbol{k}$-space. The electrons are spatially distributed in accordance to the doping density profile [49]. This results in an initial charge-neutral state. In $\boldsymbol{k}$-space the electrons are assigned wave vectors according to a Maxwellian distribution of velocities at the lattice temperature [49].

In addition to the above, the initiation phase includes the initialisation of parameters that are used in subsequent *repetitive* procedures such as the resolution of the internal field distribution (section 2.2.5) and the temperature profile (section 2.2.6) as well as tabulating the scattering rates (section 2.2.2). These activities are primarily aimed at saving the computation time, which would have been required for the repeated calculation of these parameters.

The area of simulation is divided into a mesh of cells. This is required for resolving distributions

which are functions of position, for example the electrons' velocity and density distributions as well as the internal electric field distribution and temperature profile. The field distribution is resolved through the solution of Poisson's equation [50], which is dealt with in section 2.2.5, whilst the solution of the heat equation renders the temperature profile. The latter will be discussed in section 2.2.6.

The mesh should be fine enough to ensure acceptable accuracy of the simulation results. However, an increase in the number of cells will inevitably increase the computational cost involved. A compromise between accuracy and computational cost is achieved by choosing the cell dimensions equal to the Debye length. Special consideration should be given to areas within the device where material property variations occur within extremely short lengths, such as the heterostructure hot launchers employed in this work. It must be ensured that the chosen mesh is fine enough to model these changes satisfactorily.

As stated earlier, the simulation comprises the tracking in time of the spatial evolution of an ensemble of electrons through the semiconductor. The spatial distribution is sampled at regular intervals. This spatial distribution is related to a charge density distribution through the appropriate assignment of charge to the mesh points. Various assignment procedures can be considered, for example Nearest Grid Point (NGP) and Cloud-in-Cell (CIC) assignment. NGP assignment assumes that all the charge associated with a simulated electron, or super particle in this case, is allocated to "the grid point nearest" to the electron. CIC assignment, in contrast, splits the charge of the super particle and assigns it to adjacent grid points proportional to the distances between the respective grid points and the electron. Variations of the CIC is possible by changing the sphere of influence of the particle. In the current implementation this sphere is limited to the grid points directly adjacent to the particle. Care should be taken when applying the CIC assignment scheme near abrupt heterostuctures. CIC assignment could overestimate the charge distribution on one side of the junction and underestimate it on the other [5]. The advantage of CIC assignment, compared to the NGP procedure, is the reduction of noise in the charge distribution, especially in highly doped regions. However, this does come at the expense of an increased computational load.

After the charge has been assigned to the grid points, the electric field is solved, updated and held constant during the ensuing interval. This interval is referred to as the field adjusting time step, T_{step}. For a stable simulation the choice of T_{step} has to adhere to two conditions: it must be chosen smaller than the inverse of the plasma frequency and should also be short enough to ensure that an electron will not traverse more than one cell between field updates. In the case of updating of the temperature profile, which is assumed to be the steady-state distribution, the time scales of interest are much longer than that of the field calculations. The criteria above therefore do not apply in this case, which effectively means that it is sufficient to update the temperature profile at every i^{th} T_{step} interval to reduce the associated computational cost.

The motion of the ensemble is obtained through the successive simulation of the dynamics of each electron in the ensemble for the duration of T_{step}. For each electron a free flight duration τ is generated according to distribution function [4]

$$P(\tau) = \lambda_T(\tau)\exp\left(-\int_{t=0}^{\tau}\lambda_T(t)\,dt\right) \tag{2.2}$$

where λ_T is the total of the combined scattering rates of all possible scattering mechanisms.

To avoid the time intensive numerical solution of (2.2) for τ, Rees [51] has introduced a virtual scattering mechanism, namely self scatter, which does not change the state of the electron. The self scatter rate fluctuates to ensure that λ_T, which is a function of the electron's energy, remains constant *throughout* the simulation. The self scatter algorithm, combined with the direct method [4], leads to a simplified equation for τ given by

$$\tau = -\frac{\ln(r)}{\lambda_T} \tag{2.3}$$

with r a random number evenly distributed between 0 and 1. Electron scattering will be discussed in Section 2.2.2.

The state and position of the electron at the end of the free flight are determined by solving the applicable *classical* equations of motion as discussed in Section 2.2.3. A scattering process, selected in accordance with the electron's current state, terminates free flight. The state of the electron after scattering serves as the initial state for the following free flight.

The process of particle acceleration and scattering continues until the accumulation of free flights, τ_{acc}, exceeds T_{step}. If this condition is reached while an electron is in free flight, it will continue to accelerate for the remainder of its free flight during the following T_{step}. The next electron is selected and simulated during the current T_{step}.

One-dimensional device simulations, where both ends of the simulation model are terminated with metal contacts, require a satisfactory modelling of these boundaries to ensure successful simulations. In the present work, all contacts are assumed to be perfect ohmic contacts, which implies that these regions are to be kept charge neutral throughout the simulation, while also allowing for the free flow of electrons across through the contact [2]. This is ensured by absorbing (deleting) the electrons that flow out of the device during each T_{step} interval and removing or introducing electrons to the contact region with every charge assignment. If the nett charge in the contact region is positive, electrons will be introduced to these regions to compensate for the lack of electrons, and vice verse for a nett negative charge. The electrons are injected with a Maxwellian velocity distribution, with a resultant velocity component pointing into the semiconductor. Other distributions can also be considered, for example a velocity-weighted Maxwellian distribution as proposed by González *et al* [52] for sub-micron devices.

2.2.1 The Energy Band

The energy band diagram relates an electron's energy to its momentum. Energy bands are calculated by solving Schrödinger's equation [53]. The band structure for GaAs is shown in Figure 2.1. Although both the valence and conduction bands are shown, only the conduction bands need to be considered for the study of electron dynamics, since electrons in the valence bands are bound. Energy bands are extremely complex structures and the implementation of full energy bands involves high computational cost. However, under certain conditions, simplifications could be made to the full band model to reduce the burden on the computation resources.

For expected electron energies in realistic device simulations the band structure displays three distinct "valleys", with minima at the Γ, L and X points respectively. The extent of band

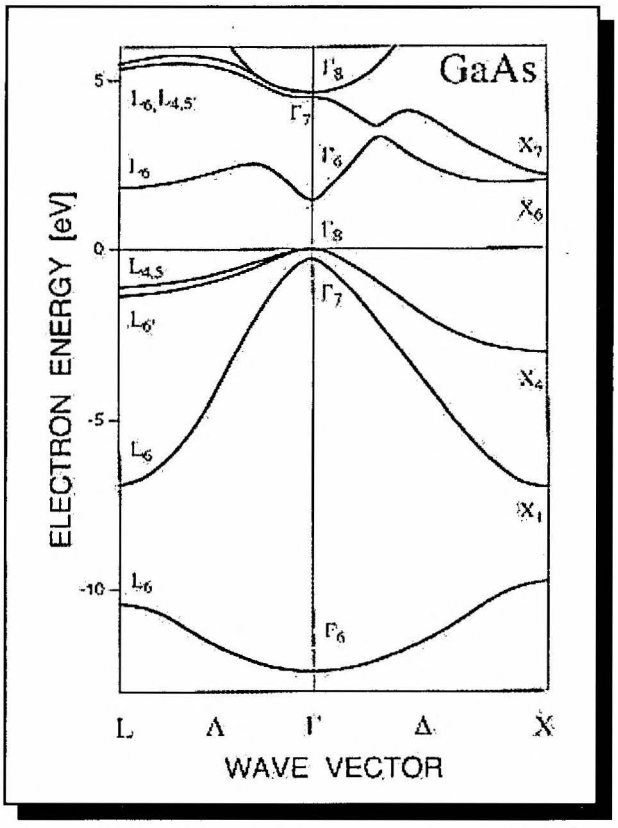


Figure 2.1: Energy band structure for GaAs (taken from [54])

structure simplification depends on the application of the simulation. For the study of electron transport, the information near the local band minima is important since electrons are usually located near the bottom of the valleys. For low electron energies, relative to these band minima, the band structure can be approximated by a parabolic *E-k* relation. Furthermore, for moderate electric fields ($< 3.10^6$ V/m), electrons gain insufficient energy in the steady state to populate the (higher energy) *X*-valley. The *X*-valley could therefore be ignored in such cases, without an unacceptable deterioration of the simulation. (Successful two-valley simulations at fields as high as 5.10^6 V/m have been reported in the literature, for example [55].) However, the simulation of high-field Gunn-diodes necessitates the inclusion of the *X*-valley in the simulation model. This is especially true for the simulation of the hot-injection launcher Gunn diodes proposed in this work.

Another factor to be considered is the non-parabolicity of the various valleys. The valley minima behave progressively more non-parabolically with increasing energy. These effects have to be

taken into account in high-field simulations. In conclusion, a three-valley, non-parabolic band structure has been implemented in the work presented here. This approximated band model is illustrated in Figure 2.2.

In terms of this non-parabolic approximation, the kinetic energy E of an electron in each valley is given by the simple relation [3]

$$E(1+\alpha E) = \frac{\hbar^2 k^2}{2m^*} \tag{2.4}$$

with k the magnitude of the wave vector, m^* the density of states effective mass of the electron and α the non-parabolicity factor associated with the valley it resides in.

It is worth noting that, in the case of GaAs, surfaces of constant energy in state space are ellipsoidal [5] and not spherical as is implied by (2.4). This is illustrated in the three-dimensional view of the Brillouin zone in Figure 2.3 where the eight L-valleys and six X-valleys together with the central Γ-valley are shown.

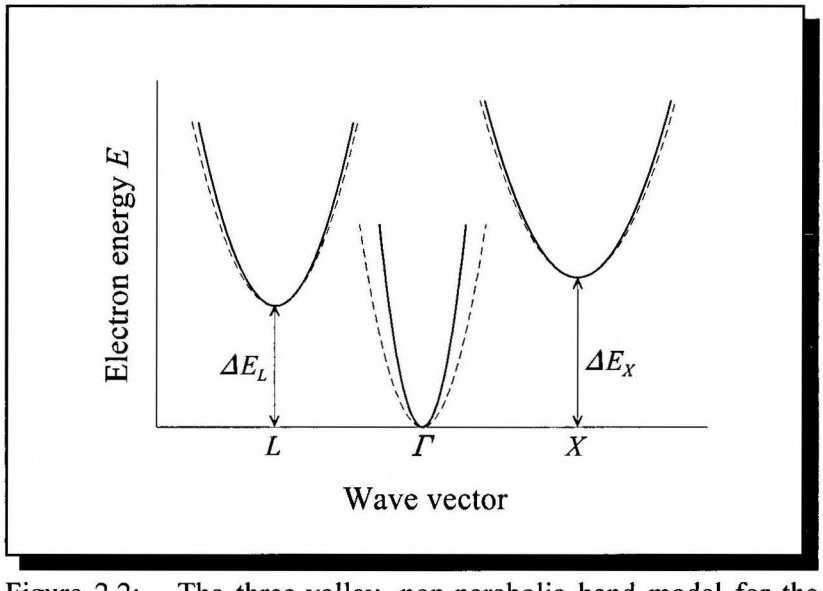


Figure 2.2: The three-valley, non-parabolic band model for the conduction band of GaAs used in this thesis. The effect of non-parabolicity is indicated by the broken lines. The energy separations ΔE_L and ΔE_X are clearly indicated. The Γ-valley minimum is taken as the reference.

These ellipsoids are described by [3]

$$E(1+\alpha E) = \frac{\hbar^2}{2}\left(\frac{k_l^2}{m_l^*} + \frac{k_t^2}{m_t^*}\right) \tag{2.5}$$

where k_l and k_t are the magnitude of the wave vector components defined, respectively, parallel (longitudinal) and transverse to the long axis of the ellipsoid (see Figure 2.3). The density of states effective mass in this case is not a scalar as in the spherical case, but a tensor with longitudinal and transverse components m_l^* and m_l^*.

The implication on the computational resources of using the ellipsoidal description for the energy band is quite staggering. Both formulations have been implemented with equally good results. The use of the more elaborate ellipsoidal band description could therefore not be justified for the diode simulations presented in this work, especially when seen against the much higher cost in terms of simulation time. The spherical band formulation is, in fact, commonly used [2].

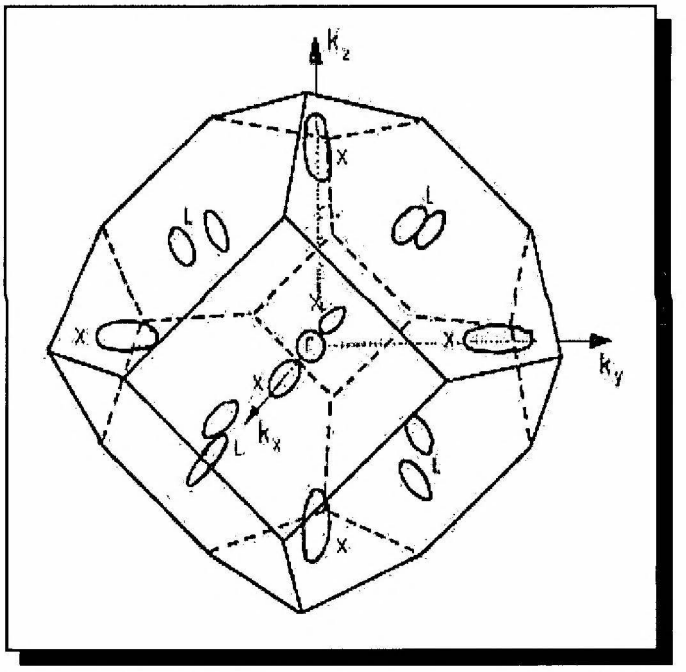


Figure 2.3: Three-dimensional view of the Brillouin zone for the GaAs crystal showing surfaces of constant energy. The centre valley has a spherical constant energy surface while the other valleys display ellipsoidal constant energy surfaces. (From [3]).

2.2.2 Electron Scattering

In a perfect periodic crystal potential an electron would have experienced a constant acceleration in a uniform electric field. Various mechanisms, however, perturb this periodic potential which is manifested as scattering of the electron. The most important scattering mechanisms usually considered are scattering due to electron-phonon interactions, ionised-impurity atoms and other electrons.

For the purposes of the present work, the following scattering processes have been taken into account:

- Acoustic phonon scattering (intra-valley scatter)
- Polar optical phonon scattering (intra-valley scatter)
- Intervalley scattering (induced by both acoustic and optical-mode phonons)
- Ionised-impurity scattering (intra-valley scatter)

Although an in-depth discussion of electron scattering mechanisms falls outside the scope of this work, two factors that have direct bearing on the Monte Carlo simulation will be discussed in the next two paragraphs, namely the *rate* of a specific scattering mechanism and its *effect* on the state of the electron after the collision.

2.2.2.1 Scattering rates

The importance of a scattering mechanism is determined by its rate, since a high rate relative to the other scattering rates increases the possibility that a certain mechanism will take place. The rates of the abovementioned mechanisms will now be discussed. The numerical values of the material parameters for GaAs is given in Section 2.2.8.

- *Acoustic phonon scattering*

The rate of acoustic phonon scattering is given by [2]

$$\lambda_{ac} = \frac{2^{1/2} K_B T \Xi_{ac}^2 \left(m_t^2 m_l\right)^{1/2}}{\pi \hbar^4 v_s^2 \rho} \frac{\sqrt{E(1+aE)}}{(1+2aE)} \left((1+aE)^2 + \frac{1}{3}(aE)^2 \right) \tag{2.6}$$

with K_B Boltzmann's constant, T the crystal temperature (in Kelvin), Ξ_{ac} the acoustic deformation potential, v_s the speed of sound in crystal and ρ the density of crystal. Acoustic scattering is assumed to be elastic due to the low phonon energy compared to the mean electron energy at room temperature.

- *Polar optic phonon scattering*

The polar optic phonon scattering rate is given by [3]

$$\lambda_{po} = \frac{q^2 \omega_{op} \left(m_t^2 m_l\right)^{1/6}}{2^{5/2} \hbar \pi \varepsilon_0} \left(\frac{1}{\varepsilon_\infty} - \frac{1}{\varepsilon_s} \right) \frac{1+2\alpha E'}{\gamma^{1/2}(E)} F_0 F_1 \tag{2.7}$$

with

$$\begin{aligned} F_0 &= \frac{1}{C} \left(A \ln \left| \frac{\gamma^{1/2}(E) + \gamma^{1/2}(E')}{\gamma^{1/2}(E) - \gamma^{1/2}(E')} \right| + B \right), \\ A &= \left(2(1+2\alpha E)(1+\alpha E') + \alpha \left(\gamma(E) + \gamma(E') \right) \right)^2, \\ B &= -2\alpha \gamma^{1/2}(E) \gamma^{1/2}(E') \left(4(1+\alpha E)(1+\alpha E') + \alpha \left(\gamma(E) + \gamma(E') \right) \right), \\ C &= 4(1+\alpha E)(1+\alpha E')(1+2\alpha E)(1+2\alpha E') \end{aligned} \tag{2.8}$$

and q the unit electronic charge, ω_{op} the polar optic phonon angular frequency, ϵ_0 the permittivity of a vacuum and ϵ_∞ and ϵ_s the high frequency and static dielectric constants , respectively, of the crystal. Polar optical phonon scattering is inelastic due to the high energy of the phonons involved.

For phonon absorption

$$\begin{aligned} F_1 &= \left(\exp\left(\frac{\hbar \omega_{op}}{K_B T} \right) - 1 \right)^{-1}, \\ E' &= E + \hbar \omega_{op} \end{aligned} \tag{2.9}$$

and for phonon emission

$$F_1 = \left(\exp\left(\frac{\hbar\omega_{op}}{K_B T}\right) - 1\right)^{-1} + 1\,, \qquad E' = E - \hbar\omega_{op}\,. \tag{2.10}$$

The condition $E' > 0$ must always be adhered to. If the electron's initial energy is less than that of the phonon, the scattering rate is set to zero.

- *Intervalley scattering*

The rate of intervalley transitions from the initial valley i to the final valley j is given by [6]

$$\lambda_{if} = \frac{Z_f \Xi_{if}^2 \left(m_t^2 m_l\right)_f^{1/2}}{2^{1/2}\omega_{if}\pi\rho\hbar^3}\left(E'\left(1+\alpha_f E'\right)\right)^{1/2}\left(1+2\alpha_f E'\right)F_{ij}(E,E')F_1\,, \qquad F_{if} = \left(\frac{1+\alpha_i E_i}{1+2\alpha_i E_i}\right)\left(\frac{1+\alpha_f E_f'}{1+2\alpha_f E_f'}\right). \tag{2.11}$$

with Z_f the number of equivalent valleys the electron can scatter into, Ξ_{if} the intervalley deformation potential field and ω_{if} the angular frequency of the phonon causing the intervalley transition. Intervalley transitions are inelastic. For phonon absorption

$$F_1 = \left(\exp\left(\frac{\hbar\omega_{if}}{K_B T}\right) - 1\right)^{-1}, \qquad E' = E + \hbar\omega_{if} - \left(\Delta E_f - \Delta E_i\right), \tag{2.12}$$

and for phonon emission

$$F_1 = \left(\exp\left(\frac{\hbar\omega_{if}}{K_B T}\right) - 1\right)^{-1} + 1, \qquad E' = E - \hbar\omega_{if} - \left(\Delta E_f - \Delta E_i\right), \tag{2.13}$$

with ΔE_i and ΔE_f the energy gaps between a fixed reference point and the minima of the initial (i) and final (f) valleys respectively. If the condition $E' > 0$ is violated, $\lambda_{if} = 0$ and that particular intervalley scattering will not take place.

- *Ionised-impurity scattering*

The rate of scatter due to ionised impurity atoms is given by [3]

$$\lambda_{imp} = \frac{2^{1/2}\left(K_B T\right)^2 \left(m_t^2 m_l\right)^{1/2}}{\pi N_I \hbar^4} \frac{\left(1+2\alpha E\right)}{1+\dfrac{8\left(m_t^2 m_l\right)^{1/3} E\left(1+\alpha E\right)}{\hbar^2 \xi_\beta^{\,2}}} \left(E\left(1+\alpha E\right)\right)^{1/2} \tag{2.14}$$

with

$$\xi_\beta^{\,2} = \frac{q^2 N_I}{\varepsilon_r \varepsilon_0 K_B T} . \tag{2.15}$$

Ionised-impurity scattering is assumed to be elastic.

2.2.2.2 State of electron after scattering

The influence of a scattering mechanism on the state of an electron is determined subject to energy and momentum conservation principles. The state of the electron immediately after a scattering event is completely defined by determining the magnitude and direction of the electron's wave vector.

It is convenient to define two frames of reference for the determination of the updated state (after scattering) of the electron, namely a fixed laboratory $(x,y,z)_L$ frame and a relative $(x,y,z)_R$ frame. For brevity, these will be simply referred to as the L-frame and R-frame respectively. The z-axis of the R-frame is defined parallel to the wave vector prior to scattering, $\boldsymbol{k}$. These two frames are illustrated in Figues 2.4 and 2.5.

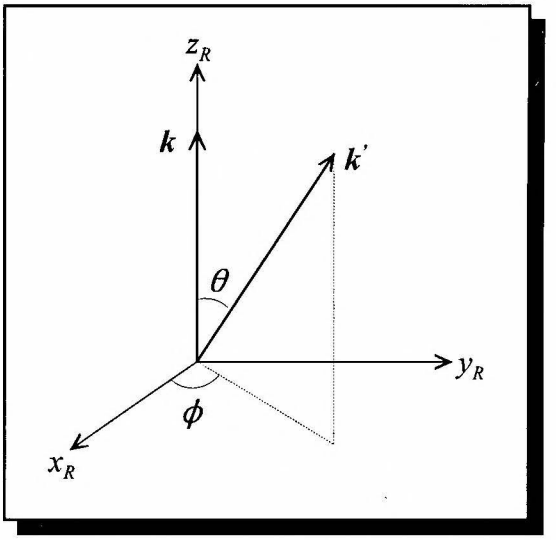


Figure 2.4: Definition of the R-frame and the relation between $\boldsymbol{k}$ and $\boldsymbol{k}'$ in terms of the polar angle θ and azimuth angle ϕ.

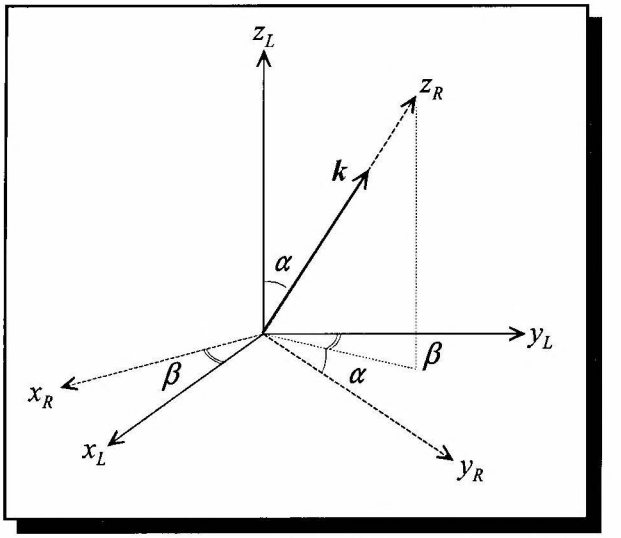


Figure 2.5: Relationship between the relative R- and fixed L-frames. The R-frame is obtained by rotating the L-frame through α about the x_L-axis, and through β about the z_L-axis.

The influence of each of the abovementioned scattering mechanisms on the state of the electron will now be discussed.

- *Acoustic phonon scattering*

Acoustic phonon scattering is isotropic. The direction of $\boldsymbol{k}'$ is therefore random in state space and independent of the direction of $\boldsymbol{k}$. The polar and azimuthal angles determining the direction of $\boldsymbol{k}'$ can thus be defined *directly* in the L-frame.

The angles θ and ϕ are determined by [5]

$$\begin{aligned} \cos\theta &= 1 - 2r_1 \,, \\ \varphi &= 2\pi r_2 \end{aligned} \tag{2.16}$$

with r_1 and r_2 two random numbers distributed evenly between 0 and 1. The components of $\boldsymbol{k}'$ in the L-frame are readily obtained in terms of θ and ϕ as

$$\boldsymbol{k}' = \begin{bmatrix} k' \sin\theta\cos\varphi \\ k' \sin\theta\sin\varphi \\ k' \cos\theta \end{bmatrix}_L \tag{2.17}$$

where the magnitude k' of the wave vector after scattering is determined through (2.4).

- *Polar optic phonon scattering*

Polar optic phonon scattering is non-isotropic. The wave vector after scattering $\boldsymbol{k}'$ is therefore first determined in the R-frame and then translated to the L-frame. The polar and azimuthal angles defining the direction of $\boldsymbol{k}'$ in the R-frame are determined by [5]

$$\cos\theta = \frac{1+f-(1+2f)^{r_1}}{f}, \qquad \varphi = 2\pi r_2 \tag{2.18}$$

respectively, with

$$f = \frac{2(EE')^{1/2}}{\left(E^{1/2} - E'^{1/2}\right)^2} \tag{2.19}$$

and r_1 and r_2 two random numbers distributed evenly between 0 and 1. The components of $\boldsymbol{k}'$ in the L-frame are readily obtained by the translation from the R-frame to the L-frame given by

$$\begin{bmatrix} k_x' \\ k_y' \\ k_z' \end{bmatrix}_L = \begin{bmatrix} \cos\beta & \cos\beta\sin\beta & \sin\alpha\sin\beta \\ -\sin\beta & \cos\alpha\cos\beta & \sin\alpha\cos\beta \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \cdot \begin{bmatrix} k' \sin\theta\cos\varphi \\ k' \sin\theta\sin\varphi \\ k' \cos\theta \end{bmatrix} \tag{2.20}$$

with

$$\sin\alpha = \frac{\sqrt{k_x^2 + k_y^2}}{k}, \qquad \cos\alpha = \frac{k_z}{k}$$
$$\sin\beta = \frac{k_x}{\sqrt{k_x^2 + k_y^2}}, \qquad \cos\beta = \frac{k_y}{\sqrt{k_x^2 + k_y^2}} \tag{2.21}$$

and k and k', respectively, the magnitude of the wave vector before and after scattering. The

latter is related to the electron energy before and after scattering through (2.4). The components of the wave vector before scattering in (2.21) are defined in the L-axis.

- *Intervalley scattering*

Intervalley scattering is also isotropic. A procedure similar to that of acoustic phonon scattering is followed to determine the state after scattering. The magnitude of the wave vector k' is determined through (2.4), together with (2.16) and (2.17).

- *Ionised-impurity scattering*

Ionised-impurity scattering is anisotropic. A procedure similar to that of polar optic phonon scattering is followed to determine the state after scattering. In this case, the polar and azimuthal angles are given by [5]

$$\cos\theta = 1 - \frac{2(1-r_1)}{1+4r_1\left(\frac{k}{\xi_\beta}\right)^2}, \qquad \varphi = 2\pi r_2 \tag{2.22}$$

with r_1 and r_2 two random numbers distributed evenly between 0 and 1. The magnitude of the wave vector k is determined by (2.4).

2.2.2.3 Selection of scattering event

As stated earlier, the probability of a certain scattering event taking place is dependent on its scattering rate relative to the respective rates of the other possible events. A simple procedure can therefore be used to select a scattering event based on the generation of a random number.

The respective rates λ_i of all possible scattering events are calculated for the electron energy immediately prior to the scattering event and added successively. A random number r, distributed evenly between 0 and λ_T, is generated and the n^{th} scattering event is chosen if

$$\sum_{i=1}^{i=n-1} \lambda_i(E) < r \leq \sum_{i=1}^{i=n} \lambda_i(E) \tag{2.23}$$

with $i=1,2,3,...N$ all possible scattering events.

2.2.2.4 Tabulation of scattering rates

The determination of the scattering rates, which involves the calculation of complex mathematical expressions after every free flight for each electron, represents a significant portion of the overall computational activity of the simulation. It is therefore customary to tabulate the respective rates as a function of energy. In the simulations presented in this work the rates are not merely functions of energy, but also of material composition (relevant to the heterostructures) and temperature. The scattering rates have subsequently been tabulated in a multidimensional array as:

$$[\lambda] = \left[[i][j][k]\right]$$

with i = 1,2,...5000 the sampled energy values taken from 0 to 3eV,

j = 1,2,...15 the sampled mole fraction x taken form 0 to 0.3,

k = 1,2 ...5 the sampled temperature values taken from 350K to 500K.

2.2.3 Electron dynamics

The dynamics of an electron in free flight, in the absence of a magnetic field, is described by [5]

$$\frac{d\boldsymbol{k}}{dt} = -\frac{1}{\hbar}\nabla H \tag{2.25}$$

and

$$\frac{d\boldsymbol{r}}{dt} = \frac{1}{\hbar}\nabla_k H \tag{2.26}$$

where ∇ and ∇_k are, respectively, del operators with respect to the position vector $\boldsymbol{r}$ and the wave vector $\boldsymbol{k}$. The Hamiltonian H is chosen as

$$H = E_k + E_c(\boldsymbol{r}) \tag{2.27}$$

with E_k the kinetic energy of the electron and $E_c(\boldsymbol{r})$ the conduction band minimum given by

$$E_c(\boldsymbol{r}) = \text{constant} - \chi(\boldsymbol{r}) + q\Phi(\boldsymbol{r}) \tag{2.28}$$

where $\chi(\boldsymbol{r})$ is the electron affinity, q the unit electronic charge (defined with negative value) and $\Phi(\boldsymbol{r})$ the electrostatic potential.

It is clear from the above that, whilst in free flight, the electron is treated as a classical point-like particle. However, it can be shown that this semi-classical approach is valid for average free flight times τ greater than $10e^{-13}$ s to $10e^{-14}$ s [3] or when considering devices with dimensions in the micrometer range [2]. Both of these requirements are met in the context of the work presented here.

In the case of compositional uniform regions, $\chi(\boldsymbol{r})$ is constant and can therefore be ignored in terms of the free flight modelling of the electron. The effect of non-constant electron affinity on the electron dynamic behaviour will be discussed in Section 2.2.4.

In line with the one-dimensional simulations executed in this work, the solution (2.25) and (2.26) for a one-dimensional description of the electron dynamics will now be discussed.

The free flight dynamic behaviour of the electron in $\boldsymbol{k}$- space is described by substituting (2.27) and (2.28) into (2.25). This gives, for argument's sake movement in the y-direction,

$$\frac{dk_y}{dt} = \frac{q}{\hbar}\xi_y \tag{2.29}$$

with k_y and ξ_y, respectively, the y-components of the wave vector and electric field vector. Integrating (2.29) with respect to time gives the wave vector at the end of a free flight τ as

$$k_y(\tau) = \frac{q\xi_y}{\hbar}\tau + k_y(0) \tag{2.30}$$

where $k_y(0)$ is the initial state of the electron. The electric field vector is assumed to be constant throughout the flight.

The free flight dynamic behaviour of the electron in real space is determined by substituting (2.27), (2.28) and (2.4) for the kinetic energy of the electron into (2.26).

This gives, after manipulation, the electron velocity v_y in the y-direction as

$$v_y = \frac{\hbar k_y}{\left(1+2\alpha E_k\right)m^*} \cdot \tag{2.31}$$

It is clear from (2.31) that the effect of non-parabolicity is to slow down the electron through an increase in the electron's effective mass for increasing kinetic energy.

To obtain the electron's positional displacement as a function of time (2.31) has to be integrated with respect to time. However, since both k_y and E_k are functions of time, this is not straightforward. To simplify the procedure it can be assumed that, for short periods of free flight,

$$v(\tau/2) = \frac{\hbar k_y(\tau/2)}{\left(1+2\alpha E_k(\tau/2)\right)m^*} \approx \frac{y(\tau)-y(0)}{\tau} \tag{2.32}$$

with

$$k_y(\tau/2) = \frac{k_y(\tau)+k_y(0)}{2} \tag{2.33}$$

and

$$E_k(\tau/2) = -\frac{1}{2\alpha} + \frac{1}{2\alpha}\sqrt{1+\frac{2\alpha\hbar^2}{m^*}\left(k_x^2(\tau/2)+k_z^2(\tau/2)+k_y^2(\tau/2)\right)} \cdot \tag{2.34}$$

In (2.34) $k_x(\tau/2) = k_x(0)$ and $k_z(\tau/2) = k_z(0)$ because it is assumed that there is no acceleration in these directions during free flight for the one-dimensional case.

2.2.4 Heterostructure modelling

The heterostructures employed in this thesis all have the same generic form shown in Figure 2.6. The Al mole fraction x (not to be confused with dimension x) is graded with slope $p = x_H/y_H$ in the direction of the electron flow. The heterostructure terminates abruptly at $y = y_H$. Modelling the electron dynamics in the heterostructure requires both a classical and a quantum mechanical treatment. In the graded region the electron is treated classically, whereas quantum mechanical processes govern the boundary conditions at the edge.

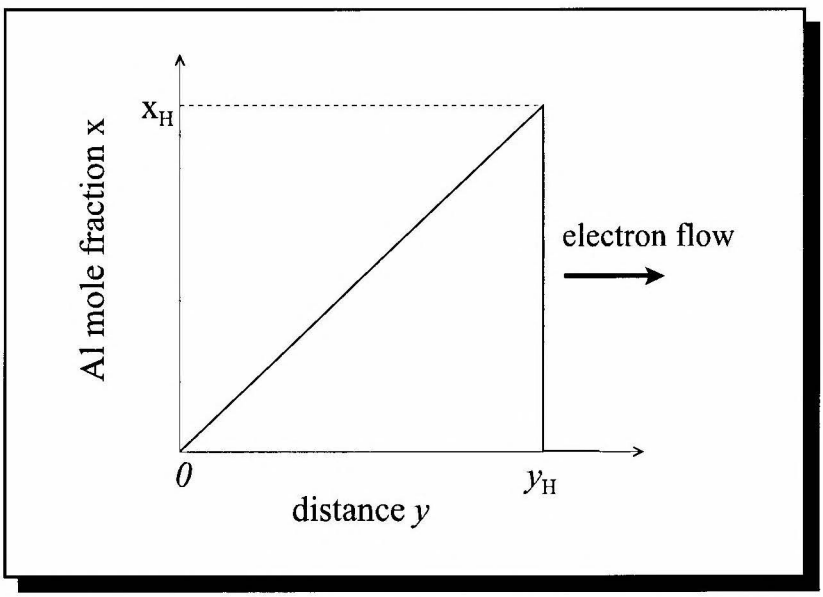


Figure 2.6: A graded heterostructure with linearly increasing Al mole fraction x in the direction of the electron flow.

2.2.4.1 Electron dynamics in the graded region $0 \le y < y_H$

The dynamic description of the electron is still described by (2.25) and (2.26). However, $\chi(\mathbf{r})$ is non-uniform and of the form

$$\chi(y) = \text{constant} - \chi_f p\, y \tag{2.35}$$

for the one-dimensional case with χ_f an appropriate factor (see code listing "dynamix_x.h"). The solution of (2.25), (2.27) and (2.28) now yields

$$\frac{dk_y}{dt} = -\frac{\chi_f p}{\hbar} + \frac{q}{\hbar}\xi_y \tag{2.36}$$

from which it is clear that the graded x-fraction produces an additional acceleration term.

2.2.4.2 Transport of electrons across the potential step of the heterostructure

The modelling of the movement of electrons across potential barriers has been dealt with extensively in the literature, e.g. [56], [57] and [58] to mention but a few. The following discussion follows the treatment by Wu and Yang [59].

To clarify the model, the abrupt edge of the heterostructure is modelled as the potential barrier shown in Figure 2.7 in the vicinity of the step. It is convenient to define two energies, namely

$$E_{//}\left(1+\alpha E_{//}\right) = \frac{\hbar^2 k_y^2}{2m^*} \tag{2.37}$$

and

$$E_{\perp}\left(1+\alpha E_{\perp}\right) = \frac{\hbar^2\left(k_x^2 + k_z^2\right)}{2m^*} \tag{2.38}$$

which are, respectively, the kinetic energy of the electron associated with the wave vectors parallel and perpendicular to direction of current flow across the barrier.

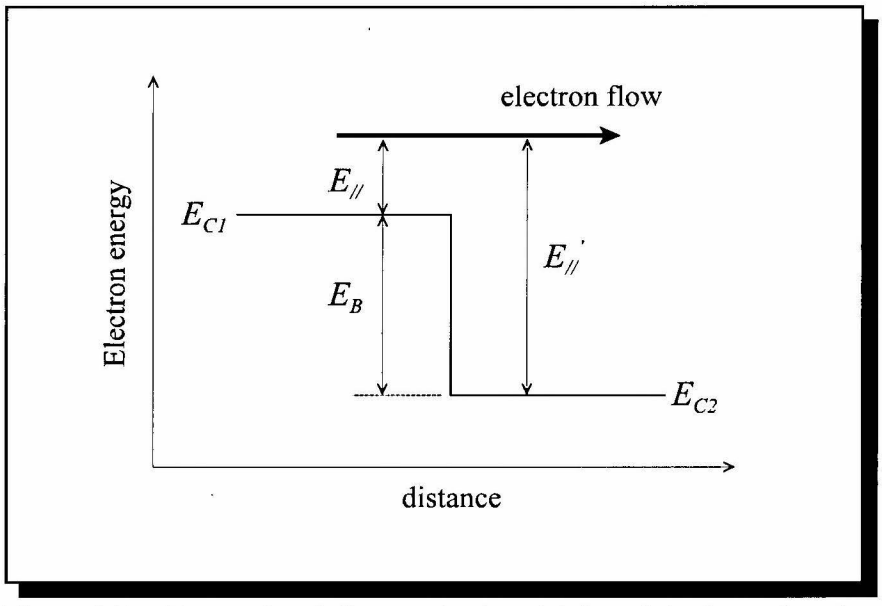


Figure 2.7: Energy band diagram in the vicinity of the heterojunction interface. E_{C1} and E_{C2} are, respectively, the conduction band minima to the left and right of the junction with $E_B = E_{C1} - E_{C2}$ the magnitude of the step.

The conservation of total energy and perpendicular momentum yields

$$E_{//} + E_{\perp} = E_{//}' + E_{\perp}' - E_B \tag{2.39}$$

and

$$\begin{aligned} E_{\perp}' &= \frac{1}{\varsigma} E_{\perp} , \\ \varsigma &= \frac{m^{*'}}{m^*} \end{aligned} \tag{2.40}$$

where the prime indicates the state after crossing the barrier. The factor $\varsigma \neq 1$ since the effective mass is dependent on the Al mole fraction x. Substitution of (2.40) into (2.39) and applying (2.37) gives

$$\begin{aligned} k_y' &= \left(\frac{2m^{*'}}{\hbar^2} K(\alpha K + 1) \right)^{1/2} , \\ K &= E_{\perp}\left(1 - \frac{1}{\varsigma}\right) + E_{//} + E_B \end{aligned} \tag{2.41}$$

where the positive root is selected for k_y' since the electron is accelerated in the positive y-

direction. To determine the wave vector components k_x' and k_z' (2.38) is substituted into (2.40) which gives

$$k_x'^2 + k_z'^2 = \frac{2m^{*'}}{\hbar^2} K(\alpha K + 1), \qquad K = \frac{\hbar^2}{2m^{*'}}\left(k_x^2 + k_z^2\right) \tag{2.42}$$

from which it follows that

$$k_x'^2 + k_z'^2 = \left(k_x^2 + k_z^2\right)\left(\frac{\alpha\hbar^2}{2m^{*'}}\left(k_x^2 + k_z^2\right) + 1\right) \approx k_x^2 + k_z^2 \tag{2.43}$$

under the assumption that the energy associated with the perpendicular wave vectors is small enough for non-parabolicity to be negligible (α=0). This assumption is valid since there is no electric field, and therefore no acceleration, in these directions in one-dimensional simulations. The perpendicular energy can thus be assumed to be of the order

$$\frac{2}{3} k_B T \approx 26\, meV \tag{2.44}$$

for T = 460K due to thermal excitations. Equation (2.43) does not specify the perpendicular wave vector components k_x' and k_z' uniquely. For simplification it is assumed that these components are unaffected by the potential step, yielding

$$k_x' = k_x, \qquad k_z' = k_z. \tag{2.45}$$

This completes the determination of the electron state immediately after crossing the abrupt junction. However, the probability that an electron will in fact traverse the junction still needs to be addressed.

The quantum mechanical transmission coefficient (*QMT*) is defined as the ratio of transmitted current to the incident. Solving Schrödinger's equation across the barrier, subject to appropriate boundary conditions at the junction edge [56], yields

$$QMT = \frac{4 k_y k_y'}{\left(k_y + k_y' \right)^2} \tag{2.46}$$

which can be written in terms of $E_{//}$ in a straightforward manner by applying (2.37) to (2.40).

The solution of the *QMT* is clearly time consuming. For practical energy values, however, the *QMT* is approximately unity as is evident from the graph in Figure 2.8. This is even more true for practical junctions which tend to be rounded (private conversation with John Davies, author of [56]). Nevertheless, (2.46) has been implemented in the work presented here.

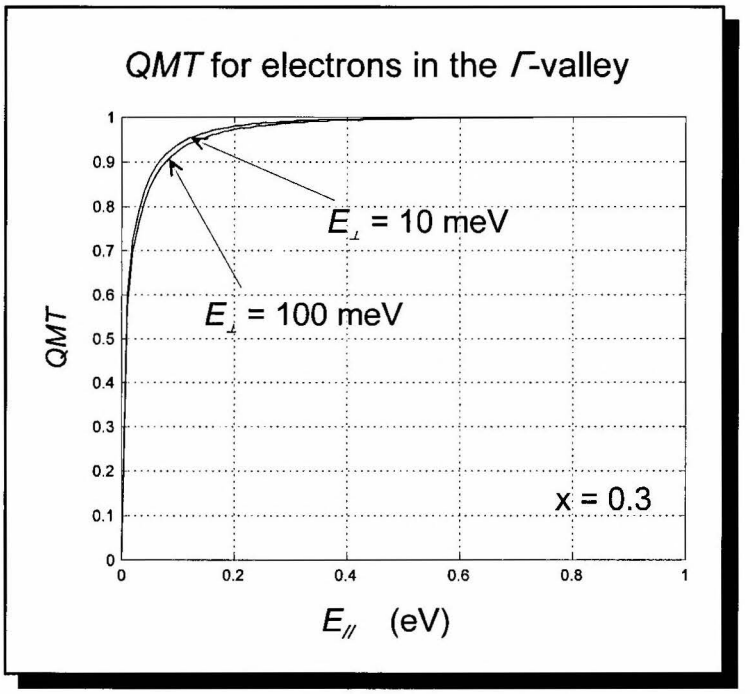


Figure 2.8: A graph of the quantum mechanical transmission coefficient as a function of $E_{//}$ for an electron crossing a heterojunction edge from $Al_{0.3}GaAs_{0.7}$ to GaAs. For both cases $E_{\perp}$=10meV and $E_{\perp}$=100meV $QMT \approx 1$.

2.2.5 The Field Equation in one dimension

The internal electric field and charge distributions are related through Poisson's equation which is given by [50]

$$\nabla \cdot (\varepsilon_r \varepsilon_0 \boldsymbol{\xi}) = \rho \tag{2.47}$$

where $\boldsymbol{\xi}$ and ρ are, respectively, the electric field vector and nett charge distributions. The incorporation of heterostructures in the simulation area results in a position-dependent relative dielectric constant ϵ_r. Therefore, expanding (2.47) gives

$$\varepsilon_r (\nabla \cdot \boldsymbol{E}) + \boldsymbol{E} \cdot (\nabla \varepsilon_r) = \frac{\rho}{\varepsilon_0} \quad . \tag{2.48}$$

When considering one-dimensional simulations in, say, the y-direction, (2.48) reduces to

$$\varepsilon_r \frac{d\xi_y}{dy} + \xi_y \frac{d\varepsilon_r}{dy} = \frac{\rho}{\varepsilon_0} \tag{2.49}$$

with E_y the y-component of the field vector. It is assumed implicitly in (2.49) that ϵ_r, E_y and ρ are functions of y. As indicated in section 2.2.8, ϵ_r is a linear function of y with slope $d\epsilon_r/dy=k_{\epsilon 2}$ within the heterostructures considered in this work and constant elsewhere in the device. By applying this, (2.49) can be rewritten as

$$\varepsilon_r \frac{d\xi_y}{dy} - p \cdot \xi_y = \frac{\rho}{\varepsilon_0} \tag{2.50}$$

with $p=k_{\epsilon 2}$ within the heterostructures and $p=0$ in the rest of the device. In one dimension the electric field distribution is related to the internal potential distribution $\Phi(y)$ through [50]

$$\xi_y = -\frac{d\Phi}{dy} \quad . \tag{2.51}$$

Substituting (2.51) into (2.50) gives

$$-\varepsilon_r \frac{d^2\Phi}{dy^2} + p\frac{d\Phi}{dy} = \frac{\rho}{\varepsilon_0} \tag{2.52}$$

which is to be solved for the internal potential distribution.

2.2.5.1 Numerical solution of the field equation

In order to solve (2.52) numerically for the internal potential distributions, the device is divided into a one-dimensional grid with uniform grid spacings. This is graphically illustrated in Figure 2.9.

The discretisation of (2.52) is obtained by applying a three point difference scheme for the first and second derivatives. The resulting difference equation is

$$\left(\frac{p_i}{2\Delta y}-\frac{\varepsilon_{ri}}{\Delta y^2}\right)\Phi_{i+1}+\left(\frac{2\varepsilon_{ri}}{\Delta y^2}\right)\Phi_i+\left(-\frac{p_i}{2\Delta y}-\frac{\varepsilon_{ri}}{\Delta y^2}\right)\Phi_{i-1}=\frac{\rho_i}{\varepsilon_0} \tag{2.53}$$

with p_i, ϵ_{ri} and Φ_i, respectively, the values of p, ϵ_r and Φ at gridpoint i and Δy the uniform grid spacing shown in Figure 2.9.

To generate a unique solution for (2.53), two boundary conditions, at $y=0$ and $y=L_{diode}$, will be applied. The potential at each contact is enforced by the external circuitry. It can therefore be assumed that Φ_0 and $\Phi_{N\rho}$ are both known. The solution of the set of linear equations represented

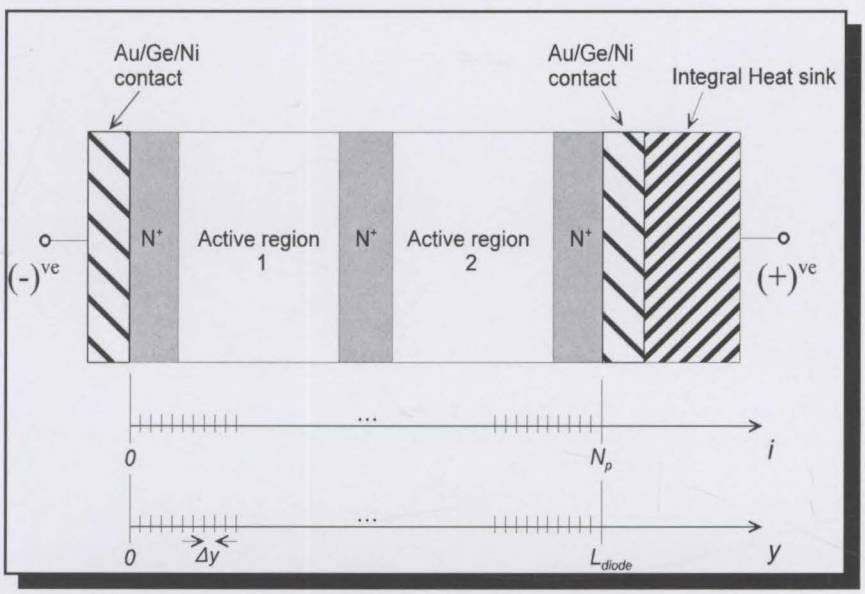


Figure 2.9: A graphic illustration of the discretisation of the diode into a grid of N_ρ uniform spacings.

by (2.53) for the potentials $\Phi_1 \ldots \Phi_{N\rho-1}$ through Gauss elimination [49] will now be discussed briefly.

Rewriting (2.53) in the form

$$\Phi_{i-1} + a_i \Phi_i + b_i \Phi_{i+1} = c_i \tag{2.54}$$

for $i=1,2 \ldots N_{\rho-1}$ and with

$$\begin{aligned} a_i &= -\frac{4\varepsilon_{ri}}{p_i \Delta y + 2\varepsilon_{ri}} \\ b_i &= -\frac{p_i \Delta y - 2\varepsilon_{ri}}{p_i \Delta y + 2\varepsilon_{ri}} \\ c_i &= -\left(\frac{2\Delta y^2}{p_i \Delta y + 2\varepsilon_{ri}}\right)\frac{\rho_i}{\varepsilon_0} \end{aligned} \tag{2.55}$$

and applying the boundary conditions gives, after appropriate matrix reduction, the tri-diagonal matrix equation

$$\begin{bmatrix} 1 & w_1 & 0 & & & \cdots & & 0 \\ 0 & 1 & w_2 & 0 & & & \iddots & \\ & & & \ddots & & & & \vdots \\ & & 0 & 1 & w_i & 0 & & \\ \vdots & & & & & \ddots & & \\ & \iddots & & & & & 1 & w_{N_\rho-2} \\ 0 & & \cdots & & & & 0 & 1 \end{bmatrix} \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \vdots \\ \Phi_i \\ \vdots \\ \Phi_{N_\rho-2} \\ \Phi_{N_\rho-1} \end{bmatrix} = \tag{2.56}$$

$$\begin{bmatrix} g_1 & g_2 & \cdots & g_i & \cdots & g_{N_\rho-2} & g_{N_\rho-1} \end{bmatrix}^T$$

with

$$
\begin{aligned}
w_1 &= \frac{b_1}{a_1} \\
w_i &= \frac{b_i}{a_i - w_{i-1}} \qquad \text{for} \quad i = 2,3,\ldots N_\rho - 2 \\
g_1 &= \frac{c_1 - \Phi_0}{a_1} \\
g_i &= \frac{c_i - g_{i-1}}{a_i - w_{i-1}} \qquad \text{for} \quad i = 2,3,\ldots N_\rho - 2 \\
g_{N_\rho-1} &= \frac{c_{N_\rho-1} - b_{N_\rho-1}\Phi_{N_\rho}}{a_{N_\rho-1} - w_{N_\rho-2}}
\end{aligned}
\tag{2.57}
$$

The unknown potential values $\Phi_1 \ldots \Phi_{N\rho-1}$ are obtained by a simple backward substitution routine:

$$
\begin{aligned}
\Phi_{N_\rho-1} &= g_{N_\rho-1} \\
\Phi_i &= g_i - w_i \Phi_{i+1} \quad \textit{for } i = N_\rho - 2,\ N_\rho - 3, \ldots 1
\end{aligned}
\tag{2.58}
$$

It is clear from (2.58) that the w_i parameters are dependent only on predefined geometric dimensions. They can therefore be calculated once at the beginning of the simulation and stored in a lookup table to save computing time. On the other hand, the g_i parameters are dependent on geometric dimensions as well as the charge density distribution at a certain moment in time. As a result, these parameters have to be recalculated with each field update.

2.2.5.2 Calculation of the electric field distribution

For the one-dimensional case the y-component of the field vector at each gridpoint i, ξ_{yi}, is calculated as

$$
\xi_{yi} = -\frac{\Phi_{i+1} - \Phi_{i-1}}{2\Delta y} \tag{2.59}
$$

for $i = 1, 2, \ldots N_\rho\text{-}1$. The electric field values on the contact boundaries are assumed to be zero in accordance with realistic ohmic contact conditions [52].

2.2.6 The steady-state heat flow equation in one dimension

The steady-state heat flow equation is given by [60]

$$\nabla(K\nabla T) + Q = 0 \tag{2.60}$$

where K represents the thermal conductivity of the material, Q the steady-state heat dissipation density within the device and T the temperature distribution.

For the solution of (2.60) Zybura *et al* [61] divided the device into layers, with the assumption that the thermal conductivity and heat generation *throughout* each of these layers are uniform. The active region, for instance, is treated as a single layer, i.e. uniform thermal conductivity and heat generation are assumed for the whole active region. However, this assumption of constant heat generation throughout the active region is crude. It is apparent from Batchelor [32], [33] that this is clearly not the case.

In this work each highly doped region in the diode is treated as a single layer due to the minimal heat generation occurring in these regions, as is also assumed by Zybura [61]. However, the active region is divided into N_{AR} subsections, each treated as a layer with uniform K and Q. This refinement to the model proposed by Zybura surely increases the computation time but will inevitably render a more realistic temperature profile throughout the active region, or regions in the case of multiple domain diodes.

Given the lower thermal conductivity of the surrounding air, it can be assumed that the heat flow will be concentrated within the device. It can further be assumed that all the heat generated in the device will flow through the device layers into the highly conductive heat sink. The thermal analysis can therefore be reduced to one dimension, in the y-direction, with no heat loss through the top of the mesa.

The heat equation (2.60) consequently reduces to

$$\frac{d}{dy}\left(K(y)\frac{dT(y)}{dy}\right) + Q(y) = 0 \ . \tag{2.61}$$

The numerical solution of (2.61) for the temperature profile throughout the diode is based on the scheme proposed by Zybura, which is outlined in the sections below.

2.2.6.1 Discretisation of the diode structure into sections of constant heat conductivity and heat generation.

For the numerical solution of (2.61) the device is divided into a one-dimensional grid of layers as illustrated in Figure 2.10. The metal contacts and heat sink are shown for the sake of completeness. Each active region is divided into a uniform mesh of N_{AR} layers. The highly doped regions are taken as single layers due to the minimal heat generation in them. The diode is divided into N_{TH} layers in total.

The resulting mesh should not be mistaken for the uniform grid defined earlier for the solution of Poisson's equation. As implemented, the grid used in the Poisson-solver is much finer than the grid defined here. Within each layer it is assumed that K and Q are constant. For clarity, K_i and Q_i refer, respectively, to the constant K and Q values associated with the i^{th} layer.

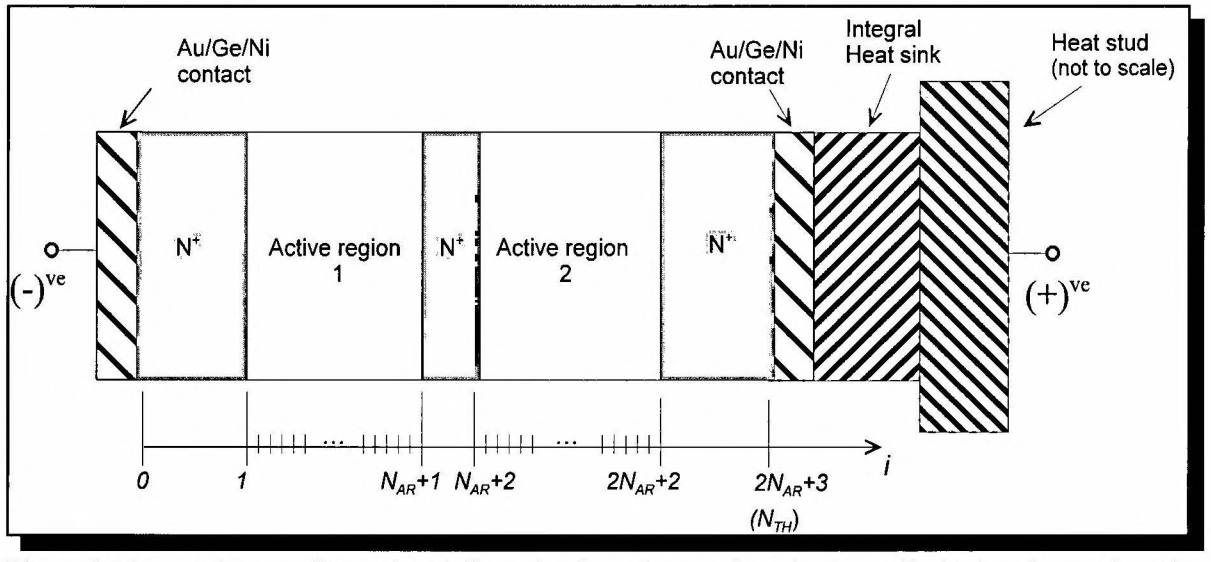


Figure 2.10: The one-dimensional discretisation of a two domain Gunn diode showing each active region divided into N_{AR} subsections. The diode is divided into N_{TH} layers in total.

2.2.6.2 Determination of the steady state power dissipation density Q in each layer

The steady state power density distribution $Q(y)$ within the device is given by (see for example [32])

$$Q(y) = J_{DC}(y) \cdot \xi_{DC}(y) \tag{2.62}$$

where $J_{DC}(y)$ and $\xi_{DC}(y)$ are, respectively, the d.c. components of the magnitude of the current density and electric field as a function of the position y in the device. In MC analyses these d.c. values are, however, not known at the beginning of the simulation. The d.c. components can only be estimated by the convergence of the time-average of the relevant variables over a long time. Mathematically this can be written as

$$J_{DC}(y) \approx \frac{1}{T_{sim}} \int_{t=0}^{T_{sim}} j(y,t)dt \tag{2.63}$$

where $j(y,t)$ is the time-evolution of the particle current at a certain point y and T_{sim} the total simulated time which is assumed to be long enough for convergence.

In time-discrete form (2.63) can be written as

$$J_{DC}(y) \approx \frac{1}{s} \sum_{l=1}^{s} j(y, l\Delta t_{TH}) \tag{2.64}$$

where s is the number of iterations and t_{TH} the time between successive time-averaging calculations ($T_{sim} = s \cdot \Delta t_{TH}$). Similar relations hold for determining $\xi_{DC}(y)$. Taking successive time-averages results in an unavoidable and time-consuming iteration of calculated temperature profiles to an eventual converged distribution (usually only after at least 5 simulated RF cycles).

By implementing (2.64), and a similar equation of ξ_{DC}, and substitution of these into (2.61) gives the time-average of the power density distribution at a certain time. This averaged distribution is determined at regular intervals, Δt_{TH}, and is used to update the internal temperature profile. This ensures that the temperature distribution evolves consistently with the internal drift processes which are based on temperature-dependent material parameters. As pointed out

above, if enough time has elapsed, the time-averaged distribution will converge to the desired steady-state distribution $Q(y)$. Throughout the calculations below $Q(y)$, the *steady-state* distribution, is used with the understanding that it actually refers to each of the successive time-averaged distributions until convergence has indeed been reached.

In the simulation the continuous $\xi(y)$ and $J(y)$ distributions mentioned above are sampled on the fine grid defined earlier with the Poisson-solver. Consequently, $Q(y)$ is also sampled on this fine grid. The relation between the sampled $Q(y)$ and Q_i for the i^{th} layer is illustrated in Figure 2.11. The power dissipation density Q_i for each layer is simply calculated as the average of the sampled $Q(y)$ within the cell boundaries.

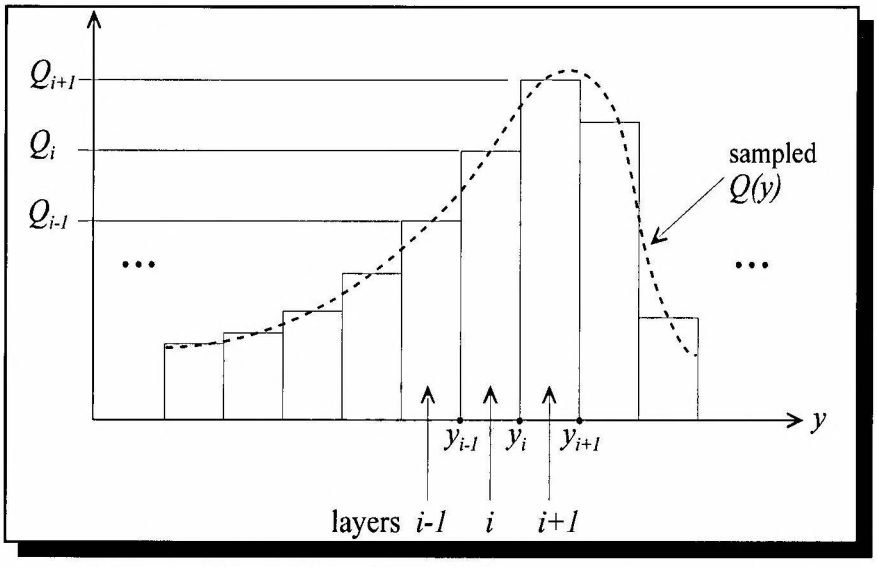


Figure 2.11: An illustration of the relation between the continuous $Q(y)$ distribution, which is sampled on the fine grid defined previously for the Poisson solver, and Q_i for the i^{th} layer.

2.2.6.3 Determination of the temperature profile $T_i(y)$ in the i^{th} layer as a function of its interface temperatures T_{i-1} and T_i

For any given layer i with constant thermal conductivity K_i and power dissipation density Q_i, the heat equation (2.61) reduces to

$$K_i \frac{d^2 T_i(y)}{dy^2} + Q_i = 0 \tag{2.65}$$

at a given moment with $T_i(y) = T(y)$ for $y_{i-1} < y \leq y_i$ and $T_i(y) = 0$ elsewhere. Integration of (2.65) with respect to y gives

$$K_i \frac{dT_i(y)}{dy} + Q_i y + a_i = 0 \tag{2.66}$$

and

$$K_i T_i(y) + \frac{1}{2} Q_i y^2 + a_i y + b_i = 0 \tag{2.67}$$

with a_i and b_i arbitrary integration constants.

From (2.67) it follows directly that the temperature profile $T_i(y)$ in each section is given by

$$T_i(y) = \frac{-\frac{1}{2} Q_i y^2 - a_i y - b_i}{K_i} \quad \text{for} \quad y_{i-1} < y \leq y_i. \tag{2.68}$$

The integration constants a_i and b_i have to be written in terms of the (assumed known) interface temperatures T_{i-1} and T_i.

By identifying $T_{i-1} = T(y_{i-1})$ and $T_i = T(y_i)$ it follows from (2.67) that

$$K_i T_i + \frac{1}{2} Q_i y_i^2 + a_i y_i + b_i = 0 \tag{2.69}$$

and

$$K_i T_{i-1} + \frac{1}{2} Q_i y_{i-1}^2 + a_i y_{i-1} + b_i = 0 \quad . \tag{2.70}$$

Solving for a_i and b_i from (2.69) and (2.70) and substitution of these expressions in (2.68) then yields the temperature profile across the i^{th} section in terms its interface temperatures:

$$T_i(y) = \frac{Q_i}{2K_i}\left(-y^2 + y(y_i + y_{i-1}) - y_i y_{i-1}\right) + T_i\left(\frac{y - y_{i-1}}{y_i - y_{i-1}}\right) - T_{i-1}\left(\frac{y - y_i}{y_i - y_{i-1}}\right) \quad (2.71)$$

2.2.6.4 Solving the interface temperatures

To solve the interface temperatures the continuity of heat flux [60]

$$K_i \left.\frac{\partial T_i(y)}{\partial y}\right|_{y=y_i} = K_{i+1} \left.\frac{\partial T_{i+1}(y)}{\partial y}\right|_{y=y_i} \quad (2.72)$$

is enforced at every interface. Substitution of (2.71) into (2.72) results in the system of (N_{TH}-1) linear equations

$$T_{i-1}\left(\frac{K_i}{y_i - y_{i-1}}\right) - T_i\left(\frac{K_i}{y_i - y_{i-1}} + \frac{K_{i+1}}{y_{i+1} - y_i}\right) + T_{i+1}\left(\frac{K_{i+1}}{y_{i+1} - y_i}\right) = \frac{1}{2}\left[Q_i(y_{i-1} - y_i) - Q_{i+1}(y_{i+1} - y_i)\right] \quad \text{for} \quad 1 \le i \le N_{TH} - 1 \quad (2.73)$$

which has to be solved for the unknown interface temperature T_0, T_1, ... $T_{N_{TH}}$. To generate a unique solution to (2.73) two boundary conditions, at $y = y_0$ and $y = y_{N_{TH}}$, will be applied.

- *Temperature at* $y = y_{N_{TH}}$

  The contact temperature adjacent to the integral heat sink is given by [32]

$$T_{HS} = T_{amb} + \left(\sum_{i=0}^{NY} Q_i \Delta y\right) \cdot \left(\frac{a}{K_{HS}} + \frac{L_{IHS}}{K_{IHS}}\right) \quad (2.74)$$

  with T_{amb} the ambient temperature, a the radius of the diode, L_{IHS} the length of the integral heat sink and K_{HS} and K_{IHS} the thermal conductivity of the heat stud and integral heat sink

respectively.

- *The top-cap boundary condition at* $y = y_0$

It is assumed that no heat is transferred from the top-cap contact to the cavity [32], i.e.

$$K_1 \frac{\partial T_1(y)}{\partial y}\bigg|_{y=y_0^+} = 0 \ . \tag{2.75}$$

Applying this condition (2.75) to (2.71) for $i = 1$ and $y_0 = 0$ gives

$$\begin{aligned} K_1 \frac{\partial T_1(y)}{\partial y}\bigg|_{y=y_0^+} &= \frac{Q_1}{2K_1}\left(-2y + y_1 + y_0 - y_1 y_0\right) + \frac{T_1 - T_0}{y_1 - y_0} \\ &= \frac{Q_1}{2K_1} y_1 + \frac{T_1 - T_0}{y_1} \\ &= 0 \end{aligned} \tag{2.76}$$

from which it follows directly that

$$T_1 - T_0 = -\frac{Q_1 y_1^2}{2K_1} \ . \tag{2.77}$$

This completes the mathematical model for the solution of the full set of interface temperatures T_i. The solution of the set of linear equations represented by (2.73) for the interface temperatures $T_0 \ldots T_{N_{TH}-1}$ by standard Gauss elimination [49] will now be discussed briefly.

Rewriting (2.73) in the form

$$T_{i+1} + a_i T_i + b_i T_{i-1} = c_i \tag{2.78}$$

for $i = N_{TH}-1, N_{TH}-2, \ldots 1$ and with

$$
\begin{aligned}
a_i &= -\left(1+\frac{K_i}{K_{i+1}}\cdot\frac{y_{i+1}-y_i}{y_i-y_{i-1}}\right) \\
b_i &= -a_i-1 \\
c_i &= \frac{y_{i+1}-y_i}{2K_{i+1}}\left[Q_i\left(y_{i-1}-y_i\right)-Q_{i+1}\left(y_{i+1}-y_i\right)\right]
\end{aligned}
\tag{2.79}
$$

and applying (2.74) and (2.77) gives, after appropriate matrix reduction, the tri-diagonal matrix equation

$$
\begin{bmatrix}
1 & w_{N_{TH}-1} & 0 & & & \cdots & & 0 \\
0 & 1 & w_{N_{TH}-2} & 0 & & & \iddots & \\
 & & & \ddots & & & & \vdots \\
 & & 0 & 1 & w_i & 0 & & \\
 & & & & & \ddots & & \\
\vdots & & & & 0 & 1 & w_2 & 0 \\
 & \iddots & & & & 0 & 1 & w_1 \\
0 & & \cdots & & & & 0 & 1
\end{bmatrix}
\begin{bmatrix}
T_{N_{TH}-1} \\ T_{N_{TH}-2} \\ \vdots \\ T_i \\ \vdots \\ T_2 \\ T_1 \\ T_0
\end{bmatrix}
=
\tag{2.80}
$$

$$
\begin{bmatrix} g_{N_{TH}-1} & g_{N_{TH}-2} & \cdots & g_i & \cdots & g_2 & g_1 & g_0 \end{bmatrix}^T
$$

with

$$
\begin{aligned}
w_{N_{TH}-1} &= \frac{b_{N_{TH}-1}}{a_{N_{TH}-1}} \\
w_i &= \frac{b_i}{a_i-w_{i+1}} \qquad \text{for} \quad i=N_{TH}-2,N_{TH}-3,\ldots 1 \\
g_{N_{TH}-1} &= \frac{c_{N_{TH}-1}-T_{N_{TH}}}{a_{N_{TH}-1}} \\
g_i &= \frac{c_i-g_{i+1}}{a_i-w_{i+1}} \qquad \text{for} \quad i=N_{TH}-2,N_{TH}-3,\ldots 1 \\
g_0 &= \frac{g_1+\dfrac{Q_1y_1^2}{2K_1}}{1+w_1} \quad .
\end{aligned}
\tag{2.81}
$$

The interface temperatures are now easily obtained by forward substitution:

$$\begin{aligned} T_0 &= g_0 \\ T_i &= g_i - w_i T_{i-1} \quad \text{for} \quad i = 1,2,\ldots N_{TH} - 1 \end{aligned} \tag{2.82}$$

and $T_{N_{TH}} = T_{IHS}$ from (2.74).

It is clear from (2.82) that the w_i parameters are dependent only on predefined geometric dimensions. They can therefore be calculated once at the beginning of the simulation and stored in a lookup table to save computing time. On the other hand, the g_i parameters are dependent on geometric dimensions and the power dissipation density distribution at a certain moment in time. As a result, these parameters have to be recalculated with each temperature calculation.

The temperature at any point in the device is readily determined by a linear interpolation between the two adjacent interface temperatures.

2.2.7 Device output characterisation

The device terminal voltage and current are required to determine the power delivered by the device to an external circuit. To mimic the effect of placing the diode in cavity a terminal voltage $v_D(t)$ of the form

$$v_D(t) = V_B + V_1 \sin(\omega_0 t) + V_2 \sin(2\omega_0 t + \phi) \text{ V} \tag{2.83}$$

is applied to the diode where V_B is the bias voltage, ω_0 the fundamental frequency, V_1 and V_2 the amplitude of the fundamental and second harmonics respectively, and ϕ the phase difference.

The total terminal current at any given point comprises the particle current as well as the displacement current [62]. The particle component of the terminal current can be determined by simply counting the number of electrons, or rather, super particles, that cross the terminal contact during each time step. The displacement current at the contact is dependent on the rate of change

of the electric field at the contact.

It has been found that the method of counting particles that traverse the contacts yields a very noisy particle current that can potentially corrupt power calculations. It was therefore decided to determine the particle current i_P at as follows:

$$i_P = \frac{q_s A}{k_2 - k_1} \sum_{m=k_1}^{k_2} n_m v_m \tag{2.84}$$

with: q_s the super particle charge,

A the cross-sectional area,

n_m the number of particles at mesh point m,

v_m the electron velocity at mesh point m and

k_1, k_2 the mesh points corresponding to the respective boundaries of the ohmic regions.

By averaging the particle current over the ohmic regions, the noise is greatly reduced. Averaging is acceptable since the particle current is, ideally, constant across these regions. The total terminal contact current is dominated by the particle current in the highly doped ohmic regions in the proximity of the contacts. The displacement current in these regions is negligible because the magnitude of the electric field is very small. The particle current can therefore be substituted for the total current for power calculation. The output power can easily be determined once the current response to the applied voltage is known. The applied voltage and terminal current are Fourier transformed, denoted by V_D and I_P respectively, from which the complex device admittance Y_D can be calculated.

Resistive losses within the device, for example in the contact regions, as well as circuit losses external to the diode also need to be taken into account to determine the output power realistically. This is done with the aid of the diode model in Figure 2.12. The diode can be modelled by a parallel combination (Y_D) of capacitance (jB_D) and negative resistance ($-G_D$). The internal loss mechanisms are represented by a series resistor R_{loss}. External losses can also be incorporated into R_{loss}. (Y_D therefore represents the admittance of the lossless diode.)

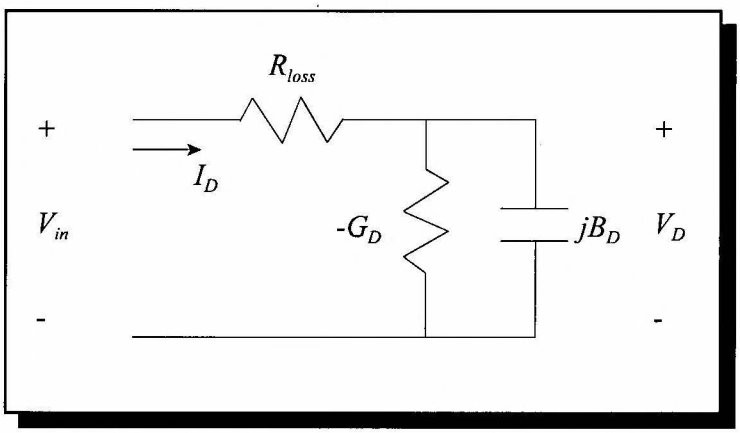


Figure 2.12: Circuit model of diode including resistive losses.

It can easily be shown that the input d.c. power P_{in} provided by the bias circuit is given by

$$P_{in} = V_B^2 \cdot Y_{DC}\left(1 + Y_{DC} \cdot R_{loss}\right) \tag{2.85}$$

where Y_{DC} denotes the d.c admittance of the diode, which has a real value.

The power generated by the diode, P_D, under lossless conditions at a certain harmonic k is given by

$$P_D = -\frac{1}{2}\left(V_D(k)\right)^2 real\left(Y_D(k)\right) \tag{2.86}$$

where the negative sign accounts for the negative conductance of the power generating diode.

Accounting for the resistive loss, the output power at the harmonic k is finally given by

$$P_{out} = P_D - \frac{1}{2}\left(V_D(k) \cdot Y_D(k)\right)^2 R_{loss}\,. \tag{2.87}$$

Resistive losses are expected to increase with frequency due to parasitic and skin effects. Values of $R_{loss} = 0.1\Omega$ at 47GHz and 0.2Ω at 94GHz have been assumed. These values yielded results that compare favourably with practical diodes.

2.2.8 Verification of Monte Carlo simulation model

The simulation model has been verified through studiously comparing experimental values with both bulk material simulations of GaAs and device simulations of a typical Gunn diode. Bulk simulations provide confirmation that the material parameter values that have been chosen, are indeed correct. Device simulations are used to verify that the model, which is now bounded by contacts, gives a true reflection of a real device. This entails solving field and temperature distributions throughout the device consistently with the movement of charge through the device.

2.2.8.1 Selection of material parameters

To ensure that the simulation model reflects real-world experiments, the values of the material parameters are iteratively adjusted until an optimum is reached. This is also referred to as "priming the model" [2]. Nominal values have been chosen from an extensive literature survey. A summary of implemented values of material parameters is given in Table 2.1. Also given below is the implementation of relevant parameters as functions of temperature and mole fraction x.

Throughout the following sections, the subscript "0" refers to values for GaAs (x=0) at 300K.

- *Electron effective mass*

The effective masses for electrons in, respectively, the Γ, L and X valleys have been implemented as follows:

$$\begin{aligned} m_\Gamma(\mathrm{x},T) &= m_{\Gamma 0} - 1.85\times 10^{-5}(T-300)m_0 + 0.083m_0\mathrm{x}\,, \\ m_L(\mathrm{x}) &= m_{L0} + 0.04m_0\mathrm{x}\,, \\ m_X(\mathrm{x}) &= m_{X0} - 0.07m_0\mathrm{x}\,. \end{aligned} \tag{2.88}$$

TABLE 2.1: Material parameters for GaAs at 300K

BULK MATERIAL PARAMETERS			
Parameter	Symbol	Values from literature [a]	Value Implemented
Sound velocity (m/s)	v_s	5220 - 5240	5240
Density (kg/m^3)	ρ	5360 - 5370	5360
Low-frequency dielectric constant	ϵ_s	12.4 - 13.18	12.53
High-frequency dielectric constant	ϵ_∞	10.6 - 10.89	10.82
Breakdown field (MVm^{-1})	E_b	40	n/a
Thermal conductivity (Wm^{-1}K^{-1})	C	35 - 55	55
VALLEY-DEPENDENT PARAMETERS			
Parameter	Symbol	Values from literature [a]	Value Implemented
Effective mass (kg)	m_Γ	0.063 - 0.069	$0.067m_0$
	m_L	0.222 - 0.292	$0.290m_0$
	m_X	0.409 - 0.471	$0.450m_0$
Band non-parabolicity (eV^{-1})	α_Γ	0.576 - 1.160	0.67
	α_L	0.204 - 0.650	0.4
	α_X	0.360 - 0.550	0.55
Valley separation (eV)	Δ_L	0.284 - 0.330	0.284
	Δ_X	0.422 - 0.486	0.447
Number of equivalent valleys	Z_Γ	1	1
	Z_L	4	4
	Z_X	3	3
Polar optic phonon frequency (r/s)	ω_{op}	(5.21 - 5.51) x10^{13}	5.37x10^{13}
Intervalley phonon frequency (r/s)	$\omega_{\Gamma L}$	(4.22 - 4.56) x10^{13}	4.60x10^{13}
	$\omega_{\Gamma X}$	4.54x10^{13}	4.60x10^{13}
	ω_{LX}	(4.45 - 4.54) x10^{13}	4.60x10^{13}
	ω_{LL}	(4.41 - 4.54) x10^{13}	4.41x10^{13}
	ω_{XX}	4.54x10^{13}	4.60x10^{13}
Intervalley deformation potential	$\Xi_{\Gamma L}$	(0.15 - 1.1) x10^{11}	1.0x10^{11}
(eV/m)	$\Xi_{\Gamma X}$	(0.5 - 1.1) x10^{11}	1.1x10^{11}
	Ξ_{LX}	(0.34 - 1.1) x10^{11}	1.1x10^{11}
	Ξ_{LL}	(1.0 - 1.1) x10^{11}	1.0x10^{11}
	Ξ_{XX}	(0.27 - 1.1) x10^{11}	1.1x10^{11}
Acoustic deformation potential (eV)	Ξ_{ac}	7	7

[a] From [63] - [67].

The temperature dependence is a linear interpolation between 300K and 500K of data from [63]. From the same reference it can be assumed that the temperature dependence of m_L and m_X is negligible. The compositional (x) dependence is a linear interpolation between x = 0 (GaAs) and x = 1 (AlGaAs) of data presented by Adachi [64].

The linear interpolation implemented here, and in the following sections, limits the computational load associated with these calculations.

- *Band non-parabolicity*

Directly from [58], the non-parabolicity of the X-valley can be assumed constant as a function of x, whereas those of the Γ- and L-valleys are given by

$$\begin{aligned} \alpha_\Gamma(\mathrm{x}) &= \alpha_{\Gamma_0} - \mathbf{0.94x}\ , \\ \alpha_L(\mathrm{x}) &= \alpha_{L_0} - \mathbf{0.038x} \end{aligned} \tag{2.89}$$

respectively.

- *Valley separation*

The valley separations between the conduction band minima of the Γ-valley and those of the L-valleys and X-valleys are given by

$$\begin{aligned} \Delta_L(\mathrm{x},T) &= \Delta_{L_0} - 5.7\cdot 10^{-5}(T-300) - 0.605\mathrm{x}\ , \\ \Delta_X(\mathrm{x},T) &= \Delta_{X_0} + 7.1\cdot 10^{-5}(T-300) - 1.122\mathrm{x} + 0.143\mathrm{x}^2 \end{aligned} \tag{2.90}$$

respectively.

The temperature is a linear interpolation between 300K and 500K of the non-linear relationship given by [63]. The x-dependence is directly implemented from [64].

- *Material density*

The compositional dependence of the material density is given by [64] as

$$\rho(\mathrm{x}) = \rho_0 - 1600\mathrm{x}\,. \tag{2.91}$$

- *Dielectric constant*

The static and high-frequency dielectric constants are given by

$$\begin{aligned} \varepsilon_s(\mathrm{x},T) &= \varepsilon_{s0}\left(1 + 1.2\cdot 10^{-4}\,T\right) - 3.12\mathrm{x}\,, \\ \varepsilon_\infty(\mathrm{x},T) &= \varepsilon_{\infty 0}\left(1 + 9\cdot 10^{-5}\,T\right) - 2.73\mathrm{x} \end{aligned} \tag{2.92}$$

respectively, where the temperature dependence is obtained from [63] and the compositional dependence from [64].

2.2.8.2 Bulk material simulations

Bulk material simulations can be categorised into low- (<0.1MVm^{-1}), medium- (0.1 - 1MVm^{-1}) and high-field (1 - 10MVm^{-1}). Comparison with experimental measurements has been sought as far as possible. The respective simulation results are given in the sections below.

- *Low-field simulations*

These simulations essentially give the low-field mobility of GaAs. Figure 2.13 summarises the dependence of mobility on temperature and impurity doping concentration.

- *Medium-field simulations*

The simulated steady state electron velocity as a function of electric field for intrinsic GaAs at room temperature is shown in Figure 2.14. Also shown are experimental values obtained from various sources.

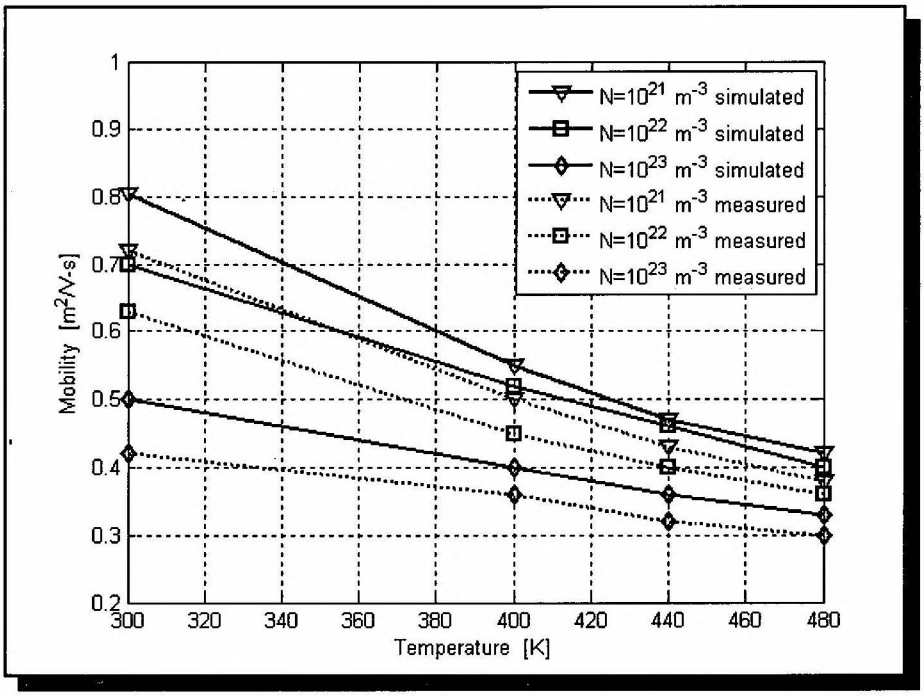


Figure 2.13: Low-field mobility of GaAs as a function of temperature for various doping levels. The solid lines indicate simulation results. Experimental results (from [68]) are presented by the dashed lines.

Noteworthy is that a threshold velocity of 1.8×10^5 ms^{-1} is reached at about $0.35 MVm^{-1}$ after which the velocity decreases - this constitutes the negative differential resistance that is utilised for Gunn operation. The decrease in velocity is attributed to the transfer of electrons from the central valley, where they have a small mass, to the L-valleys which is characterised by a relatively heavier mass.

Of importance to Gunn diodes, which typically operate at elevated temperatures, is the dynamic behaviour of electrons at these temperatures. The simulated drift velocity curve generated at various temperatures is shown in Figure 2.15. As is expected, an increase in temperature translates to lower drift velocities due to increased electron scattering. However, the negative differential region is still evident, although with a reduced slope. This indicates a lower negative differential resistance, which is associated with lower output power.

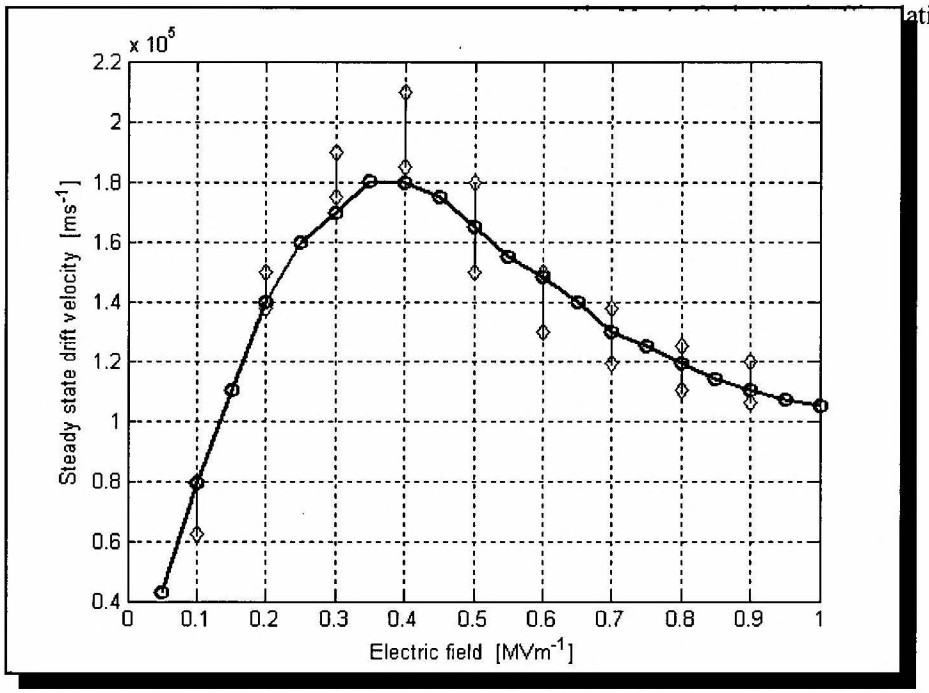


Figure 2.14: Simulated steady state drift velocity as a function of electric field for intrinsic GaAs at 300K. The simulated curve is presented by the solid line. The diamonds indicate the scope of experimental values as obtained in the literature [69], [70], [71].

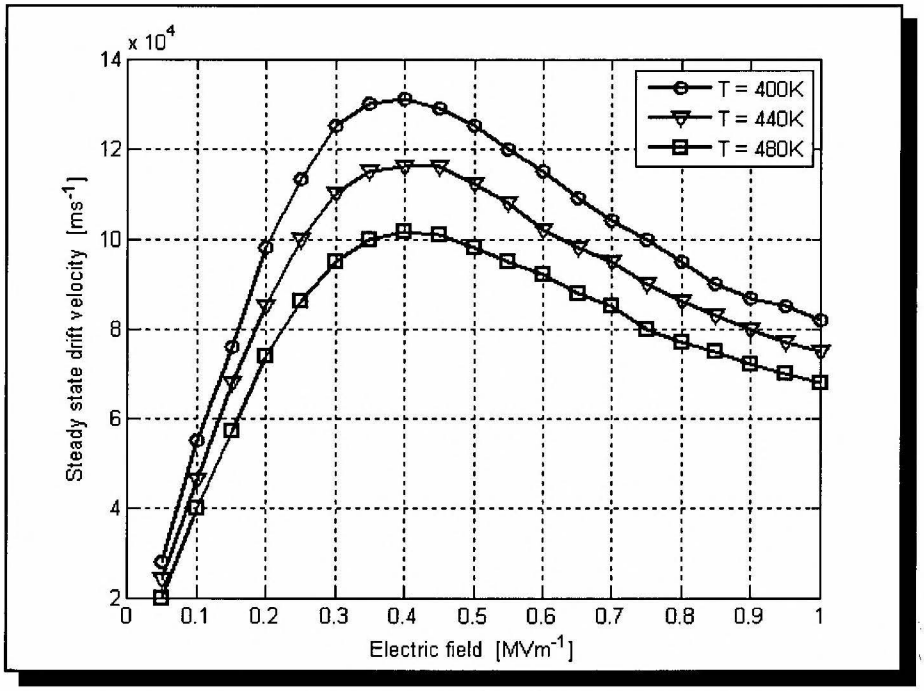


Figure 2.15: Simulated steady state drift velocity as a function of electric field for intrinsic GaAs at 400K, 440K and 480K.

The dependence of the drift velocity on doping concentration is shown in Figure 2.16. As seen clearly from the graph, impurity scattering influences the drift velocity in the lower field region only.

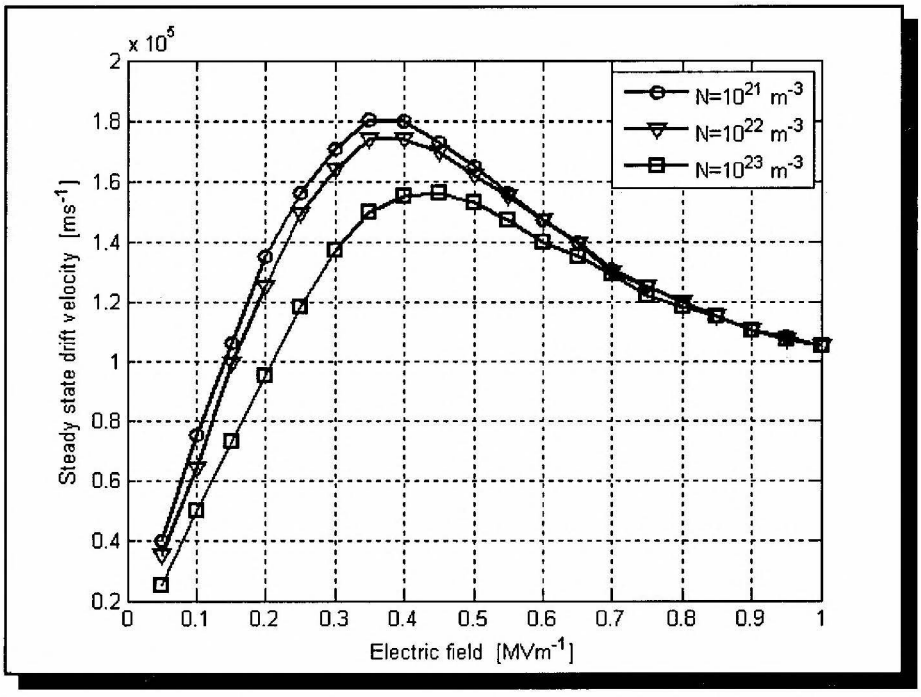


Figure 2.16: Simulated steady state drift velocity as a function of electric field for GaAs with various doping levels N.

- *High-field simulations*

Figure 2.17 illustrates the drift velocity versus electric field curve for intrinsic GaAs at room temperature. Also shown are experimental values obtained from various sources (see figure heading for references). It is clear that the velocity curve levels out at higher fields, and will eventually increase at a low rate due to most electrons occupying the X-valleys.

As an extension of Figure 2.15, the behaviour of the drift velocity at elevated temperatures in the high-field region is shown in Figure 2.18. Both measured and experimental values are shown.

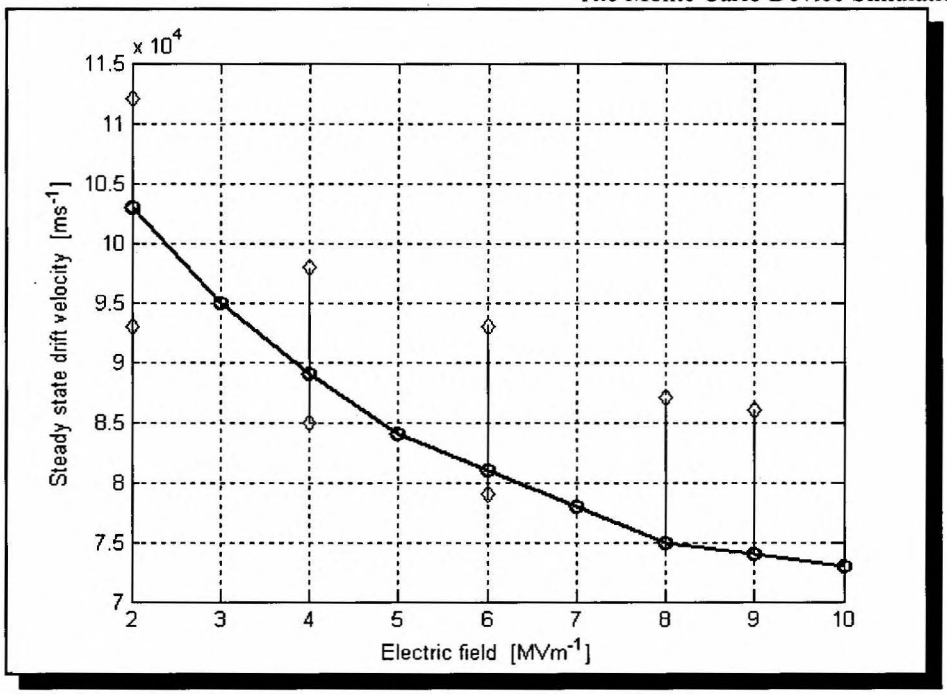


Figure 2.17: Simulated steady state drift velocity as a function of electric field for intrinsic GaAs at 300K. The simulated curve is presented by the solid line. The diamonds indicate the range of experimental values as obtained in the literature (from [70]).

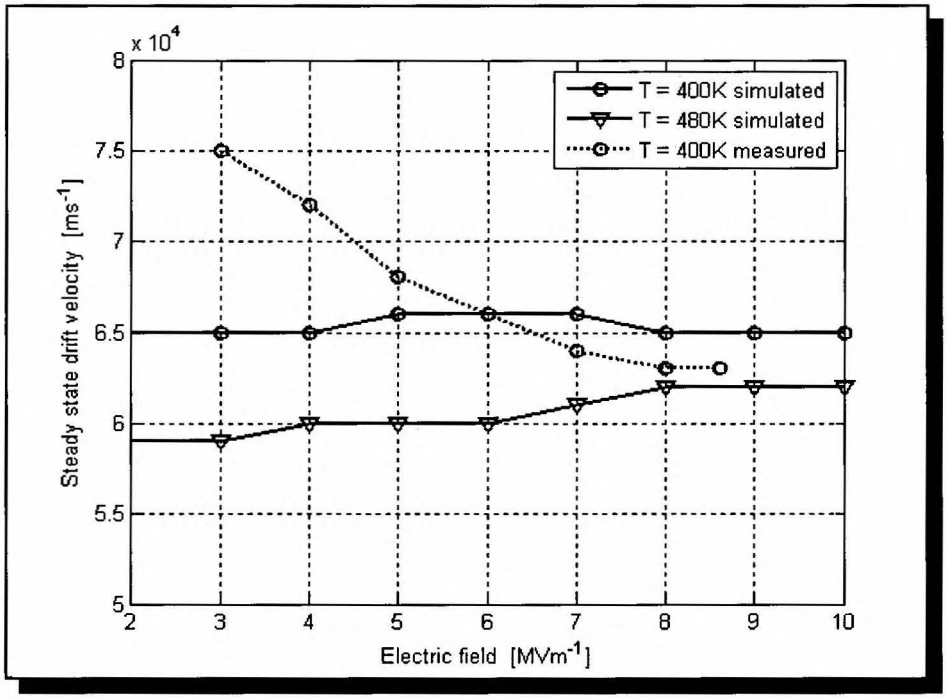


Figure 2.18: Simulated steady state drift velocity as a function of electric field for intrinsic GaAs at elevated temperatures. The simulated curve is presented by the solid line. Experimental results from [70] are indicated by the dashed line.

2.2.8.3 Device simulations

A 65GHz Gunn diode, experimentally characterised by Batchelor [33] in1992, has been simulated. As far as the author could ascertain, this is the only reported mm-wave diode to be simulated and measured experimentally in such detail, and therefore serves as an adequate reference for verification purposes. Both d.c. and a.c. analyses have been carried out and compared with the experimental values. The diode's active region has a length of 2.6μm, nominal doping of $0.7 \times 10^{22} m^{-3}$ and a diameter of 90μm. From Figure 4.3 it is evident the diode operates in the second harmonic mode because of its length at this frequency.

- *d.c. current-voltage simulation*

The simulated d.c. current-voltage curve is shown in Figure 2.19. Also shown is the variation of experimental results obtained from a batch of diodes. The onset of Gunn oscillations (in the negative differential resistance region) appear at 1.5V. Below this value the diode functions as a normal resistor.

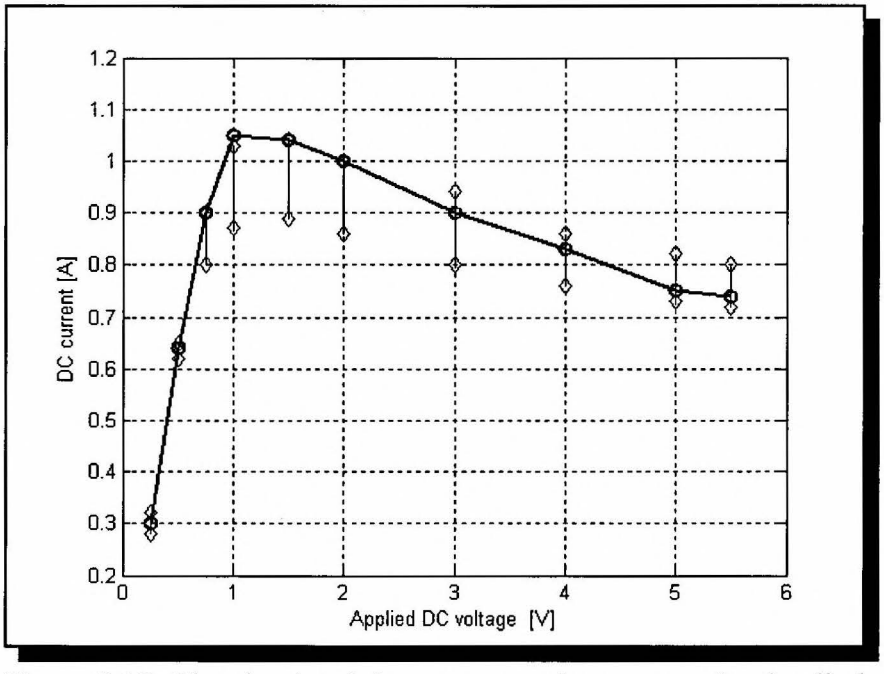


Figure 2.19: The simulated d.c. current-voltage curve for the diode characterised by [32]. Also shown is the range of experimental results that has been obtained from a single batch of diodes (indicated by the diamonds).

- *a.c. simulation*

An a.c. simulation has been performed to verify that the simulation model predicts the output RF power characteristics of the diode accurately. For this simulation the contact voltage across the diode $v_D(t)$ is assumed to be of the form given by (2.83). Both harmonics have to be presented to the diode due to the second harmonic operation of the diode.

The variables in (2.83) are adjusted until the output power is optimised for a given bias voltage. For V_{DC} = 6V, the following variables values have been chosen: V_1 = 4V, V_2 = 1.4V and ϕ = 320°. The output power is determined as described in Section 2.2.7.

The following performance quantities at 6V bias have been simulated (the experimental values for a given batch of diodes are given in brackets):

Output power:	40mW	(20 - 39mW)
Bias current:	0.58A	(0.6A)
Efficiency (P_{output}/P_{input})	1.15%	(1%)

To illustrate the temperature range at which Gunn diodes typically operate, the following temperature values are noted:

Temperature at heat sink	422K
Temperature at cathode	474K
Temperature drop over active region	10K

2.2.8.4 Discussion of results

- *Bulk material simulations*

The low-field simulations (see Figure 2.13) show that the model predicts the mobility of GaAs to within 10% of that obtained through experiment across a wide range of doping concentrations and temperatures. This is deemed satisfactory.

In the medium-field region, which is of great importance to the onset of Gunn-operation, the simulated results indicate excellent correlation with experiment (see Figure 2.14). Also shown in Figures 2.15 and 2.16 is the dependence of electron steady-state dynamic behaviour on temperature and doping levels. The model reliably predicts the expected trends.

It will be apparent from Chapter 5 that large sections of Gunn-diode active regions are subjected to high electric fields at elevated temperatures. It must therefore be expected from the model to accurately reflect experiment under these conditions. This is indeed the case. Figure 2.17 shows that, at room temperature, the simulated curve falls completely within experimental values. The simulation results at higher temperatures also show excellent correlation with experiment to within 15% at 3MVm^{-1} and to within 5% at higher electric fields.

- *Device simulations*

From the results obtained, it is clear that the simulated values for both d.c (see Figure 2.19) and a.c simulations correlate excellently with measured values.

In conclusion it can therefore be assumed that both the chosen material parameter values and the implementation of their temperature and compositional dependence are optimised for the simulation model. Furthermore, the model can be accepted to accurately predict device performance.

Chapter 3

MC-PVM: A PARALLEL MONTE CARLO SIMULATOR

"Parallel processing, the method of having many small tasks solve one large problem, has emerged as a key enabling technology in modern computing. The past several years have witnessed an ever-increasing acceptance and adoption of parallel processing, both for high-performance scientific computing and for more 'general-purpose' applications, as a result of the demand for higher performance, lower cost, and sustained productivity."

- A. Geist, 1994

3.1 INTRODUCTION

Any meaningful device research based on Monte Carlo (MC) simulations requires fast implementations of these algorithms. By its very nature the MC particle simulation technique is extremely time consuming. Historically the MC technique has been shunned by researchers at smaller institutions, notably those in developing countries. This is partly due to the complexity of MC algorithms and, more importantly, the restricted access these researchers have to expensive powerful supercomputers.

Parallel computing affords us the ability to vastly speed up the execution of computationally intensive algorithms by solving several sub tasks, assigned to different processors, simultaneously. The parallel approach has been applied to the MC simulation by Goodnick *et al* [72] on a distributed memory nCUBE multicomputer containing a vast array of independent processors. However, financial constraints place massive multiprocessor computers out of the reach of smaller universities and research centres.

An efficient and cost-effective solution to this problem is proposed here whereby an array of personal computers (pc's) is employed into a parallel virtual machine (PVM) on a computer network in a master-slave model. This proves to be an extremely viable option because most universities support large numbers of pc's connected to a network. The proposed algorithm, MC-PVM [9], [10], also facilitates relative ease of converting existing single processor Monte Carlo simulation code into parallel implemented simulation packages.

In this chapter an introduction to parallel computing in general will first be given. To appreciate the implications of running parallel applications across a computer network - i.e. *distributed computing* - this will be followed by some introductory notes on relevant issues pertaining to computer networking. A discussion on the PVM software package as required by the MC-PVM simulator will follow. MC-PVM, together with a more efficient leap-frog derivative of the algorithm, will then be presented. The accuracy and efficiency of the proposed parallel algorithms have been investigated by applying them to a simulation of a millimeter-wave Gunn-effect relaxation oscillator.

3.2 PARALLEL PROCESSING

3.2.1 Background

The rationale behind parallel computing is obvious. It presents us with vastly increased computing power by combining the resources of individual processors. When the boundaries of

existing computing technology have been reached, parallel processing is the next logical step in our efforts to fulfill our insatiable need for faster computers.

Although parallel processing had its origin in the 1950s with early computers processing the bits of a word in parallel, the most notable advance in computing power came in 1976 with the introduction of the CRAY-1. Special detail to parallelism made it the fastest computer of its time. The next advance in parallel processing came in the early 1980s with the advent of massively parallel processor (MPP) machines. These machines consist of large arrays of independent processors, ranging from a few hundred to a few thousand, contained in a single cabinet. The individual processors are typically linked via high speed busses to vast amounts of memory. MPP machines are presently the fastest computers in the world and are extensively used to solve "Grand Challenge" computational problems, such as global meteorological modelling. Of course this phenomenal computing power comes at a price, and a very steep one at that. The financial burden these machines pose to smaller research facilities effectively place them out of their reach.

The 1980s also saw the advent of *distributed computing* whereby a set of individual computers is linked via a network to collectively solve a single large problem. It proves to be an extremely viable option because most universities support large numbers of pc's connected to a network. Distributed computing has brought the power of parallel processing within reach of those who cannot afford the more expensive MPP machines. It should be noted that several MPP's can of course be linked together in a distributed computing topology to realise phenomenal computational power.

A common factor in any parallel process scheme is the passing of information among the individual processors, whether they be the single processors of MPP's or individual personal computers (pc's). A need for the effective orchestration of passing these messages among groups of cooperating processors has culminated in the development of various dedicated software packages. One such package is the PVM system [73] which will be discussed in more detail in the following paragraph. The PVM system is by no means the only such package available. Various alternatives to PVM exist, a few of which will be mentioned in Section 3.2.3.

3.2.2 Budget supercomputing on a cluster

A budget supercomputer (MC^2 - acronym for *Monte Carlo Cluster*) [11] has been implemented for the purposes of this research work.

A cluster is a group of commercial personal computers that, together, create a high-performance

computing tool. There are essentially three components required to create such a clustered supercomputer, namely the hardware, interconnect or networking technology and software. Each of these will be discussed in the following sections.

- Hardware

A typical high performance Beowulf Cluster [74] comprises a network of individual computers, or nodes, that are usually configured in a master-slave arrangement. MC^2 is the culmination of successive upgrading of previous clusters. It consists of 1 master node (2.8GHz Pentium 4) and 19 diskless slave nodes (2.4GHz Pentium 3). The master node is commonly more powerful in terms of its processor and memory. In its current implementation, it is the only node that contains a hard drive (in this case a SCSI HDD), two network interface cards and peripheral access.

- Interconnection

The cluster is linked via a dedicated ethernet network, separable from the institution's networks. The latest implementation incorporates a gigabit ethernet switch which facilitates networking speeds of up to 1000 megabit per second. This greatly reduces inter-node communication times, compared to the conventional 100 megabit per second ethernet.

- Software

As with many computer-based systems, the software plays an important role in co-ordinating the cluster's behaviour. The host operating system chosen for the cluster is Linux and more specifically the RedHat Linux, Version 9 [75]. To aid in the setup and booting of the slave nodes, the use of LTSP (Linux Terminal Services Project) [76] is employed. LTSP is a collection of software applications to allow the rapid deployment of diskless clients, whether graphical- or terminal-based. LTSP is fundamental to the operation of the cluster in that it provides the slave nodes with networked booting service, a shared root file system, access to a user's home directory on the master node and RSH (remote shell execution) access required by PVM.

As a requirement for the cluster, support of the Parallel Virtual Machine (PVM) was added. Modifications to the LTSP distribution were required to enable functionality of PVM, for example required libraries were added to support PVM and PVM applications, making this cluster's implementation unique. PVM is discussed in Section 3.2.4.

3.2.3 Network communication

Of profound importance when implementing networked parallel applications is the efficiency of network communication. The combined computing resources of several individual machines can seriously be hampered by ineffective data communication. It is therefore necessary to characterise network communication to investigate its influence on the overall computational efficiency. This is a complex and highly probabilistic problem.

As already stated, the computers are linked together via a single cable for data transmission. It is therefore possible at some stage that two or more computers will attempt to access the network simultaneously. This will of course corrupt the transmitted data. With CSMA/CD (carrier sense multiple access / collision detection), which is employed by standard ethernet, each computer first determines whether a data package from another computer is actively being transmitted before attempting to access the network. If a carrier signal is sensed (CS) the computer will delay its transmission before attempting another broadcast.

It may still happen that more than one computer, intending to transmit, find the network inactive at a certain point in time. This will result in simultaneous data transmissions and consequent corruption of the data. This event is referred to as collision. A collision is detected by a transmitting computer by correlating the data it has just sent with the data signal on the network just after broadcast. If the two signals differ a collision is assumed to have taken place. To ensure that the other computers involved are aware of the collision, the computer proceeds to broadcast for a short period of time a random data sequence, the so-called "jam" sequence. After another short time interval, the length of which is randomly generated by each computer involved in the collision, the computers attempt a re-broadcast.

In conclusion, data transmission across a CSMA/CD network is highly probabilistic and depends on the instantaneous network load at certain times. The random nature of CSMA/CD precludes a deterministic quantification of the transmission time of data packages and makes it impossible to predict exactly the effective data transmission rates on a certain network. The golden rule with distributed processing is to ensure that the algorithms do not require intensive message passing among machines - referred to as *fine grained* simulations. Optimal results can only be expected with *coarse grained* applications where network communication is limited. This has been borne in mind with the development of MC-PVM, as is evident from the computational efficiency realised by it.

The aim of this paragraph is merely to sensitise the reader to the unpredictable nature of network communication. The effect it has on the overall efficiency of the parallel application and the

constraints it places on the scope of the simulations that can be executed on a certain network will be discussed in paragraph 3.5.

3.2.4 The Parallel Virtual Machine ("PVM")

The PVM system [73] was originally developed at the Oak Ridge National Laboratory. Under the PVM parallel processing model, heterogeneous computer resources are made available as one large distributed memory computer. The overall computational problem is divided into a number of co-functional, possibly identical, processes. Each node of the virtual machine executes one or more of these processes. The cooperating processes communicate via a message passing protocol defined by the PVM software. A message passing server is installed on each node and handles all communication among the nodes of the virtual machine. The client code makes use of PVM Application Programme Interface calls in order to send and receive messages from other processes running on the virtual machine.

The PVM software environment is not the only parallel processing library available. An alternative is the MPI (Message Passing Interface) standard [77]. The MPI standard differs from the PVM software in that it is primarily aimed at implementation on homogenous computer resources. It does not implement the additional code required to integrate machines of different byte-ordering. This results in more efficient usage of the computer resources since there are less communication overheads. Another parallel processing avenue that can be explored is the use of reconfigurable custom hardware as a high performance processing resource, for example FPGA's.

The PVM-routines used in the parallel Monte Carlo simulation are grouped together in the *m_pvm.h* and *s_pvm.h* header files in the programme listing in Appendix A.

3.3 THE PARALLEL MONTE CARLO ALGORITHM

As stated before, the versatility and accuracy of MC simulations are extremely costly in terms of computational requirements. To ensure numerically sound simulation results, large ensembles of particles need to be simulated, typically containing in the order of 50 000 particles. The motion of each particle has to be simulated in turn, for the full duration of the simulation. It is assumed that these particles are effectively independent which makes the MC simulation well suited to parallel implementations to reduce computation time.

As mentioned earlier, Goodnick *et al* [72] have proposed an efficient parallel algorithm for

implementation on a multicomputer, namely the PMC-3D program. The algorithm was implemented on a 1024-node distributed memory nCUBE multicomputer. The algorithm is based on the spatial division of the device into subgrids with each of the processors assigned to a particular subgrid. Each processor simulates the motion of particles in its subgrid and is therefore responsible for the simulation of particles in a certain area of the device. The spatial division also makes it possible to solve Maxwell's equations for the internal fields in parallel.

MC-PVM is an efficient and cost-effective alternative to PMC-3D. As will be evident shortly, it is implemented in a master-slave(s) configuration. The fundamental difference between the MC-PVM and PMC-3D is the division of the ensemble of particles into subensembles, rather than dividing the *device* spatially into subgrids. The subensembles are therefore not associated with particular regions in the device but rather with different slave processors. Each of these processors is responsible for the subensemble of particles dedicated to it by the master processor. The PMC-3D algorithm necessitates the passing of particles among processors due to the movement of the particles through the device. This, together with the parallel solution of Maxwell's equation, results in intricate communication among processors. The proposed PVM implementation, on the other hand, requires only limited communication between the master and slaves and no communication among the slaves. This coarse grained implementation is therefore very suitable for implementation on a computer network.

The MC-PVM algortihm will now be discussed together with a novel leap-frog modification to the algorithm whereby the computational efficiency is markedly improved.

3.3.1 The MC-PVM Algorithm

The MC-PVM algorithm is based on a master-slave model. The ensemble of particles is divided into subensembles, each of which is dedicated to a separate processor (slave). The slaves are solely responsible for simulating the particles' dynamics. The master processor updates the field distribution consistently with the port conditions enforced by the external circuitry and the spatial evolution of the charge particles by solving Poisson's equation after every T_{step}- interval. The master also serves as user interface. The MC-PVM algorithm will now be discussed with the aid of the flow chart shown in Figure 3.1.

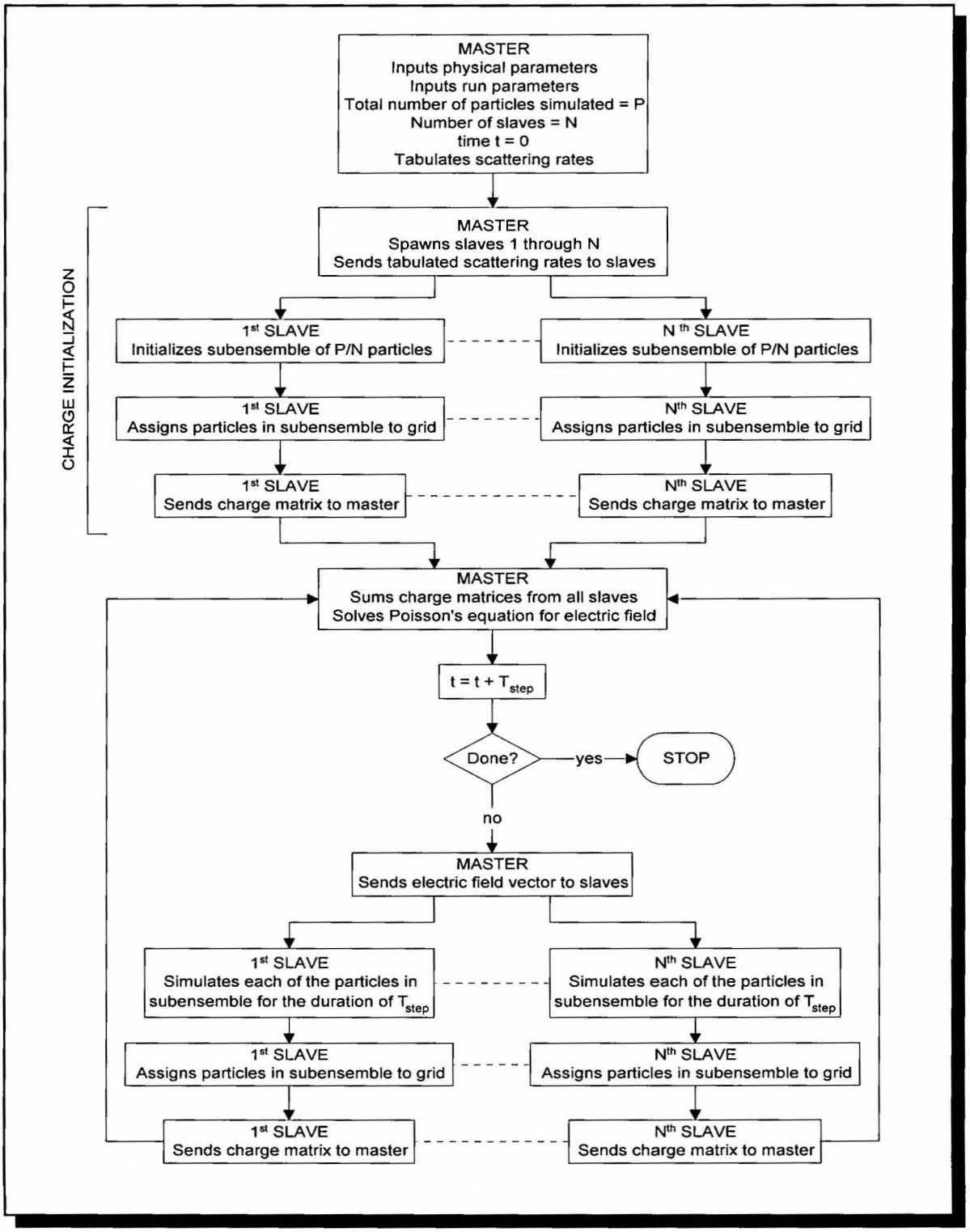


Figure 3.1: A flow chart of the MC-PVM algorithm

Initialisation

- The master inputs the material and run parameters needed for the simulation. It also tabulates the various scattering rates as a function of particle energy to save computing time. The total number of particles simulated is P and the number of slave processors N.
- The master then spawns the slave executable code on N different slaves.
- Each slave initializes a subensemble of P/N particles in real and ***k***-space. The *combination* of all the subensembles adheres to initial charge neutrality throughout the device.
- Each slave assigns the particles in its subensemble to grid points according to the particles' positions. It sends this charge distribution matrix to the master.
- Any other required spatial distribution pertaining to the state of the subensemble of electrons of each slave is also sent to the master.

Updating of the electric field and thermal distributions

- The master receives the charge matrix, together with any other matrices as described above, from each slave at the end of each T_{step}-interval. It sums these matrices to obtain the overall distributions of the quantities involved. The master proceeds to calculate the updated electric field matrix and temperature profile, which are sent to each slave for the next T_{step}-interval.

Particle simulation and charge assignment

- Each slave simulates the evolution of the particles in its subensemble in real and ***k***-space for the duration of the T_{step}-interval.
- After each T_{step}-interval the slaves assign the particles in the subensembles to grid points and send these charge matrices to the master.
- The slaves now wait for the updated electric field matrix from the master to continue with the simulation of the particles during the following T_{step}-interval.

It is very clear that the slaves are idle while the master solves Poisson's equation. This impacts negatively on the overall computational efficiency of the MC-PVM algorithm. This has led to the development of a more efficient leap-frog (LF) algorithm whereby the slaves' idling time is eliminated.

3.3.2 The Leap-Frog MC-PVM algorithm

The LF algorithm will be discussed with the aid of the timing diagram in Figure 3.2. Charge assignment by the slaves and the updating of the field distribution by the master still take place at regular T_{step}-intervals. However, each field update *lags* its associated charge assignment by $T_{step}/2$. Charge assignments and field updates correspond to the *m*-and *n*-points respectively. After the slaves have assigned the charge in their respective subensembles to gridpoints and have sent these charge matrices $[\rho]_m$ to the master, they do not wait for the master to solve Poisson's equation for the updated field vector matrix $[\vec{\xi}]_n$, as is the case with the MC-PVM algorithm. Instead, they immediately continue to simulate their respective subensembles for the remainder of the current T_{step}-interval *while* the master updates the field distribution. The master sends the updated field distribution $[\vec{\xi}]_n$ to the slaves at an instant $T_{step}/2$ after it has received the charge matrices $[\rho]_m$. The slaves use $[\vec{\xi}]_n$ for the next T_{step}-interval. This consecutive assignment of charge and the updating of the field distribution at intermediate points in time lead to a leap-frog arrangement of these two processes. The LF algorithm results in less time wasted because the master solves Poisson's equation concurrently with the charge simulations by the slaves.

At the instant $[\vec{\xi}]_n$ is sent to the slaves, the associated charge distribution $[\rho]_m$ of each slave has changed because the slaves have continued to simulate the particles for a period of $T_{step}/2$. It has been found that the use of $[\vec{\xi}]_n$, which already lags $[\rho]_m$ by $T_{step}/2$, for the full duration of the following T_{step}-interval leads to unsatisfactory simulation results. A successful solution to this problem is the appropriate prediction in time of the electric field by each slave to minimise the

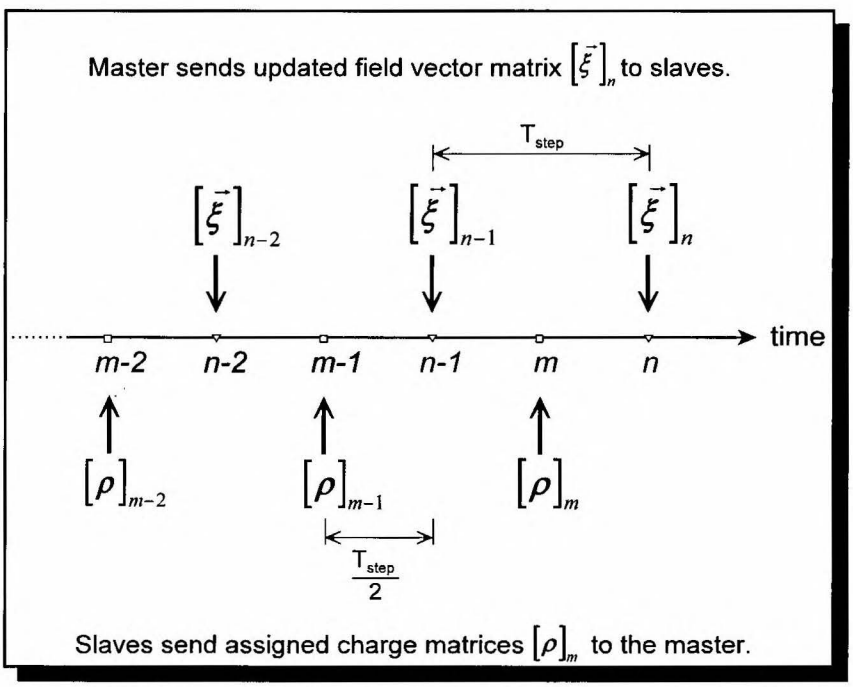


Figure 3.2: Time scale diagram for the leap-frog parallel algorithm.

effect of the time lag between $[\vec{\xi}]_n$ and $[\rho]_m$. The slaves receive an updated field distribution, for example $[\vec{\xi}]_{n-1}$ which is associated with $[\rho]_{m-1}$. The slaves use $[\vec{\xi}]_{n-1}$ for the next $T_{step}/2$ interval after which they send their respective $[\rho]_m$ matrices to the master. Instead of proceeding with the particle simulation using $[\vec{\xi}]_{n-1}$, the slaves predict a field distribution $[\vec{\xi}^*]_m$ based on the current $[\vec{\xi}]_{n-1}$ distribution and the previous predicted field distribution $[\vec{\xi}^*]_{m-1}$. The predicted field distribution $[\vec{\xi}^*]_m$ will be valid until the master sends the new updated field distribution $[\vec{\xi}]_n$ to the slaves, a period of $T_{step}/2$ later. It has been found that a simple linear prediction given by yields satisfactory results.

$$\left[\vec{\xi}^*\right]_m = \left[\vec{\xi}\right]_{n-1} + \left[\left[\vec{\xi}\right]_{n-1} - \left[\vec{\xi}^*\right]_{m-1}\right] \tag{3.1}$$

3.3.3 Computational Efficiency

3.3.3.1 A typical device simulation

The all-important issue of computational efficiency of the MC-PVM and LF algorithms will now be investigated by comparing the results for a one-dimensional simulation of a millimeter-wave Gunn-effect relaxation oscillator to those obtained by Tully [78]. The results incidentally also confirm the accuracy of these algorithms.

The doping profile of the Gunn-diode is shown in Figure 3.3. To calculate the terminal currents, the device area is assumed to be 5.10^{-5} cm^2 in accordance with physical devices. An ensemble of 70 000 particles, divided equally among the slave processors, has been employed. The

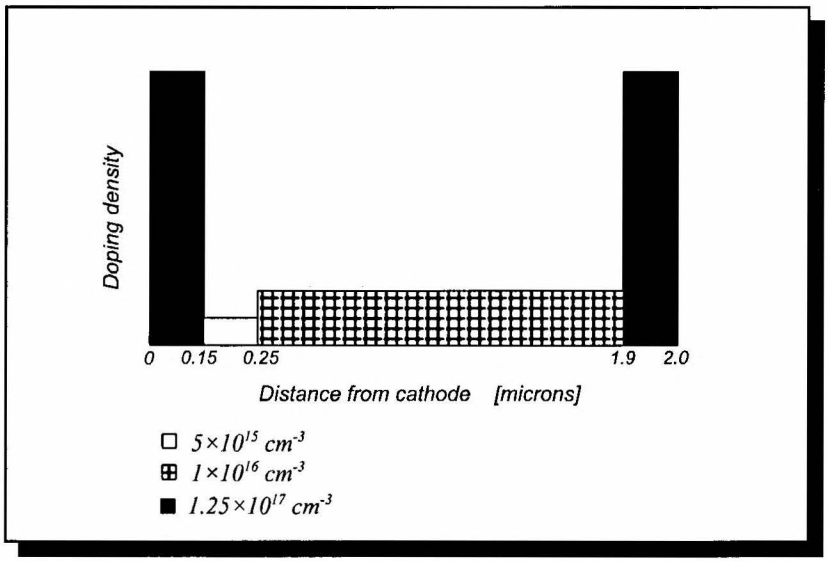


Figure 3.3: The doping profile of the simulated Gunn-diode.

oscillator circuit is modelled as a parallel resonant circuit as shown in Figure 3.4. A 5 fs field adjusting time step has been used. The device is divided spatially into 1000 segments for the calculation of the field distribution.

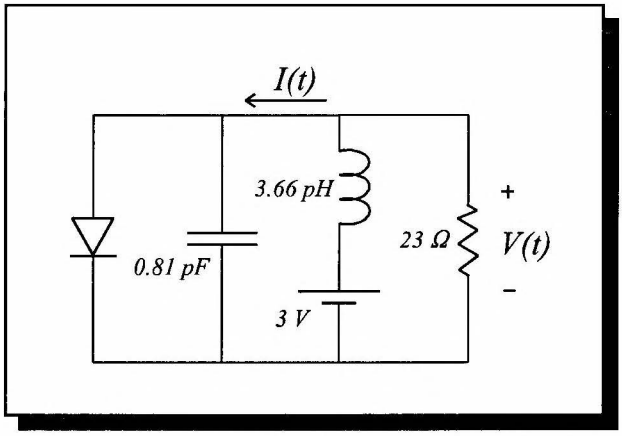


Figure 3.4: The circuit schematic of the simulated relaxation oscillator.

The simulated voltage and current waveforms obtained by the LF algorithm are given in Figure 3.5. The results are also identical to those predicted by the MC-PVM algorithm. The simulation shows excellent correlation with the results obtained by Tully. It should be noted here that the LF algorithm has been used in the verification of the algorithm as discussed in Chapter 2, where it has been found to imitate real-life bulk and device experiments very well.

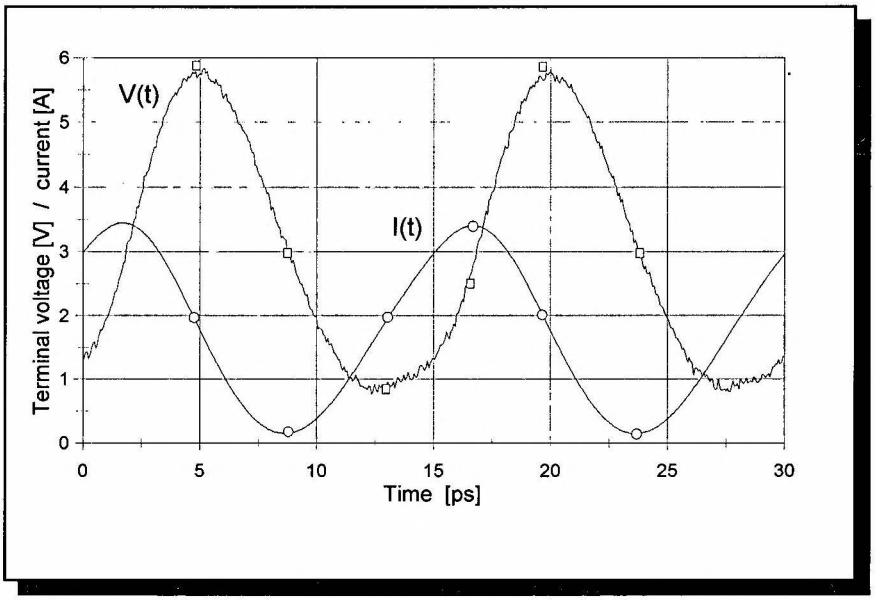


Figure 3.5: The simulated voltage V(t) and current I(t) waveforms obtained by the leap-frog algorithm. Simulation results by Tully [78] are indicated with squares.

The efficiency of both the MC-PVM and LF algorithms has been quantified as the gain in computational speed achieved by employing multiple slaves relative to a master with single slave configuration. The curves of the obtained speed-up as a function of the number of slaves are given in Figure 3.6. The LF algorithm shows a remarkable 87% efficiency when employing 19 slave processors, as apposed to a 71 % efficiency obtained by the conventional MC-PVM algorithm.

In absolute timing terms, the LF implementation is able to execute 385ps simulation time in a real-time hour. The poorer performance of the MC-PVM in terms of speed-up can be attributed to the slaves being idle while the master solves Poisson's equation. Any increase in the master's computational load resulting from, for example, performing two-dimensional device simulations, will exacerbate this situation. The LF implementation will have to be considered in these instances.

3.3.3.2 Load sharing

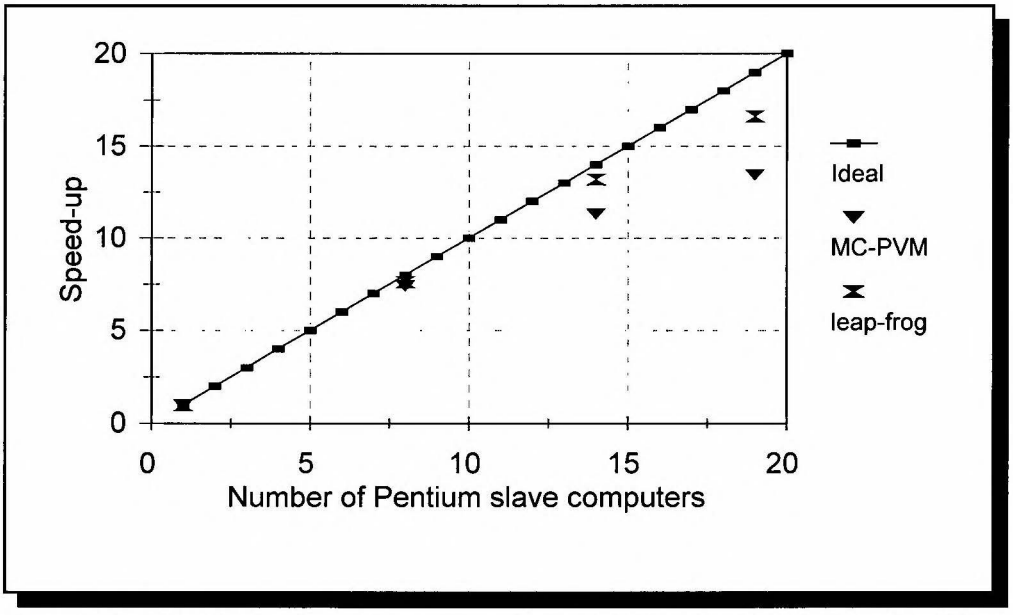


Figure 3.6: The obtained speed-up curves for the MC-PVM and leap-frog algorithms as a function of the number of slaves employed. The ideal (linear) speed-up curve is also shown.

For optimal efficiency, the computational load has to be shared among the processors in relation to their respective processing power. The computational load of each slave processor is directly proportional to the size of the subensemble dedicated to it. For a network of equally fast

processors this implies that the size of each subensemble must be kept equal *throughout the simulation*. This is not directly enforced in the MC-PVM algorithm, or its LF derivative. However, the random nature of the simulation ensures that the respective sizes of the subensembles stay equal, on average, for the duration of the simulation because each subensemble essentially undergoes the same stochastic processes. This is graphically illustrated in Figure 3.7 for a simulation comprising 14500 particles divided between two slaves. There is a maximum difference of 4% between the two subensemble sizes which, incidentally, also holds true for more than two slaves.

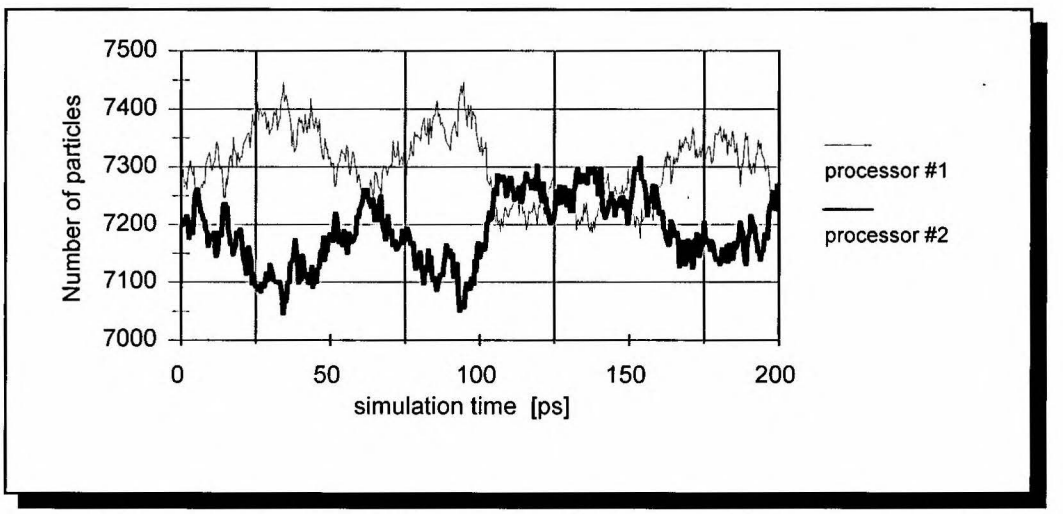


Figure 3.7: A graphical illustration of load sharing for a typical MC-PVM simulation. An ensemble of 14 500 particles, divided equally between two slave processors, has been simulated. The computational load of each processor, which is directly proportionate to the respective subensemble sizes, shows a maximum relative difference of 4 %.

It could be argued that additional load sharing control mechanisms be implemented to improve the computational efficiency even further. The network, however, is a serializing component in the sense that only one slave can communicate its data to the master at any given time. Simultaneous network access by more than one slave will enhance the data package collision rate resulting in lower efficiency. Small variations in the load of each slave will therefore aid the overall computational efficiency by reducing the occurrence of simultaneous access of the network by two or more slaves. Dynamic load sharing techniques will therefore artificially cause slight imbalances of the load of each slave. This is inherent to the MC simulation as discussed above. Additional load sharing techniques, which will have resulted in more network communication and complex coding due to the exchange of particles among processors, are therefore unnecessary. It should be noted that, in contrast, dynamic load sharing has to be explicitly implemented in the PMC-3D algorithm for optimal performance due to the transfer of particles among processors as they move through the device.

3.3.3.3 Network loading

The transmission of data across the network will undoubtably have an impact on the performance of PVM and needs to be investigated. However, the CSMA/CD nature of the network makes it difficult to quantify network loading in terms of data transfer rates.

It has been measured with file transfers between different nodes in the network that for the local Gigabit Ethernet network a data transfer rate of at least 50 MB/s can be sustained under nominal loading conditions. For the simulation presented in the previous section, where the transmitted one-dimensional data structures consist of 2000 double precision floating point numbers (an electric field and a temperature value at each of the 1000 gridpoints), approximately 16 KB of data is sent to and from each slave with every time step. Each time step took approximately 0.05 seconds to be simulated. This translates into a 320 KB/s data stream when employing 19 slaves. Assuming a 50 MB/s available data transfer rate, the network communication accounts for a small part of the total simulation time, approximately 0.6%. This also illustrates the superiority of the Gigabit Ethernet to the more conventional 100MBps Ethernet network, in which case it may be assumed that inter-node communication will account for >6% of the total simulation time.

Internal network communication will play an increasing role when using more processors, as is evident from the flattening of the speed-up curves, or more gridpoints. When considering two-dimensional simulations the device is divided spatially into a two-dimensional mesh of typically1000×100 grid points. The size of the transmitted data package will consequently increase hundredfold, resulting in a significant rise in network communication. Faster coarse grained parallel architectures, such as the Beowulf Clusters will have to be considered in these instances.

3.4 CONCLUSION

An efficient parallel implementation of the Monte Carlo particle simulation technique on a network of personal computers has been introduced. The parallel implementation, together with a more efficient LF derivative, have been applied successfully to the simulation of mm-wave Gunn-effect oscillators. It has been shown that, for one-dimensional simulations, the network communication does not have a significant effect on the overall speed-up obtained by the proposed parallel algorithms. The utilisation of a network of personal computers places the Monte Carlo particle simulation well within reach of the smaller research centres and universities that do not readily have access to massive multiprocessor machines.

Chapter 4

THE GUNN OSCILLATOR: FUNDAMENTALS

4.1 INTRODUCTION

The aim of this chapter is to sensitise the reader to the various principles and characteristics of the Gunn oscillator. Emphasis is placed on those features that will be referenced in the following chapter on the optimisation problem. Extensive reviews of the Gunn effect oscillator may be found in the literature, for example [79], [80].

The transferred electron effect and how it enables microwave power generation - the Gunn effect in the strict sense - will be discussed in the following section. This is followed by a treatment of various Gunn oscillator fundamentals.

4.2 THE GUNN EFFECT IN THE STRICT SENSE

4.2.1 The transferred electron mechanism and Negative Differential Resistance

When no bias is applied to a semiconductor, almost all the electrons occupy the Γ-valley, since their respective thermal energies are usually much less than the energy gap Δ. If the sample is biassed, the electrons are accelerated by the applied electric field and may gain sufficient energy to transfer to the L- and X-satellite valleys. This phenomenon is verified by Monte Carlo simulations and illustrated by the graphs in Figure 4.1 [81]. (These simulations are based on a simple two valley energy band model with $\Delta = 0.36$ eV.)

It is clear from the graphs in Figure 4.1 that the mean electron energy increases for increasing field bias. This results in an ever increasing number of electrons gaining enough energy (Δ) to bridge the gap between the Γ- and L-valleys and *transfer* from the lower Γ-valley to the upper L-valleys. Significant population of the L-valley takes place for electric field bias exceeding 0.4 Mvm^{-1}.

The electrons that have transferred from the Γ-valley to the L-valleys will immediately move slower due to the increase in their effective mass. The average drift velocity of the electrons, and

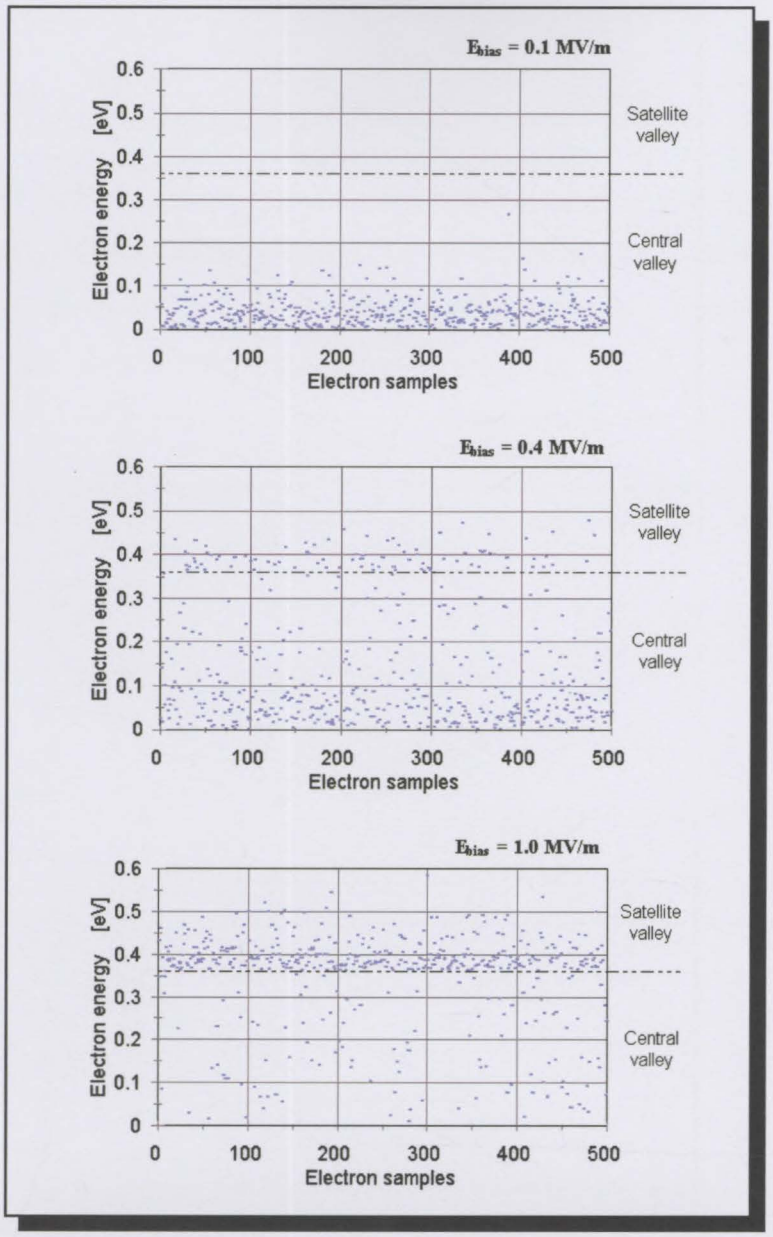


Figure 4.1: Valley occupation of electrons in bulk GaAs for three applied electrical fields E_{bias}, namely $0.1MVm^{-1}$, $0.4MVm^{-1}$ and $1MVm^{-1}$ respectively. The mean energy of the ensemble of electrons increases with stronger applied fields. Significant population of the satellite L-valley takes place at fields exceeding of $0.4MVm^{-1}$.

consequently the current, will therefore decrease with an increase in the applied electric field. This manifests a region of negative differential resistance (NDR) for applied fields exceeding about 0.4 MVm^{-1}.

4.2.2 The formation of Gunn domains

The question of exactly how the NDR phenomenon in GaAs results in Gunn oscillations can now be answered with the aid of Figure 4.2 [81]. A sample of uniformly doped n-type GaAs of length L is biassed with a constant voltage source V_0. The initial electrical field is therefore constant and its magnitude given by $\xi_0=V_0/L$. From the bottom graph in Figure 4.2 it is clear that the electrons flow from cathode to anode with constant velocity v_3.

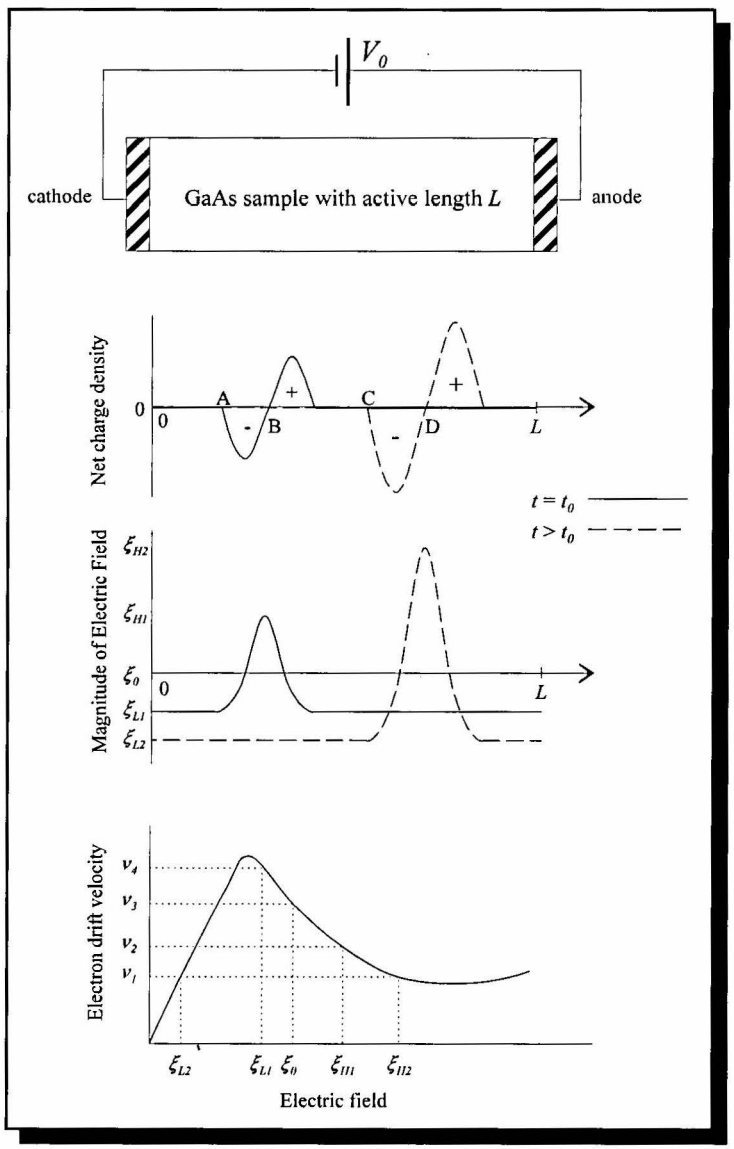


Figure 4.2: An illustration of the formation of Gunn domains.

It is now assumed that a small local perturbation in the net charge arises at $t = t_0$, indicated by the solid curve in Figure 4.2. This non-uniformity can, for example, be the result of local thermal drift of electrons. The resulting electrical field distribution is also shown (solid curve). The electrons at point A, experiencing an electric field ξ_{L1}, will now travel to the anode with velocity v_4. The electrons at point B are subjected to an electrical field ξ_{H1}. They will therefore drift towards the anode with velocity v_2 which is smaller than v_4. Consequently, a pile-up of electrons will occur between points A and B, increasing the net negative charge in that region. The region immediately to the right of point B will become progressively more depleted of electrons, due to their higher drift velocity towards the anode than those at point B.

The initial charge perturbation will therefore grow into a dipole domain, commonly known as a Gunn domain. Gunn domains will grow while propagating towards the anode until a stable domain has been formed. A stable Gunn domain is shown at a time instance $t > t_0$, indicated by the dashed curve. At this point in time, the domain has grown sufficiently to ensure that electrons at both points C and D move at the same velocity, v_1, as is clear from the bottom graph in Figure 4.2.

It is important to note that the sample had to be biassed in the NDR region to produce a Gunn domain. Once a domain has formed, the electric field in the rest of the sample falls below the NDR region and will therefore inhibit the formation of a second Gunn domain.

As soon as the domain is absorbed by the anode contact region, the average electric field in the sample rises and domain formation can again take place. The successive formation and drift of Gunn domains through the sample leads to a.c. current oscillations observed at the contacts.

4.2.3 Transit-time devices

The mode of operation described above is referred to as the Gunn mode. In this mode the frequency of the oscillations is determined primarily by the distance the domains have to travel before being annihilated at the anode. This distance is roughly the length of the active region,

L, of the diode. These diodes are therefore also referred to as "transit-time" devices. The approximate relationship between the transit length of the diode and the fundamental harmonic component of the output power is given in Figure 4.3.

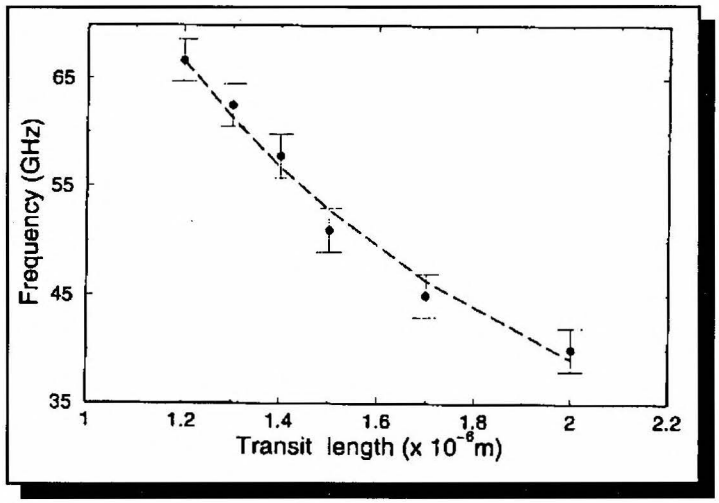


Figure 4.3: Variation of transit frequency with active layer length (from [82]).

4.2.4 The $n_0 \cdot L$ and $n_0 \cdot L^2$ products

The discussion of the Gunn effect is concluded by addressing two device criteria required for domain formation, namely the $n_0 \cdot L$ and $n_0 \cdot L^2$ products.

Small signal stability analyses of Gunn diodes biassed at constant voltage [79], [80] have shown that these devices are unstable only if the product of the nominal doping n_0 and length L of the active region exceeds a certain critical value $(n_0 \cdot L)_{crit} \approx 10^{16} m^{-2}$. Satisfactory dipole domain formation, which constitutes instability, is therefore only possible if

$$n_0 \cdot L > 10^{16} \, m^{-2} \, . \tag{4.1}$$

Another phenomenon that affects instability is charge diffusion. Any charge perturbation will be "smoothed out" through diffusion over a few Debye lengths. Diffusion acts as a loss mechanism inhibiting the growth of a charge disturbance. However, it can be shown that

sustainable domain growth is still possible in the presence of diffusion if [80]

$$n_0 \cdot L^2 > 5 \cdot 10^9 \text{m}^{-1} . \tag{4.2}$$

Both these criteria need to be satisfied to ensure proper domain formation. The diffusion limitation given in (4.2), however, is the dominant condition for very small devices (typically with sub-micron active lengths) due to the L^2 factor. Although the numerical values given in (4.1) and (4.2) are only estimates, they serve as plausible starting points when designing the doping profile of a Gunn diode.

4.3 OVERVIEW OF GUNN OSCILLATOR FUNDAMENTALS

4.3.1 Modes of operation

The Gunn domain formation, and the subsequent domain drift, lies at the core of microwave power generation by Gunn diodes. Various modes of operation, however, do exist. These modes are dependent on factors such as device geometry, doping profile, frequency and external circuitry. The various modes are generally classified as [79]

- Delayed dipole-domain
- Quenched dipole-domain
- Limited space-charge accumulation (LSA)
- Travelling accumulation layer

For the purposes of this work, only the travelling accumulation layer mode will be discussed briefly, since, as will be evident later, mm-wave devices are too short for dipole domains to form.

- *Travelling accumulation layer*

The travelling accumulation layer mode is the dominant mechanism of power generation in millimetre wave applications. The accumulation layer is a form of domain first studied by Kroemer [38]. In contrast to the dipole-domain modes, where the domains consist of successive

accumulation and depletion regions, the "domains" in the accumulation mode consist of only layers of excess electrons.

The charge accumulation layer grows as it traverses the active region from cathode to anode. As is the case with the dipole-domain modes, there also exists a "dead zone" at the cathode side of the active layer which diminishes the diode's output efficiency. The reduction of the dead zone lies at the core of the techniques to optimise the efficiency of Gunn diodes. The optimisation problem is the central theme of the next chapter.

4.3.2 Microwave performance

The microwave performance characteristics of the Gunn oscillator, or TEO (transferred electron oscillator) will now be discussed. The main emphasis is on the output power characteristics and frequency stability of the TEO, since these will be of primary concern to the optimisation problem covered in the following chapter.

The output power of a certain TEO, for a given diode, depends mainly on three parameters, namely frequency of operation, bias conditions and operating temperature. All of these parameters play a role in the device simulations of the next chapter. To "measure" and optimise the output power of a TEO by means of the MC device simulations therefore require insight into the effects that these parameters have on the output power of the oscillators.

4.3.2.1 Output power versus frequency characteristic

The Gunn diode is a transit-time device. Its "free running" transit-frequency is largely determined by the time the successive dipole or charge accumulation domains take to traverse the active region of the device. It can therefore be expected that optimum power generation will occur around the transit-frequency of a given diode, in which case the domains have maximum time to grow.

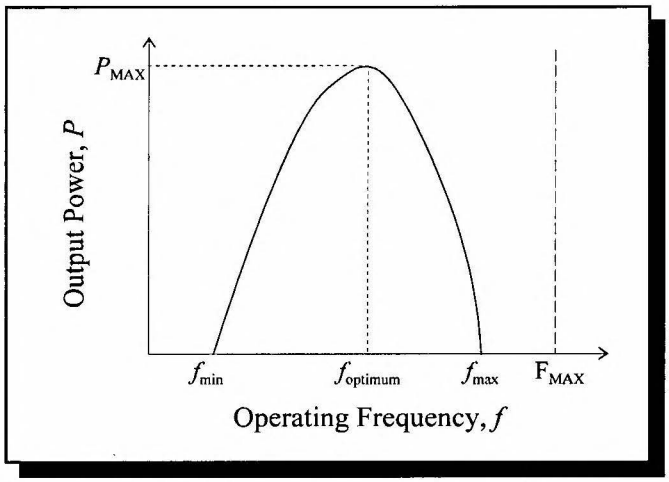


Figure 4.4: Typical output power variation with operating frequency of a Gunn diode. Optimal power output is reached at $f_{optimum}$, which is close to the transit-time frequency of the device. There exists an absolute maximum frequency, F_{MAX}, above which no microwave power is generated.

Gunn diodes are, however, capable of operating over a large frequency range around the transit-time frequency if placed in a properly designed resonant circuit. It has been found experimentally that Gunn diodes can be tuned over as much as 50% of their transit-time frequency [22], but with diminished output power. The output power variation with frequency for a typical diode is illustrated in Figure 4.4.

It is appropriate at this stage to discuss the upper frequency limit of operation for Gunn diodes. In addition to the "dead zone" effect, which becomes increasingly prominent in shorter devices, there is also a mechanism intrinsic to the material that determines the absolute cut-off frequency.

The Gunn diode's negative differential resistance characteristic, on which power generation is dependent, is a direct consequence of electrons scattering from the central valley to the satellite valleys. These scattering events are of finite duration, given by the energy-relaxation time. It can therefore be expected that, when the oscillation period reaches the same order as these scattering durations, the NDR region is to deteriorate and vanish. Attempts at predicting this limit have, however, not been precise in the region of 30 - 150GHz [22]. A recent Monte Carlo study of high-field electron transport in GaAs estimates an upper limit of 105GHz [84].

4.3.2.2 Output power versus temperature characteristic

The effect that temperature has on the input-to-output power conversion efficiency of the Gunn diode is of considerable practical interest. Self-heating of the device through internal power dissipation raises the chip's temperature to well above room temperature. (This will be illustrated in the next chapter.)

An increase in temperature causes the NDR slope of the velocity-electric field characteristic to flatten (see Figure 2.15) This effectively decreases the NDR of the diode and its ability to generate microwave power. A rise in operating temperature therefore reduces the input-to-output power conversion efficiency of the device. Figure 4.5 illustrates this effect for a commercially available diode in the K-band.

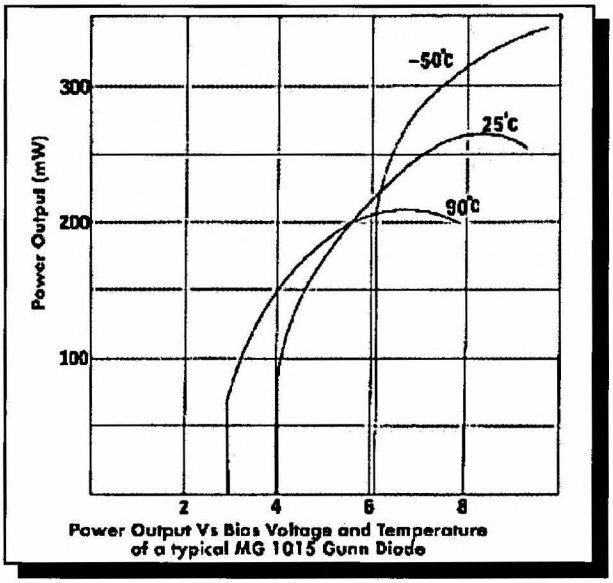


Figure 4.5: Temperature dependance of the output power of a commercially available Gunn diode at frequencies in the region of 20 GHz [18].

Proper heat sinking is evidently of the utmost importance in Gunn diodes due to their inherently low power conversion efficiencies. Heat sinking is discussed in Section 4.3.3 below.

4.3.2.3 Output power versus bias voltage characteristic

The dependence of the output power on the bias voltage is illustrated in Figure 4.6 (from [32]).

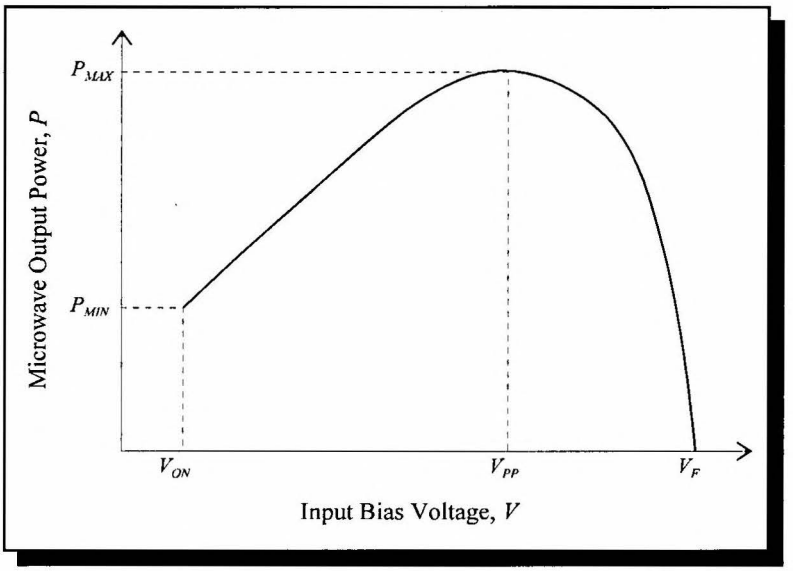


Figure 4.6: Output power as a function of bias voltage for a typical Gunn diode [32].

A minimum bias voltage - the "turn-on" voltage V_{TO} - is required for any power generation. This is a direct consequence of the transferred electron effect where the electrons need to be heated sufficiently by the bias field before any electron transfer to the satellite valleys can take place.

As the bias voltage is increased from V_{TO}, the output power increases in an almost linear fashion, before levelling off and reaching a maximum at the "peak-power voltage", V_{PP}. The linear increase of the output power is attributed to more power being supplied to the diode by the biasing circuit. This in turn increases the power dissipation, and consequently the temperature, within the device. The higher device temperature reduces the diode's input-to-output power conversion efficiency, as indicated in the previous section. The peak in the output power is reached at the bias point where the reduction in efficiency compensates fully for the increase of bias voltage and input power.

The output power continues to fall as the bias voltage is increased past V_{PP}. The diode will eventually "burn out" and fail at the device failure voltage, V_F. The output power continues to fall as the bias voltage is increased past V_{PP}. Avalanche breakdown, that sets in at high electric fields, may contribute to the heating of the device and speed up its destruction [80].

4.3.2.4 Noise and frequency stability

Oscillators are incapable of generating perfect sinusoidal waves. The desired output frequency is always contaminated by stochastic fluctuations of the amplitude and frequency, or *noise*. In typical continuous wave applications, where a constant frequency is required, the random modulation of the oscillator's frequency, or FM noise, is of particular importance.

Two significant contributing factors to FM noise in Gunn diodes are noise in the bias voltage and the random nature of domain nucleation.

Fluctuations in the diode's bias voltage modulate the electrons' velocity. This causes frequency drift around the transit-frequency. In general, the diode's transit-frequency *decreases* with an increase in bias voltage. The electrons gain, on average, more energy if the bias is increased, and as a result their effective masses will increase due to enhanced transfer from the central to satellite valleys.

The contribution of domain nucleation to FM noise is readily understood in light of the underlying random processes. The transfer of electrons from the central to satellite valleys is governed by random scattering events, as described in Chapter 2. Domain nucleation, which depends on the transferred electron effect, is therefore fundamentally non-deterministic. Indeed, domains will not nucleate from the exact same point with every cycle. This effectively modulates the diode's active length and consequently the transit time frequency at random. An apparent way of minimising the noise contribution associated with domain nucleation, is to reduce the uncertainty of where the domains will nucleate from at the start of every transit cycle. The hot-launcher technique described in Chapter 1 accomplishes this.

4.3.3 Heat sinking

The detrimental effects of temperature on the output power and frequency stability of the Gunn oscillator have been stated in the previous sections. Gunn diodes are notoriously inefficient, which means that much of the input power is converted into heat. It is therefore very important to construct the diode in such a way that excess heat is efficiently channelled away from the diode.

Most of the heat is generated in the active region of the diode, and more specifically, near the anode. The heat sink should therefore be placed as near to the active region as possible, and on the anode side of the diode. The so-called Integral Heat Sink (IHS) is a practical method of achieving this. The fabrication of the IHS is illustrated in Figure 4.7. The IHS is bonded to a

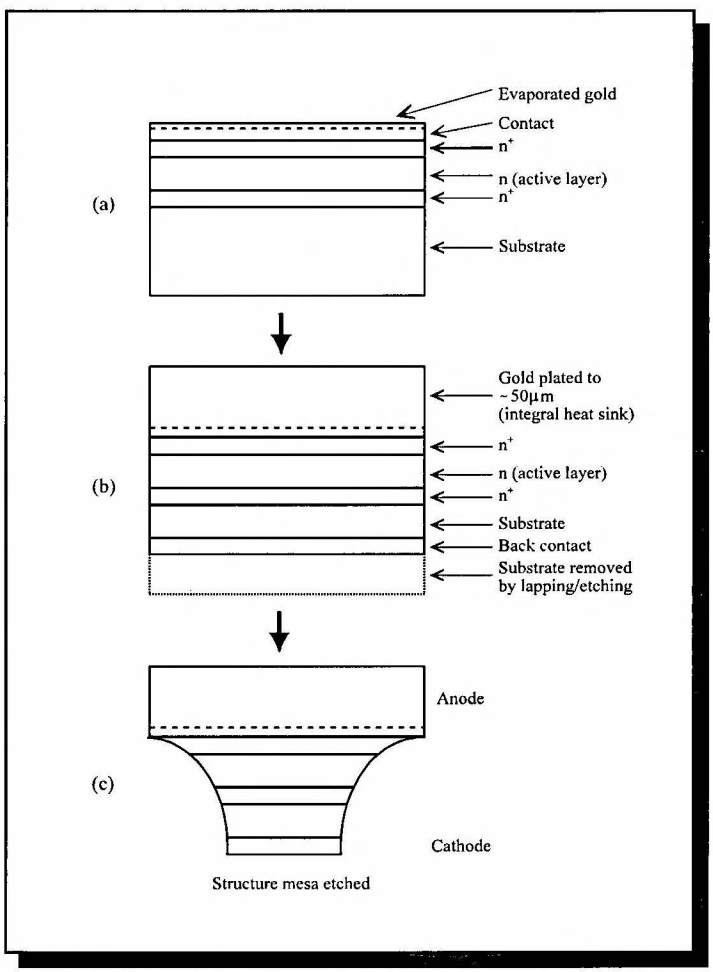


Figure 4.7: Integral heat sink fabrication process.

heat stud for support. In general, gold is used for the IHS and copper for the heat stud. The thermal conductivity of gold and copper is, respectively $K_{AU} = 311$ $Wm^{-1}K^{-1}$and $K_{Cu} = 393$ $Wm^{-1}K^{-1}$ at 400K.

Especially in high frequency continuous wave applications (with the focus here on 94 GHz) the Gunn diode's efficiency drops even further. Severe thermal degradation of the output power and possible overheating of the device become critical factors that have to be addressed. Furthermore, for the double-domain diodes which will be considered in the following chapter, the input power is approximately four times the input power of a single domain diode. The need for improved heat sink material in these applications is apparent. (This will indeed be verified by the simulations presented in the next chapter.)

The use of diamond IIa for heat sinking has been found to be remarkably good for spreading the thermal resistance at the semiconductor-heat sink interface. This is due the high thermal conductivity of diamond, even at elevated temperatures. From Figure 4.8 it is clear that the thermal conductivity of diamond IIa is $K_{IIa} \approx 1250$ $Wm^{-1}K^{-1}$ at 450K, almost five times higher than that of copper.

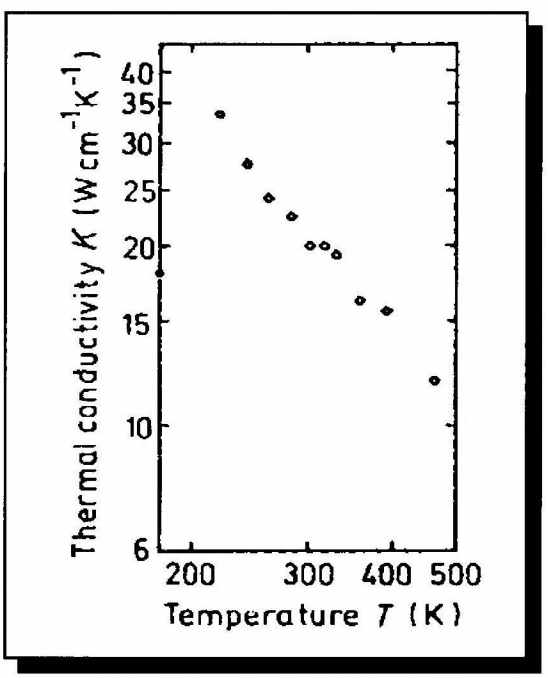


Figure 4.8: Thermal conductivity of diamond IIa as a function of temperature (from [85]).

Chapter 5

GUNN DIODE OPTIMISATION

5.1 INTRODUCTION

The broad approach to the optimisation problem was outlined in Chapter 1. The aim of this chapter is to detail the application of this approach to a 94 GHz double-domain diode. The simulation tool that will be used, namely a parallel implementation of the Monte Carlo particle simulator, has been discussed and evaluated in great detail in previous chapters of this work.

The chapter outline follows the logical steps in the optimisation procedure. First, a single-domain benchmark diode that compares with state of the art commercial diodes, will be established. This diode will then be replicated into a double-domain diode and evaluated. Thereafter, further optimisation of the double-domain will be done to compensate for increased operating temperatures compared to that of the single-domain diode. As pointed out previously in Chapter 1, in the absence of the temperature effects, the double-domain diode will yield a four-fold increase in output power over its single-domain counterpart. This is, however, a very optimistic view, as thermal constraints play an integral role in multi-domain diodes.

An important aspect that has to be evaluated is the sensitivity of the double-domain diode to profile variances in the two domains. The chapter will conclude with an investigation of these effects on the microwave performance of the diode.

A critical evaluation of the double-domain diode will follow in the next chapter.

5.2 OPTIMISATION OF SINGLE-DOMAIN DIODE

As already alluded to in Chapter 1, the optimisation avenues available to the designer are:

- Hot injection of electrons at the cathode through heterostructure injectors
- Doping notch on the cathode side of the active region
- Graded doping profile of the active region

These avenues afford the designer with the necessary tools to tailor the electric field throughout the device to ensure optimum operation. The interrelated effects of each of these mechanisms make it impossible to evaluate them completely independently. It was therefore decided to establish a benchmark diode, based on design parameters implemented in practical diodes and reported in the literature. The hot injection diode of Couch *et al* [27], based on work done in 1989, has been chosen for this purpose, since the structure of its hot launcher, as well as active region, is well documented. Also incorporated in the initial design, that was not implemented in the diode by Couch *et al*, are a doping notch and grading of the active region for improved efficiency. Once a set of design parameters has been established, the design can be optimised.

The doping profile of a typical single-domain Gunn diode with hot launcher is shown below in Figure 5.1. It consists of the following regions:

(I), (V): Highly doped ohmic contact regions

(II): Undoped, graded $Al_xGa_{1-x}As$ layer with linearly increasing Al concentration

(III): Doping notch

(IV): Active region (with graded doping profile - grading not shown)

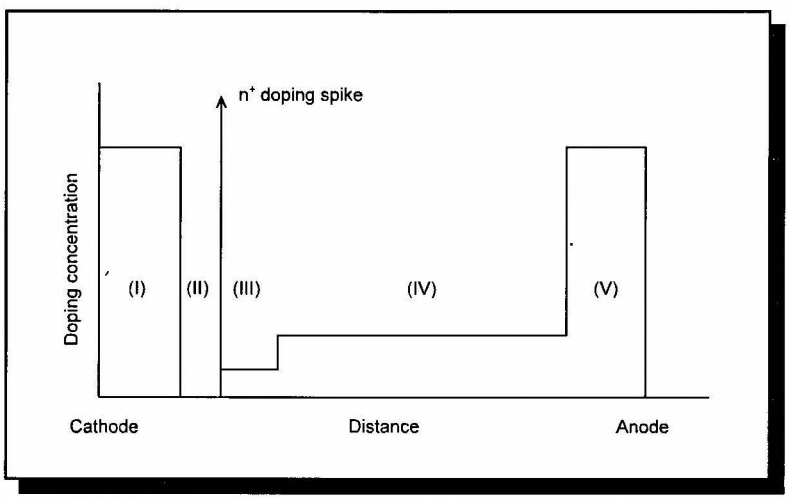


Figure 5.1: Doping profile of a typical single-domain Gunn diode with hot injection. Numerical values are given in the text.

The launcher consists of two layers, namely region (II) followed by a narrow n^+ doping spike.

This spike serves as a non-equilibrium connector [28] to prevent depletion, set up by the forward biased injector, from extending into the active region. This will prevent the formation of Gunn accumulation domains.

The launcher employed is the same as that proposed by Greenwald [23]. The general triangular profile of a launcher is shown in Figure 5.2 for zero and forward bias. It is important that the critical field for Gunn domain formation be fixed close to the cathode for a wide range of bias voltages. If this is not the case, the point at which the critical field is achieved will greatly vary with applied voltage. This in turn will cause the operating frequency to be highly dependent on bias voltage because of the modulation of the active region length. This is undesirable. By ensuring that the slope of the second arm of the triangle is high, i.e. l_2 very short, the critical field can be forced adjacent to the launcher, irrespective of bias.

The implemented launcher has a linearly graded, increasing Al composition over the distance l_1, followed by an abrupt discontinuity ($l_2 = 0$). This will translate into a diode of which the operating frequency is essentially independent of bias voltage.

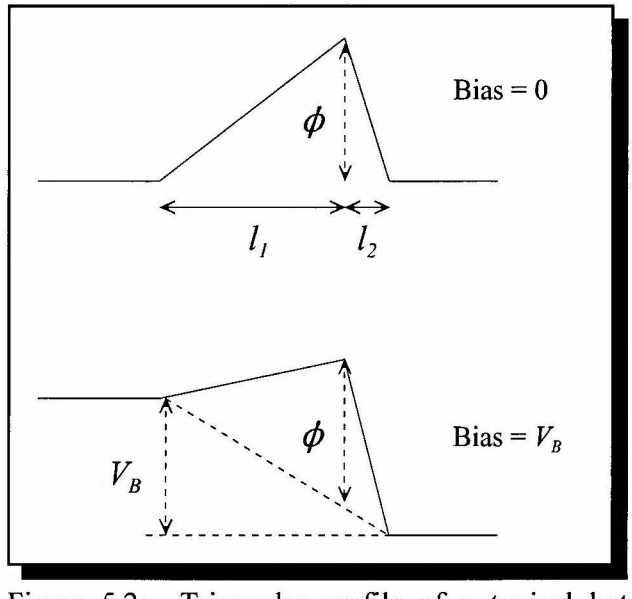


Figure 5.2: Triangular profile of a typical hot electron launcher at zero and forward bias.

The energy gain of electrons traversing the launcher is dependent on the Al composition profile.

Selecting an appropriate value of x deserves consideration. A value of >0.3 would ensure immediate transfer of electrons to the higher L-valley. Selecting too high a value would, however, reduce the current to such an extent as to lower harmonic efficiency. A further limiting factor is the existence of deep levels in GaAs for x > 0.23. This will also lower the current density and efficiency [23]. A maximum value of x = 0.3 has been chosen as compromise. This is also the value implemented by Couch *et al.* . The energy gained by electrons crossing the heterojunction with x = 0.3 is approximately 0.250eV. This will greatly enhance intervalley transfer (the valley separation between the central and L-valleys is 0.284eV).

5.2.1 Benchmark diode

5.2.1.1 Structure description

- Hot launcher

The Al concentration is graded from 0 to 0.3 across 50nm, followed by a 10nm wide n^+ spike with doping 1×10^{24} m^{-3}.

- Active region doping profile

A doping notch, preceding the active region on the cathode side, has been implemented. It consists of a 0.2μm undoped region. From an extrapolation of Figure 4.3 the active region should be ≈1.6μm long to coincide with a fundamental frequency of 47GHz. The required active region length has been shortened accordingly to 1.4μm to compensate for the inclusion of the notch. For efficient Gunn operation, the minimum required doping density is determined from (4.1) and (4.2) to be $0.71\times10^{22}m^{-3}$. Choosing the doping concentration close to the minimum will degrade the diode efficiency, as will a value that is much higher. The latter will cause a marked increase in the d.c.bias current which will lead to excessive thermal losses. This is especially the case for double-domain operation where the d.c. power dissipation increases quadratically with an increase in d.c. current.

An initial value of $1.2\times10^{22}m^{-3}$ has therefore been chosen. (As will be evident later, this value is non-optimal for single-domain operation, but proves to be optimal in the case of double-domain operation.)

Grading of the active region is incorporated in anticipation of higher efficiencies compared to flat doping profiles. The doping concentration is increased over the last 25% of the active region (toward anode) from $1.2\times10^{22}m^{-3}$ to $2\times10^{22}m^{-3}$ (grading factor 1.67).

- Diode diameter

From Couch *et al*, a diameter of 75μm has been chosen. This yields favourable admittance levels for matching the diode to external circuitry.

- Heat sinking

A copper heat sink is used. As will be discussed later, the use of a diamond heat sink is proposed for double-domain operation.

5.2.1.2 Microwave performance

The diode has been simulated with an applied voltage of the form given in (2.83). The simulation results are summarised in Table 5.1. Also given are the harmonic components of the impressed voltage at which the maximum output power for each bias voltage is obtained. It is clear from Table 5.1 that the optimum bias voltage occurs at ≈3.5V. The output power flattens off for $V_{DC} > 3.5$V, while the efficiency decreases due to increased power dissipation.

The simulated RF admittance of the diode represents a parallel combination of a capacitor and negative resistance. This agrees with the model used by Haydl [22]. The value of the resistor, ≈-8Ω, presents a favourable value to the external matching network, and is in line with typical practical values.

TABLE 5.1

SIMULATED RESULTS OF BENCHMARK DIODE FOR VARIOUS BIAS VOLTAGES

Active region doping density N_{AR}	[10^{22}m^{-3}]	1.2	1.2	1.2
Bias voltage V_{DC}	[V]	3	3.5	4
Fundamental harmonic V_1	[V]	2.5	3.5	3.25
Second harmonic V_2	[V]	0.7	0.8	0.8
Phase difference ϕ	[°]	320	320	320
DC current	[A]	0.75	0.73	0.70
DC input power	[W]	2.3	2.6	2.9
Output power at 94GHz	[mW]	**30**	**36**	**37**
RF Efficiency	[%]	**1.3**	**1.4**	**1.3**
Diode admittance	[S]	-0.13 + j0.2	-0.12 + j0.18	-0.12 + j0.15

It is also evident from these simulations that bias current does not significantly decrease with increased bias voltage. This is to be expected from diodes that incorporate hot launchers, in contrast to conventional diodes where a marked decrease in current can be expected.

The next step will be to investigate the effect of increased doping concentration in the active region at the optimum bias voltage V_{DC} = 3.5V. This will then establish a benchmark diode against which to compare optimised double-domain diodes. The simulation results for the benchmark diode with increased active region doping densities are listed in Table 5.2.

The simulation results show that a maximum output power of ≈60mW may be achieved with ≈1.7% efficiency at 3.5V bias and active region doping densities in excess of 1.5×10^{22}m^{-3}. This corresponds to an n_0l-product of 2.1×10^{16}m^{-2}. It should be noted that this is not necessarily an optimum doping level for double-domain diodes. As can be seen from Figure 5.4, there is a marked increase in temperature with increased doping density. This will be even more severe for double-domain. Optimisation of the double domain will therefore include revisiting the active region doping profile to compensate for thermal effects.

The benchmark results compare favourably with similar diodes noted in Chapter 1 that yield, reproducibly, output power in the range 20 to 70mW at ≈1.6% efficiency. The benchmark diode is therefore indeed an appropriate standard against which further iterations can be evaluated.

TABLE 5.2

SIMULATED RESULTS OF BENCHMARK DIODE WITH VARIOUS ACTIVE REGION DOPING DENSITIES

Active region doping density N_{AR}	[$10^{22}m^{-3}$]	1.5	1.8	2.1
Bias voltage V_{DC}	[V]	3.5	3.5	3.5
Fundamental harmonic V_1	[V]	2.75	2.75	2.75
Second harmonic V_2	[V]	0.8	0.8	0.8
Phase difference ϕ	[°]	320	320	320
DC current	[A]	0.83	0.91	1.00
DC input power	[W]	3.0	3.3	3.6
Output power at 94GHz	[mW]	**49**	**57**	**59**
RF Efficiency	[%]	**1.6**	**1.7**	**1.6**
Diode admittance	[S]	-0.17 + j0.21	-0.20 + j0.27	-0.22 + j0.33

In conclusion, the benchmark values against which the final optimised diode will be evaluated, are an output power of ≈60mW and efficiency of ≈2%. Throughout, a negative resistance of the order ≈-5Ω would be sought to ensure appropriate impedance levels for matching purposes.

5.2.1.3 Internal field distribution

The internal electric field distribution at successive time intervals is shown in Figure 5.3 to illustrate the propagation of charge accumulation domains throughout the active region of the benchmark diode. It is clear from these profiles that hot injection ensures charge accumulation close to the cathode. The pronounced electric field at the anode side of the active region is also evident.

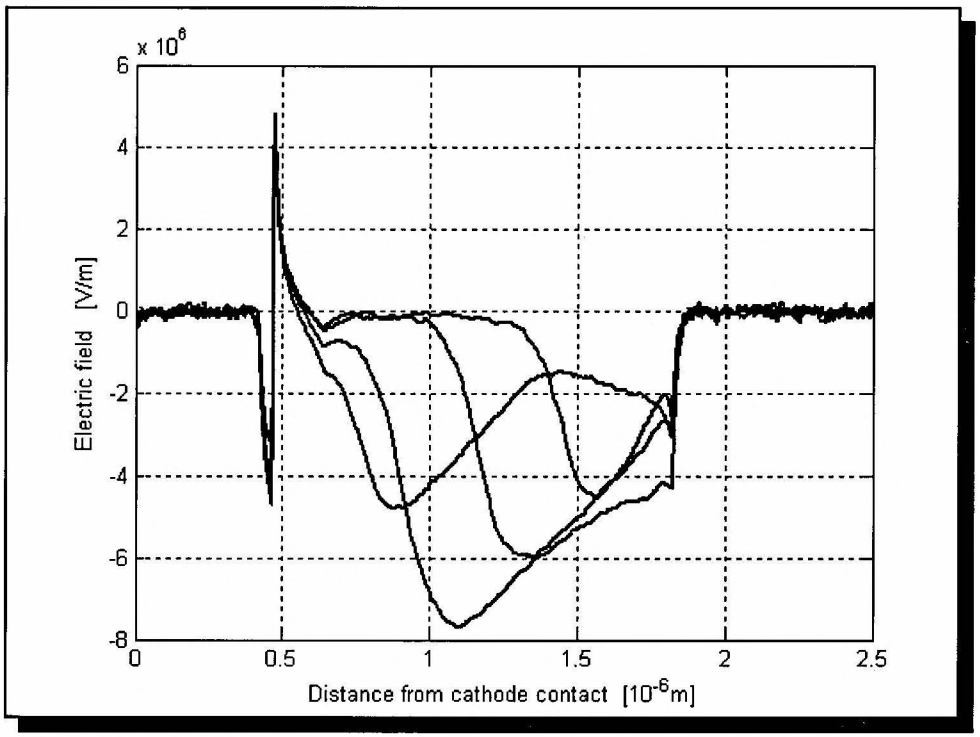


Figure 5.3: Internal electric field distribution of the benchmark diode at successive time intervals.

5.2.1.4 Operating temperature

For reference purposes the temperature profiles throughout the active region of the diode at various active region doping densities are shown in Figure 5.4. The bias voltage is 3.5V. There is a marked increase in operating temperature with increased active region doping density with the operating temperature being approximately proportional to the nominal active region doping density.

The dependence of operating temperature on bias voltage is illustrated in Figure 5.5 where a nominal active region doping density of $1.2\times10^{22}m^{-3}$ is assumed. The operating temperature is shown to be highly sensitive to an increase in bias voltage. This is to be expected because the power dissipation is proportional to the square of the bias voltage.

These figures point to the origin of non-linear thermal effects that are typical of Gunn diodes.

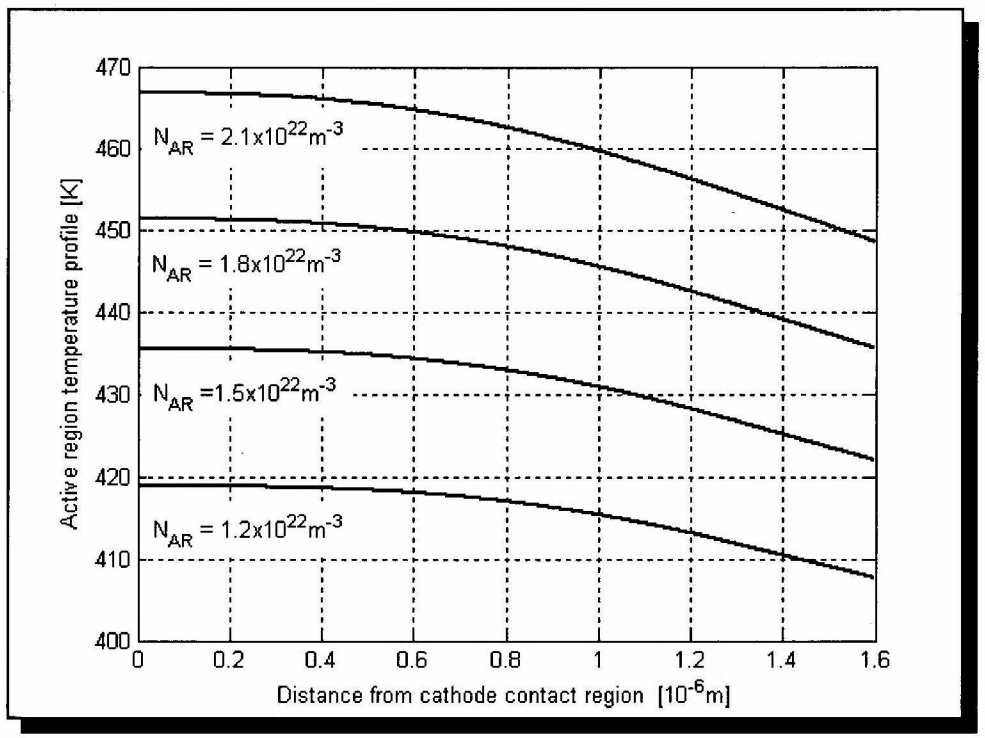


Figure 5.4: Active region temperature profile for various active region doping densities at $V_{DC} = 3.5$V.

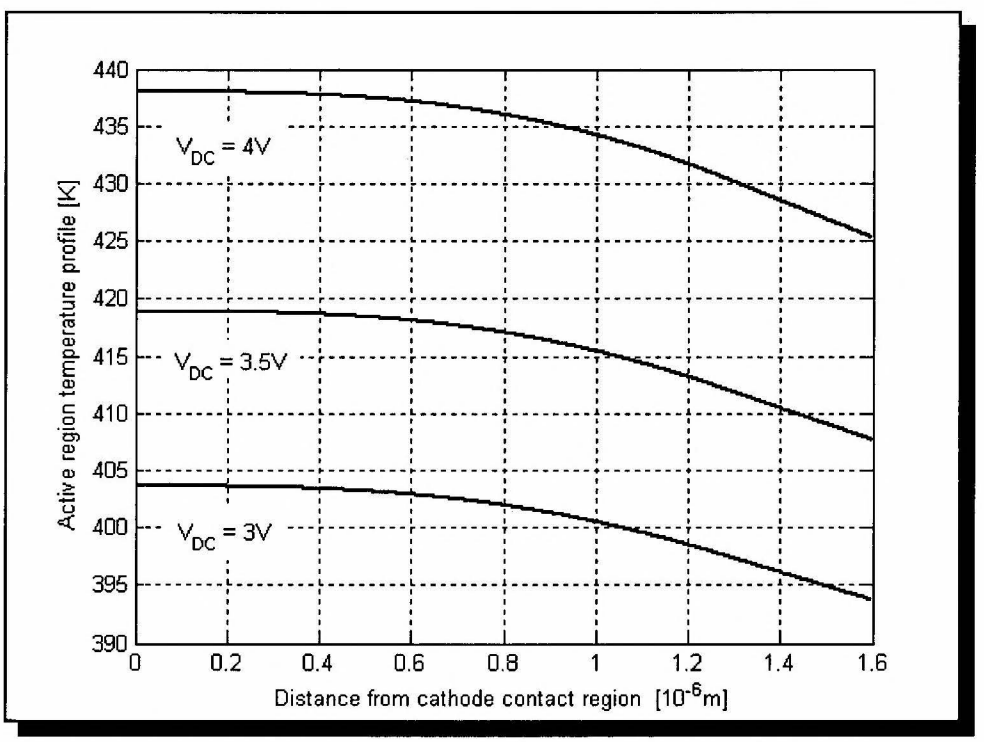


Figure 5.5: Active region temperature profile for the benchmark diode for various bias voltages ($N_D = 1.2\times10^{22}$m^{-3}).

5.3 Optimisation of double-domain diode

Once the benchmark has been established, further optimisation can be explored. It has been decided not to expend further effort on the optimisation of a single-domain diode, but rather to investigate the optimisation of a double-domain diode. This is because of the vast difference in operating temperature of a single-diode, compared to its double-domain counterpart, which would inevitably necessitate further optimisation of the double-domain diode.

The vastly increased operating temperature of the double-domain diode, compared to the single-domain diode, is due to two factors. First, the bias voltage has to essentially double to compensate for the two series active regions, while the admittance of the diode remains unchanged by doubling the diode's cross-sectional area. Second, significant heat generation occurs in the active region on the cathode side (Domain 1 in Figure 5.6) that is not in the vicinity of a heat sink.

The doping profile of a typical double-domain Gunn diode with hot launcher is shown in Figure 5.6. It consists of two series-coupled, hot-injection single-domain diodes. The two diodes are separated by a highly doped buffer region. The buffer region can be thought of as a low resistance connector between the two single-domain diodes. The width of the buffer region must be chosen large enough to quench the domains that enter it from the cathode. However, choosing it excessively long, will remove the first active region further away from the heatsink. This will result in increased operating temperatures in this domain. Furthermore, the buffer regions introduce additional series (and positive) resistance which translates into lower efficiency. This mechanism is of lesser importance due to the negligible resistance of the highly doped buffer regions [42].

A minimum buffer width is proposed by Tsay *et al* [42], based on simplified domain models. An extrapolation of their results suggests a minimum of $\approx 0.2\mu m$ for doping densities exceeding $2\times10^{22}m^{-3}$. In the following sections, an initial buffer width of $0.5\mu m$, with a doping density of $2\times10^{23}m^{-3}$, is assumed. It has been found that this is adequate for effective domain quenching.

Each of the single-domain diodes is identical to the single domain benchmark diode. A heat sink is applied to the anode side of the diode.

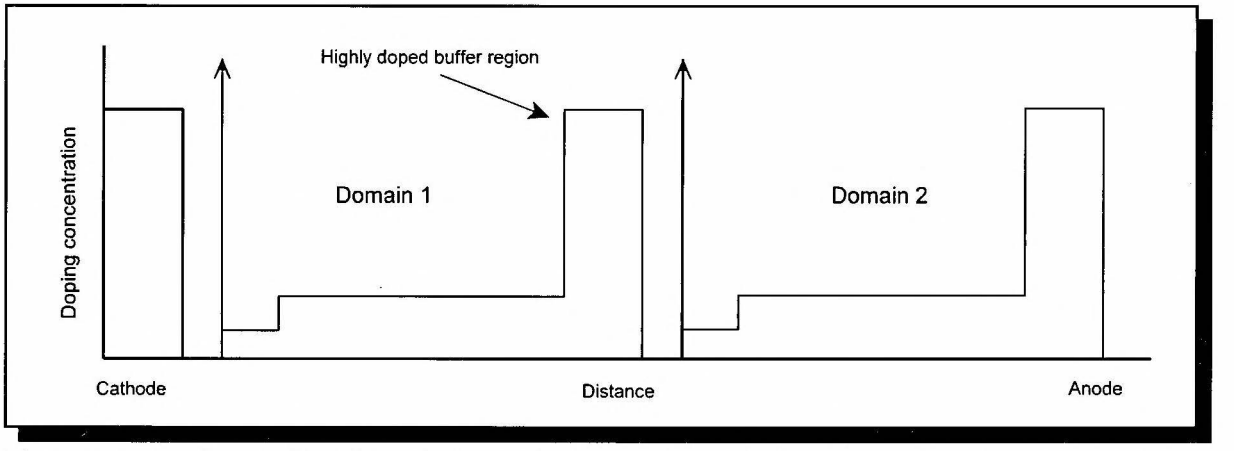


Figure 5.6: Doping profile of a typical double-domain Gunn diode with hot injection. Numerical values are given in the text.

5.3.1 Diode iteration #2D_01

A similar optimisation strategy to that of the single-domain benchmark diode will be followed. An initial active region doping density of $N_{AR} = 1.2\times10^{22}\text{m}^{-3}$ is assumed, with grading towards the anode identical to that of the single-domain diode. The microwave performance at various bias voltages is then evaluated to determine an optimum. (It should be noted that the optimum bias voltage cannot be expected to be merely double that of the single-domain diode due to the thermal effects already discussed.)

The double-domain diode diameter is 100μm which results in a cross-sectional area double that of the single-domain diode. This will ensure that the diode admittance remains similar to that of the single-domain diode, the effective length of which is half of that of the double-domain diode. A copper heat sink is used.

The microwave performance of the diode is summarised in Table 5.3. Also listed are the cathode contact temperatures for each case.

TABLE 5.3

MICROWAVE PERFORMANCE OF DOUBLE-DOMAIN DIODE ITERATION #2D_01 FOR VARIOUS BIAS VOLTAGES

Active region doping density N_{AR}	[10^{22}m^{-3}]	1.2	1.2	1.2
Bias voltage V_{DC}	[V]	5	6	7
Fundamental harmonic V_1	[V]	3.5	4.25	2.5
Second harmonic V_2	[V]	1.2	1.1	1.0
Phase difference ϕ	[°]	320	320	320
DC current	[A]	1.14	1.04	1.08
DC input power	[W]	5.8	6.3	7.7
Output power at 94GHz	[mW]	**100**	**115**	**40**
RF Efficiency	[%]	**1.7**	**1.8**	**0.5**
Diode admittance	[S]	-0.16 + j0.30	-0.20 + j0.20	-0.10 + j0.26
Cathode contact temperature	[K]	475	505	540

The optimum bias voltage is in the region of 5-6V. Thereafter the output power rapidly decreases with increased bias due to enhanced thermal losses. Maximum output power of ≈115mW may be expected from this diode, compared to ≈40mW from its single-domain counterpart (see Table 5.1).

5.3.2 Diode iteration #2D_02

An optimum bias voltage in the region of 5-6V has been established in the previous iteration. The effect of increased nominal doping density in the active regions will now be investigated. In anticipation of increased power dissipation with increased doping levels, a bias voltage of 5V, rather than 6V, is taken to counter this effect. A copper heat sink is still used.

Simulation results for this diode are summarised in Table 5.4

TABLE 5.4

MICROWAVE PERFORMANCE OF DOUBLE-DOMAIN DIODE ITERATION #2D_02 FOR VARIOUS ACTIVE REGION DOPING DENSITIES

Active region doping density N_{AR}	[10^{22}m^{-3}]	1.2	1.5	1.8
Bias voltage V_{DC}	[V]	5	5	5
Fundamental harmonic V_1	[V]	3.5	3.5	3.5
Second harmonic V_2	[V]	1.2	1	1.2
Phase difference ϕ	[°]	320	320	320
DC current	[A]	1.14	1.28	1.41
DC input power	[W]	5.8	6.6	7.2
Output power at 94GHz	[mW]	**100**	**83**	**81**
RF Efficiency	[%]	**1.7**	**1.3**	**1.1**
Diode admittance	[S]	-0.16 + j0.30	-0.19 + j0.29	-0.14 + j0.36
Cathode contact temperature	[K]	475	500	520

The optimum active region doping level is $N_{AR} = 1.2\times10^{22}$m^{-3}, which is lower than that of the single-domain diode. This can again be attributed to thermal effects, with an increased operating temperature at higher doping levels.

What is absolutely clear from diode iterations #2D_01 and #2D_02, is that thermal factors play a crucial role in the microwave performance of the double-domain diode. The optimum bias voltage for these diodes is less than double that for the optimised single-domain benchmark (as would be expected in the absence of thermal effects). Similarly, the optimum doping level for the double-domain is less than the optimum value for the single-domain diode, because of the enhanced power dissipation that results from higher doping levels. Barring thermal effects, one could therefore have expected better performance.

Comparing the optimum double-diode output power of 115mW at 1.8% efficiency to the benchmark values of 60mW at 2% efficiency, it is clear that the double-domain diode does not yield the anticipated four-fold increase in output power. Although the double-domain diode promises RF power exceeding the goal of 100mW, substantial enhancement of its performance

may be expected if thermal factors are addressed adequately.

The single-most effective tool in this endeavour would be the incorporation of a diamond heat sink. The thermal conductivity of diamond is $\approx$1250 $Wm^{-1}K^{-1}$ at 450K [see Figure 4.8], compared to copper with a thermal conductivity of 403 $Wm^{-1}K^{-1}$. A huge reduction in operating temperature can therefore be expected.

The same optimisation strategy as with the previous two iterations (where copper heat sinks were employed) will be followed.

5.3.3 Diode iteration #2D_03

The simulation for Diode iteration #2D_01 will now be re-run for the case of a diamond heat sink. The results are summarised in Table 5.5.

TABLE 5.5

MICROWAVE PERFORMANCE OF DOUBLE-DOMAIN DIODE ITERATION #2D_03 FOR VARIOUS BIAS VOLTAGES

Active region doping density N_{AR}	[$10^{22}m^{-3}$]	1.2	1.2	1.2	1.2
Bias voltage V_{DC}	[V]	5	6	7	8
Fundamental harmonic V_1	[V]	4	5	5.5	5.5
Second harmonic V_2	[V]	1.3	1.4	1.3	1.1
Phase difference ϕ	[°]	320	320	320	320
DC current	[A]	1.28	1.20	1.14	1.16
DC input power	[W]	6.6	7.3	8.1	9.4
Output power at 94GHz	[mW]	**102**	**156**	**155**	**142**
RF Efficiency	[%]	**1.6**	**2.1**	**1.9**	**1.5**
Diode admittance	[S]	-0.14+j0.30	-0.18+j0.26	-0.20+j0.18	-0.25+j0.15
Cathode contact temperature	[K]	420	440	465	490

As expected, the incorporation of the diamond heat sink leads to reductions in the average operating temperature, compared to the copper heat sink prototypes, of the order 13% (or 65°C for 6V bias). This translates into much enhanced output power levels. The optimum output level of ≈155mW at 6V bias represents a 35% increase over the copper heat sink prototype. Efficiency is also increased from 1.8% to 2.1%. Compared to the single-domain benchmark diode, a factor 2.6 increase in output power was achieved. The theoretical fourfold increase remains out of reach, principally due to thermal losses.

5.3.4 Diode iteration #2D_04

An optimum bias of in the region of 6-7V was established in the previous iteration. The influence of increased active region doping density will now be investigated for a 6V bias voltage. The results are summarised in Table 5.6.

TABLE 5.6

MICROWAVE PERFORMANCE OF DOUBLE-DOMAIN DIODE ITERATION #2D_04 FOR VARIOUS ACTIVE REGION DOPING DENSITIES

Active region doping density N_{AR}	[$10^{22}m^{-3}$]	1.2	1.5	1.8
Bias voltage V_{DC}	[V]	6	6	6
Fundamental harmonic V_1	[V]	5	4.75	4.5
Second harmonic V_2	[V]	1.4	1.3	1.5
Phase difference ϕ	[°]	320	320	320
DC current	[A]	1.20	1.37	1.53
DC input power	[W]	7.3	8.4	9.4
Output power at 94GHz	[mW]	**156**	**161**	**133**
RF Efficiency	[%]	**2.1**	**1.9**	**1.4**
Diode admittance	[S]	-0.18 + j0.26	-0.21 + j0.27	-0.14 + j0.32
Cathode contact temperature	[K]	440	465	490

From these results, it is evident that optimal performance is obtained through a wide range of doping densities > $1.2\times10^{22}m^{-3}$. (Incidentally, this is characteristic of doping notch profiles.)

Unlike the case with the benchmark diode, though, see Table 5.2, the expected increase in output power with increased doping levels is offset by enhanced thermal losses due to increased bias current.

Through extensive further simulation it has been found that the output power of ≈160mW at ≈2% efficiency obtained, is indeed the best performance that can be expected.

A further iteration has been investigated to assess the influence of various grading profiles of the active region doping density. Before continuing, though, it is instructive to observe the internal electric field and temperature profiles of the optimised diode iteration #2D_04 with N_{AR} = $1.5\times10^{22}m^{-3}$ at 6V bias.

- Electric field distribution

The internal electric field distribution at three successive time intervals is shown in Figure 5.7.

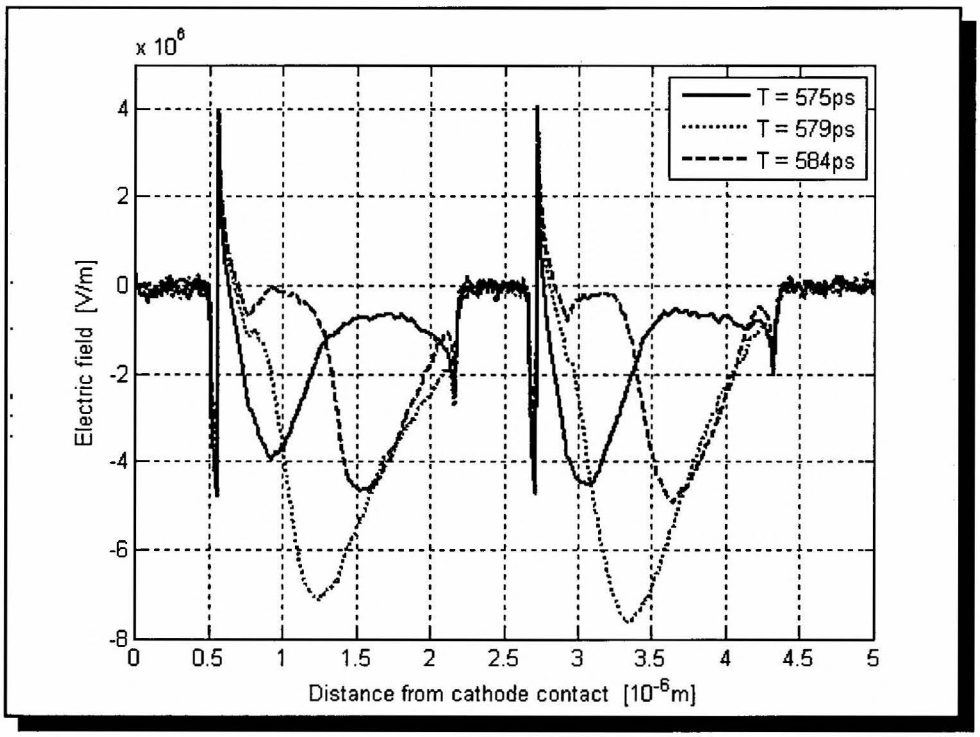


Figure 5.7: Internal electric field distribution of diode iteration #2D_04 at successive time intervals.

From these profiles it is clear that the domain growth in the two active regions is synchronised. A slightly more pronounced domain is observed in the second region. This can be attributed to the lower operating temperature in this region (see Figure 5.8). It is also noted that the choice of buffer (0.5µm wide and doped at $2\times10^{23}m^{-3}$) is adequate for complete domain quenching between the two regions.

The grading of the doping over the last 25% of the active regions evidently has the desired effect of reducing the high electric field values on the anode sides of the regions, which are characteristic of hot-launcher diodes. This subsequently reduces excessive heat generation in these regions.

- Temperature profile

The temperature profile throughout the active regions is shown in Figure 5.8.

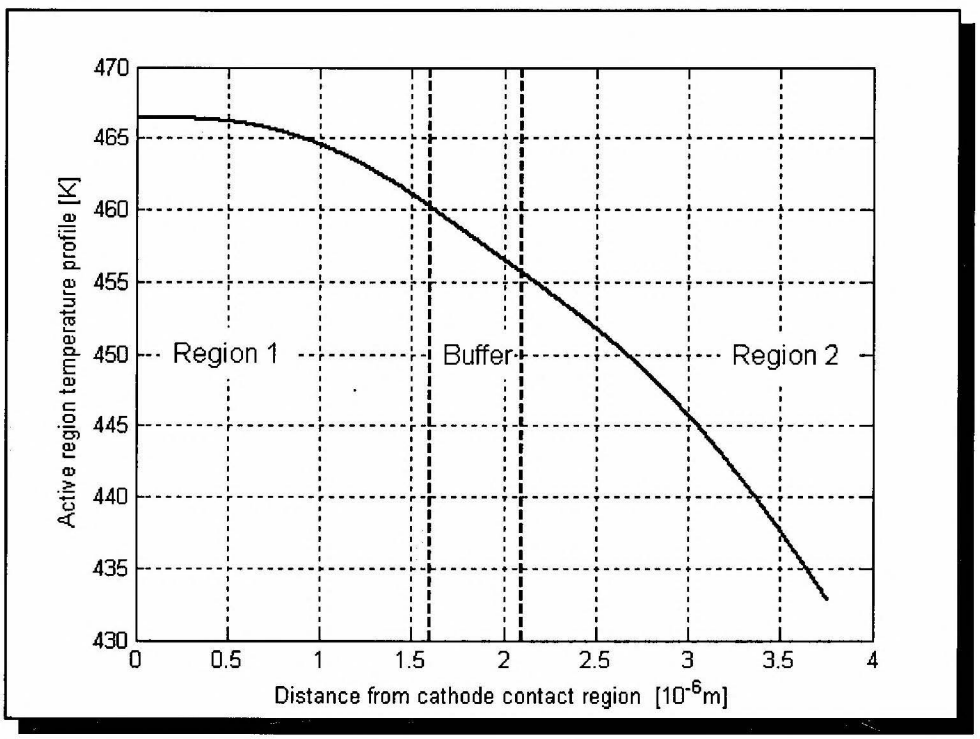


Figure 5.8: Temperature profile throughout the active regions of diode iteration #2D_04.

The temperature drop over the two regions combined is ≈35K and the average temperatures in Regions 1 and 2 are 463K and 445K respectively. This suggests that a constant operating temperature throughout the device should not be assumed.

The temperature difference of only 5K across the buffer should be noted. Choosing a narrower buffer would therefore not have a significant impact, other than running the risk of it being too narrow for adequate domain quenching.

5.3.5 Diode iteration #2D_05

In conclusion, the effect of grading the active regions' doping profiles on diode performance is investigated. In the foregoing iterations a grading over the last 25% of each active region, increasing towards the anode, was assumed. Grading will now be applied linearly over the full active region. The doping notch is still included.

Three cases are investigated where the doping density is (1) *decreased* by a factor 2 from cathode to anode, (2) *held constant* over the whole active region and (3) *increased* by a factor two from cathode to anode. For the sake of comparison, a constant "$N_{AR}L_{AR}$" product is assumed in all cases, the optimum corresponding with $N_{AR} = 1.5\times10^{22}\text{m}^{-3}$and $L_{AR} = 1.4\mu\text{m}$.

The results are tabulated in Table 5.7.

It is apparent that decreasing the doping density from cathode to anode has a very detrimental effect on the diode's performance and should not be considered. The most promising grading profile is that of an increased doping density towards the anode. Output power similar to a constant grading profile is assumed, but at enhanced efficiencies.

Thermal considerations could be attributed to these results, as is evident from the decreased operating temperature with increased doping density towards the anode. This will inevitably lead to increased efficiency. However, the *combined* effect of the temperature and doping profiles on the resistivity profile throughout the active regions (to ensure *uniform* resistivity) should also

be considered in explaining the enhanced efficiency.

Comparing the results with that of the optimum diode (Table 5.6), it seems that increasing the doping density over only the last 25% of the active region yields even higher efficiencies and output power. This is counter-intuitive when considering that the optimum diode operates at a 10K higher operating temperature. It is unclear whether the increased performance is attributable to a, supposedly, more uniform resistivity profile, or merely because of the marginally higher "$n_0 l$" product in the case of the optimum diode. Further investigation into this phenomenon is warranted.

TABLE 5.7

MICROWAVE PERFORMANCE OF DOUBLE-DOMAIN DIODE ITERATION #2D_05 FOR VARIOUS ACTIVE REGION DOPING DENSITY GRADING PROFILES

Active region doping density N_{AR} (cathode side)	[10^{22}m^{-3}]	2	1.5	1
Active region doping density N_{AR} (anode side)	[10^{22}m^{-3}]	1	1.5	2
Bias voltage V_{DC}	[V]	6	6	6
Fundamental harmonic V_1	[V]	3	4.5	5.25
Second harmonic V_2	[V]	0.9	1.4	1.3
Phase difference ϕ	[°]	320	320	320
DC current	[A]	1.43	1.38	1.27
DC input power	[W]	8.8	8.4	7.8
Output power at 94GHz	[mW]	**91**	**136**	**137**
RF Efficiency	[%]	**1.0**	**1.6**	**1.8**
Diode admittance	[S]	-0.25 + j0.28	-0.16 + j0.29	-0.18 + j0.28
Cathode contact temperature	[K]	470	465	455

The time-averaged electric field throughout the device is shown in Figure 5.9. It is clear from these profiles that a decreasing doping profile is associated with an increased dead zone and very high electric field at the anode side of each active region. This leads to enhanced heating, away

from the heat sink in the case of the first active region.

In the case of an increased doping density, the domain formation reaches maturity and very little potential is "lost" on the anode side of the active regions. This also leads to lower bias currents and reduced heating and, seen in its totality, will increase efficiency.

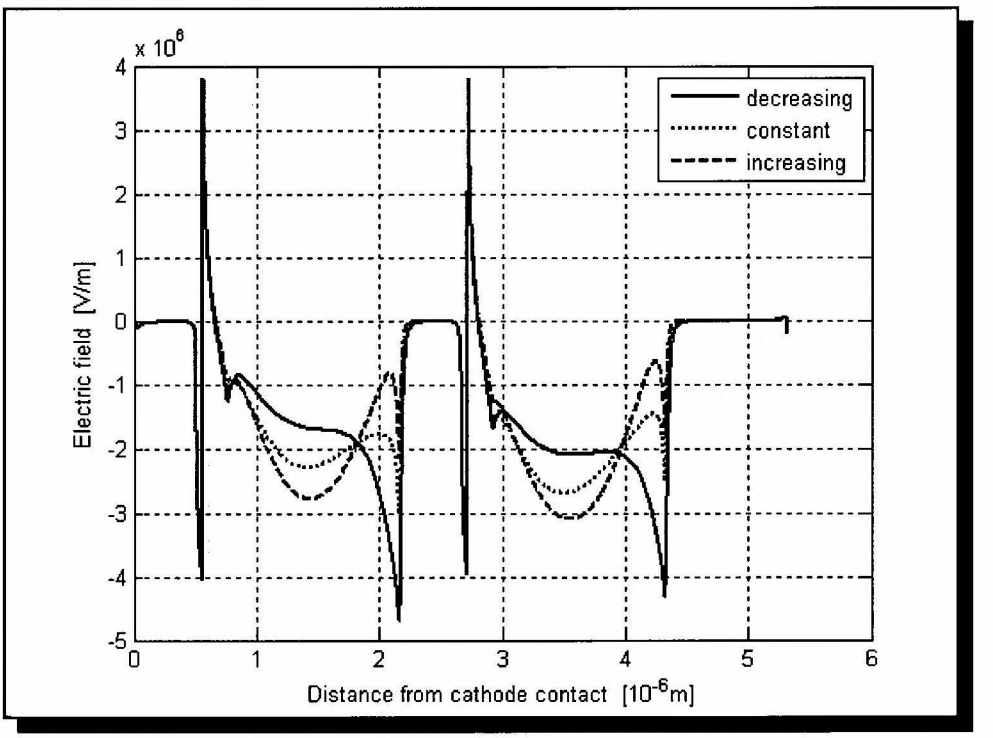


Figure 5.9: Time-averaged internal electric field distribution for the three cases of (1) decreasing doping density from cathode to anode, (2) constant doping density and (3) increased doping density from cathode to anode.

5.3.6 Yield Analysis

Manufacturing inaccuracies necessitate a yield analysis of the optimised double-domain diode. The analysis has been limited to an investigation of the effect that variances between the active regions' nominal doping densities and lengths have on microwave performance. A proper yield analysis, encompassing many more parameters will be exhaustive and impractical because of the duration of each Monte Carlo simulation. It is believed, however, that these two analyses address the most critical situations.

5.3.6.1 Variance in active region doping density

Three cases are considered: the nominal doping density (1) in the first active region exceeds that of the second region by 10%, (2) in the first active region matches that of the second region and (3) in the second region exceeds that of the first region by 10%. In all cases it is assumed that the active regions' lengths are equal.

The results are tabulated in Table 5.8.

TABLE 5.8

MICROWAVE PERFORMANCE OF OPTIMISED DOUBLE-DOMAIN DIODE (ITERATION #2D_04) FOR VARIOUS VARIANCES IN ACTIVE REGION DOPING DENSITY PROFILE

Nominal doping density in active region 1 N_{AR} (cathode contact side)	[$10^{22}m^{-3}$]	1.575	1.5	1.425
Nominal doping density in active region 1 N_{AR} (Anode contact side)	[$10^{22}m^{-3}$]	1.425	1.5	1.575
Doping variance (relative to nominal doping of $1.5\times10^{22}m^{-3}$)	%	10	0	-10
Bias voltage V_{DC}	[V]	6	6	6
Fundamental harmonic V_1	[V]	4.9	4.75	4.75
Second harmonic V_2	[V]	0.9	1.3	1.05
Phase difference ϕ	[°]	320	320	320
DC current	[A]	1.33	1.37	1.34
DC input power	[W]	8.2	8.4	8.2
Output power at 94GHz	[mW]	**118**	**161**	**129**
RF Efficiency	[%]	**1.4**	**1.9**	**1.6**
Diode admittance	[S]	-0.33 + j0.27	-0.21 + j0.27	-0.26 + j0.27
Cathode contact temperature	[K]	460	465	470

The time-averaged electric field profiles for each of these cases are shown in Figure 5.10 for comparison.

From Table 5.8 it is evident that mismatches between the active regions' doping profiles have detrimental effects on diode performance. The marginally better performance of the third case, compared to the second, is explained in Figure 5.10. The optimum diode exhibits a slightly larger domain formation in the second region, at the expense of the first. By increasing the second region's doping density this effect is countered, and not exacerbated as with the first case.

Diode admittance is not adversely affected though, and favourable impedance levels are still presented to the external circuit.

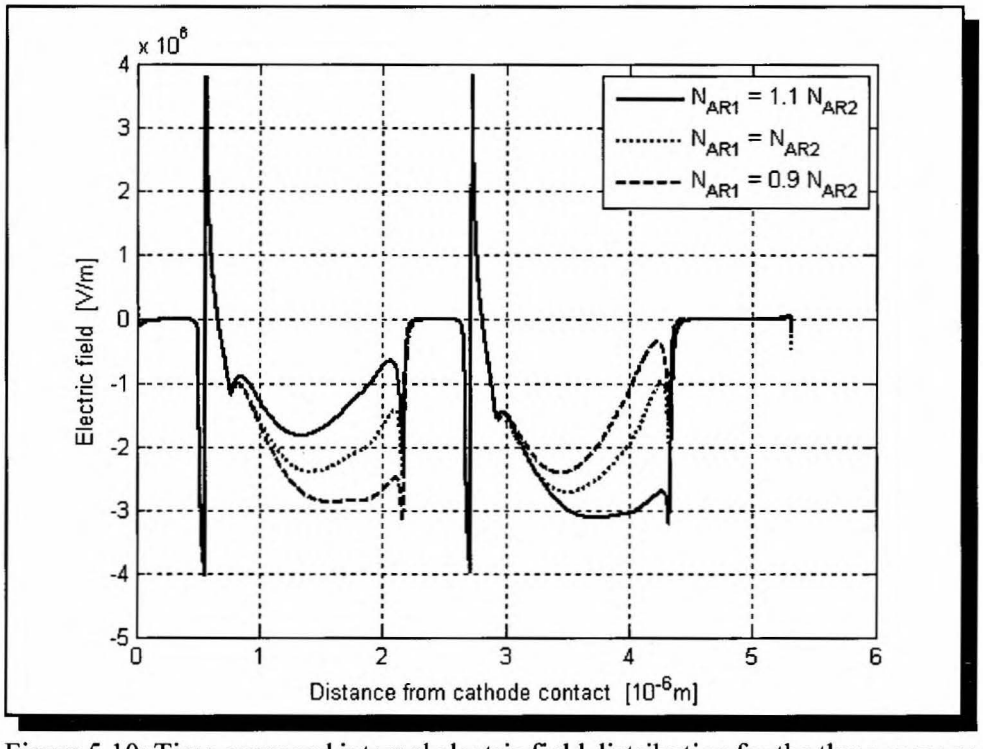


Figure 5.10: Time-averaged internal electric field distribution for the three cases as mentioned in the text.

5.3.6.2 Variance in active region length

Three cases are considered: the nominal active region length (1) of the first active region exceeds that of the second region by 10%, (2) of the first active region matches that of the second region and (3) of the second region exceeds that of the first region by 10%. In all cases it is assumed that the nominal doping densities in both active regions are equal.

The results are tabulated in Table 5.9.

TABLE 5.9

MICROWAVE PERFORMANCE OF OPTIMISED DOUBLE-DOMAIN DIODE (ITERATION #2D_04) FOR VARIOUS VARIANCES IN ACTIVE REGION LENGTH

Active region 1 length L_{AR} (cathode contact side) excluding 0.2μm doping notch	[μm]	1.47	1.4	1.33
Active region 2 length L_{AR} (Anode contact side) excluding 0.2μm doping notch	[μm]	1.33	1.4	1.47
Variance in active region length (relative to nominal length of 1.4μm)	%	10	0	-10
Bias voltage V_{DC}	[V]	6	6	6
Fundamental harmonic V_1	[V]	4.75	4.75	4.75
Second harmonic V_2	[V]	1.1	1.3	1.2
Phase difference ϕ	[°]	320	320	320
DC current	[A]	1.36	1.37	1.41
DC input power	[W]	8.4	8.4	8.7
Output power at 94GHz	[mW]	**136**	**161**	**154**
RF Efficiency	[%]	**1.6**	**1.9**	**1.8**
Diode admittance	[S]	-0.26 + j0.3	-0.21 + j0.27	-0.24 + j0.24
Cathode contact temperature	[K]	470	465	470

These results indicate the detrimental effect of variances in active region length on output performance. It would also seem, though, that this is less averse than the case of variances in doping density.

Diode admittance is not adversely affected though, and favourable impedance levels are still presented to the external circuit.

5.4 CONCLUSION

This chapter dealt with the optimisation of double-domain Gunn diodes. The optimisation strategy can be summarised as follows:

- Establishing a single-domain diode for benchmarking where a copper heat sink is assumed,
- Duplicating the single-domain diode into a double-domain diode,
- Finding optimum doping levels for the optimised diode,
- Incorporating a diamond heat sink for improved thermal performance and finding optimum doping levels for the optimised diode and
- Performing yield analyses on the optimised diode.

The results of each optimisation step have been discussed and related to internal electric field and temperature profiles where applicable.

A diode with a maximum output power of ≈160mW at ≈2% efficiency has been established. Yield analyses show that, taking manufacturing uncertainties into account, an output level of greater than ≈120mW can be expected. This is well above the desired 100mW.

Chapter 6

CONCLUSION

6.1 INTRODUCTION

In conclusion, the main achievements of the work presented here will be discussed briefly. These are:

- An optimised multi-domain Gunn diode, with multiple hot-electron launchers;
- MC-PVM: A parallel implementation of the Monte Carlo particle simulation algorithm;
- A computationally more efficient leap-frog implementation of MC-PVM; and
- The incorporation of a thermal model into the algorithm.

Extensive verification of the model has been performed throughout to ensure acceptable correlation with real-world scenarios. Model verification comprised the following scenarios:

- Bulk material simulation of GaAs;
- DC simulations and thermal analyses of a mm-wave diode based on the work of Batchelor [32]; and
- AC simulations of a mm-wave diode based on the work of Couch *et al* [26], [27].

These have been achieved beyond doubt.

6.2 OPTIMISED GUNN DIODE AT 94 GHz

A multi-domain Gunn diode with multiple hot-injection launchers is proposed and found to yield superior output performance. An optimised double-domain diode has been established which incorporates hot-electron launchers, doping notches and grading of the active layer doping profile. This "cocktail" approach renders output powers in the region of 160mW at 2% efficiency. Commercially available diodes of two prominent manufacturers, MDT [17] and E2V Technologies [18], yield in the order of 80-90mW. The output power of the suggested diode is therefore well above state-of-the-art diodes currently available.

The author has presented initial work on the strategy of incorporating multiple hot-electron

launchers at two international conferences, namely ICSDT 1998 [44] and Africon 1999 [45].

It is the opinion of the author that the incorporation of more than two active layers will not result in appreciably more output power, if any, due to enhanced thermal losses.

Although it has not been investigated, the diode should benefit from the same advantages of a single domain hot-electron launcher diode. These advantages are less sensitivity to temperature and bias variations, improved turn-on characteristics and noise performance.

A yield analysis of the diode shows that the device's operation is sensitive to variances in doping profiles and lengths between the active layers. These variances should be kept to a minimum. For a 10% difference between the doping profile of the first and second layers, an output power of about 120mW can be expected. For the same variation in the lengths of the active layers, an output of about 140mW can be expected. In both instances, this is still well above the 100mW target set for the present work.

6.3 MC-PVM algorithm

A cost-effective parallel implementation of the Monte Carlo algorithm on a network of personal computers has been established. The algorithm achieved a speed-up factor of 13 on a cluster with 19 slave nodes.

The clustered implementation of the Monte Carlo algorithm should place it within reach of smaller research centers and universities that do not have the resources to acquire more expensive supercomputers.

6.4 Leap-frog MC-PVM algorithm

A novel leap-frog version has also been implemented, where the idling time of the slave nodes has been reduced. The leap-frog implementation achieves 385ps simulation time in a real-time

hour, and realised a remarkable speed-up factor of 17 on a cluster of 19 slave nodes. Without the leap-frog MC-PVM algorithm this work would not have been possible.

The author first presented this work in 1996 in an international journal [9], and subsequent development of the algorithm at international conferences in 1998 [10] and 2004 [11].

6.5 THERMAL MODEL

A thermal model, where the temperature profile is determined consistently with the time-evolution of the terminal current and electric field profiles, has been successfully incorporated into the MC simulator. Temperature is determined with fine grid-resolution throughout the device and not assumed constant. This enables us to investigate the influence of graded doping profiles on device performance in more detail, and renders a more realistic model of high temperature Gunn diodes.

To the knowledge of the author, this has never been implemented elsewhere.

REFERENCES

[1] F.F. Mundell, "Development of a high yield fabrication process for millimeter wave Gunn diodes", M. Sc. Thesis, University of Natal, Natal, South Africa, December 1998.

[2] C. Moglestue, *Monte Carlo simulation of semiconductor devices*, Chapman & Hall, London, 1993.

[3], C. Jacoboni and P. Lugli, *The Monte Carlo method for semiconductor device simulation*, Springer-Verlag, New York, 1989.

[4] P. Lugli, "The Monte Carlo method for semiconductor device and process modeling", *IEEE Trans. Computer-aided Design*, vol. 9, no. 10, pp. 1164-1176, 1990.

[5] K. Tomizawa, *Numerical simulation of submicron semiconductor devices*, Artech House, Boston, 1993.

[6] W. Fawcett, A.D. Boardman and S. Swain, "Monte Carlo determination of electron transport properties in Gallium Arsenide", *J. Phys. Chem. Solids*, vol. 31, pp. 1963 - 1990, 1970.

[7] R.R van Zyl, G. Conradie and W.J. Perold, "An introduction to the Monte Carlo particle simulation tchnique", presented at the IEEE AP/MTTS-94 Conference, Stellenbosch University, Stellenbosch, pp. 79 - 84, 1994.

[8] R.R van Zyl, W.J. Perold and G. Conradie, "The application of the Monte Carlo method to semiconductor simulation", *Trans. SAIEE*, vol. 87, pp.58 - 64, 1996.

[9] R.R van Zyl, W.J Perold and H. Grobler, "An efficient parallel implementation of the Monte Carlo particle simulation technique on a network of personal computers", *Intl. Jnl.*

Numercial Modelling: Electronics Networks, Devices and Fields, John Wiley and Sons, vol. 13, pp. 369 -380, 1996.

[10] R.R. van Zyl, W.J. Perold and H. Grobler, "Parallel implementation of the Monte Carlo particle simulation technique on a network of personal computers", presented at the 6th International Conference on Simulation of Devices and Technologies (ICSDT '98), Cape Town, pp. 81-86, 2000.

[11] L.P. Wicomb and R.R van Zyl, "A budget supercomputer: implementation and application", presented at IEEE AFRICON 2004, Gaborone (Botswana), pp. 15-17, Sept. 2004.

[12] *The American Heritage® Dictionary of the English Language*, American Heritage Publishing Company, 4th Ed., 2000.

[13] J.B. Gunn, "Microwave Oscillations of Current in Group III-V Semiconductors", *Solid State Consm.*, vol. 1, pp. 88 - 91, Sept. 1963.

[14] B.K. Ridley and T.B Watkins, "The possibility of negative resistance effects in semiconductors", *Proc. Phys. Soc.*, vol. 78, pp. 293-304, 1961.

[15] C. Hilsum, "Transferred electron amplifiers and oscillators", *Proc. IRE*, vol. 50, pp. 185-189, Feb. 1962.

[16] J. Voelcker, "The Gunn effect", *IEEE Spectrum*, p. 24, July 1989.

[17] MDT Catalog: "Gunn diodes". [Online]. Available: http://www.mdtcorp.com/gunnosc.html

[18] E2V Technologies: "Gunn diode application notes", *A1A-Gunn Diodes AN1*, Issue 4, September 2002. [Online]. Available: http://e2v.com/applications/diodes.htm

[19] S.J. Teng and R.E Goldwasser, "High-performance second-harmonic operation W-band GaAs Gunn diodes", *IEEE ED Letters*, vol. 10, No. 9, pp. 412-414, September 1989.

[20] T.G Rutten, "Gunn diode oscillator at 95GHz", *Electron. Lett.*, vol. 11, pp. 293-294, 1975.

[21] H. Barth, "A wideband backshort-tunable second harmonic W-band Gunn oscillator", *Proc. IEEE-MTT-S Symp.*, 1981, pp. 825-826.

[22] W.H. Haydl, "Fundamental and harmonic operation of millimeter-wave Gunn diodes", *IEEE Trans. MTT.*, vol. MTT-31, 11, pp. 879-889, 1983.

[23] Z. Greenwald, "The effect of a high energy electron injection cathode on the performance of the Gunn oscillator", Ph.D. dissertation, Cornell University, New York, August 1986.

[24] Z Greenwald *et al*, "The effect of a high energy injection on the performance of mm wave Gunn Oscillators", *Solid-State Electronics*, vol. 31, no. 7, pp. 1211-1214, 1988.

[25] S. Neylon et al, "State-of-the-art performance millimetre wave Gallium Arsenide Gunn diodes using ballistically hot electron injectors", *IEEE MTT-S Digest*, L-47, pp. 519-522, 1989.

[26] H. Spooner and N.R. Couch, "Advances in hot electron injector Gunn diodes", *GEC Jnl. Research*, vol. 7, no. 1, pp. 34-45, 1989.

[27] N.R Couch et al, "High-performance, graded AlGaAs injector, GaAs Gunn diodes at 94GHz", *IEEE ED Lett.*, vol. 10, no. 7, pp. 288-290, July 1989.

[28] P.H. Beton *et al*, "Use of n^+ spike doping regions as nonequilibrium connectors", *Electronic Letters*, vol, 24, no. 7, pp. 434-435, March 1988.

[29] R. Vaidyanathan and R.P Joshi, "Simulations for improved heterostructure Gunn oscillator based on transit region doping variation", *Electronic Letters*, vol. 27, no. 17, pp. 1555-1557, August 1991.

[30] J.W. Tully, "Monte Carlo simulation of a millimeter-wave Gunn-effect relaxation oscillator", *IEEE Trans. ED.*, vol. ED-30, no. 6, June 1983.

[31] C. Lee and U. Ravaioli, "Monte Carlo comparison of heterojunction cathode Gunn oscillators", *Electron. Lett.*, vol. 26, no.7, March 1990.

[32] A.R Batchelor, "Thermal design considerations for GaAs transferred-electron devices", PhD. Dissertation, University of Leeds, Leeds, 1990.

[33] A.R Batchelor, "On the design of doping concentration profiles for GaAs transferred-electron device layers", *Solid-State Electronics*, vol. 35, no. 5, pp. 735-741, 1992.

[34] F. Hasgawa and M. Suga, *IEEE Trans. ED*, vol. ED-19, pp. 26-37, 1972.

[35] A. Paolella, R.L Ross and J. Ondria, Microwave Journal, vol. 29, p. 149, 1986.

[36] J. Ondria and R.L Ross, presented at the Proc. 17th Eur. Microwaves Conf., Rome, Italy, p.673, 1987.

[37] J.M Szubert *et al*, "W-Band GaAs Gunn diodes with high output power", Solid-State Electronics, vol. 33, no. 8, pp. 1035-1037, 1990.

[38] H. Kroemer, *Proc. IEEE*, vol. 54, pp. 1980-1981, 1966.

[39] H.W Thim, "Series-connected bulk GaAs amplifiers and oscillators", *Proc. IEEE*, vol. 56, p. 1244, 1968.A.

[40] M. Slater and R. Harrison, "An investigation of multiple domain Gunn effect oscillators",

IEEE Trans. ED, vol. ED-23, no.6, pp. 560-567, June 1976.

[41] Talwar, "A dual-diode 73-GHz Gunn oscillator", *IEEE Trans. MTT*, vo. MTT-27, no.5, pp. 510-512, May 1979.

[42] J. Tsay *et al*, "Multidomain Gunn diodes", *Microwave and Optical Technology Letters*, vol. 3, no. 2, pp. 54-60, February 1990.

[43] Y.P. Teoh and G.M Dunn, "Monte Carlo modelling of multiple transit regions Gunn diodes", *Electronic Letters*, vol. 38, no. 15, pp. 830-831, July 2002.

[44] R.R van Zyl and W.J Perold, "Optimization of millimeter wave Gunn oscillators with the aid of the Monte Carlo simulation technique", presented at the 6th International Conference on Simulation of Devices and Technologies (ICSDT '98), Cape Town, pp. 93-96, 14-16 October 1998.

[45] R.R van Zyl, W.J Perold and R. Botha, "Multi-domain Gunn diodes with multiple hot electron launchers: a new approach to mm-wave GaAs Gunn oscillator design", presented at the IEEE Africon 1999, Cape Town, pp. 1193-1196, 1999.

[46] T. Kurosawa, "Monte Carlo simulation of hot electron problems", *J. Phys. Soc. Japan*, Suppl. vol. 21, p.527, 1966.

[47] S.M. Sze, *Semiconductor devices physics and technology*, John Wiley & Sons, New York, 1985.

[48] R.R. van Zyl, "Die implementering van die Monte Carlo partikelsimulasie algoritme op GaAs strukture", M. Sc. thesis, Stellenbosch University, Stellenbosch, December 1994.

[49] R.W Hockney and J.W. Eastwood, *Computer simulation using particles*, Institute of Physics Publishing, Philadelphia, 1992.

[50] H.A. Haus and J.R. Melcher, *Electromagnetic fields and energy*, Prentice-Hall International Editions, New Jersey, 1989.

[51] H.D Rees, "Calculation of steady state distribution function by exploiting stability", *Phys. Lett.*, 26A, pp. 416-7, 1968.

[52] T. González and D. Pardo, "Physical models of ohmic contact for Monte Carlo device simulation", *Solid-State Electronics*, vol. 39, no. 4, pp. 555-562, 1996.

[53] C.M Wolfe, N. Holonyak and G.E. Stillman, *Physical properties of semiconductors*, Prentice-Hall International Editions, New-Jersey, 1989.

[54] J.R Chelikowsky and M.L. Cohen, "Nonlocal pseudopotential calculations for the electronic structure of eleven diamond and zinc-blend semiconductors", *Physical Review B*, vol. 14, no. 2, pp. 556-582, 1976.

[55] K. Tomizawa *et al*, "Monte Carlo simulation of submicron GaAs n^+-i(n)-n^+ diode", *IEE. Proc.*, vol. 129, pt. I, no. 4, August 1982.

[56] P.C.W Davies and D.S. Betts, *Quantum Mechanics*, 2nd Edition, Chapman and Hall, London, 1995.

[57] G.C Osbourn and D.L. Smith, "Carrier transport coefficients across GaAs-GaAlAs (100) interfaces", *J. Vac. Sci. Technol.*, vol. 16, no. 5, pp. 1529-1532, 1979.

[58] K. Kim and K. Hess, "Ensemble Monte Carlo simulation of semiclassical nonlinear electron transport across heterojunction band discontinuities", *Solid-State Electronics*, vol. 31, no. 5, pp. 877-885, 1988.

[59] C.M Wu and E.S. Yang, "Carrier transport across heterojunction interfaces", *Solid-State*

Electronics, vol. 22, pp. 241-248, 1979.

[60] H.S. Carslaw and J.C. Jaeger, *Conduction of heat in solids*, Oxford University Press, Oxfrod, 1950.

[61] M.F. Zybura *et al*, "Efficient computer aided design of GaAs and InP millimeter wave transferred electron devices including detailed thermal analysis, *Solid-State Electronics*, vol. 38, no. 4, pp. 873-880, 1995.

[62] S. Ramo, J.R Whinnery and T. van Duzer, *Fields and waves in communication electronics*, 2nd ed., John Wiley & Sons, New York, 1984.

[63] J.S Blakemore, "Semiconducting and other major properties of gallium arsenide", *J. Appl. Phys.*, vol. 53 (10), pp. R123-181, October 1982.

[64] S. Adachi, "GaAs, AlAs, and $Al_xGa_{1-x}As$: Material parameters for use in research and device applications", *J. Appl. Phys.*, vol. 58, pp. R1-29, 1985.

[65] M.A Littlejohn, J.R Hauser and T.H Glisson, "Velocity-field characteristics of GaAs with Γ^C_6-L^C_6-X^C_6 conduction-band ordering", *J. Appl. Phys.*, vol. 48, no. 11, pp. 4587-4590, 1977.

[66] M.V. Fischetti, "Monte Carlo simulation of transport in technologically significant semiconductors of the diamond and zinc-blende structures - Part I: Homogenous transport", *IEEE Trans. Electron Devices*, vol. 38, no. 3, March 1991.

[67] K.F Brennan *et al*, "Theory of velocity-field relation in AlGaAs", *J. Appl. Phys*, vol. 63 (10), May 1988.

[68] Ioffe Physico-Technical Institute, *New Semiconductor materials. Characteristics and properties*. [Online]. Available http://www.ioffe.rssi.ru/SVA/NSM/

[69] N. Braslau and P.S. Hauge, μwave measurement of the velocity-field characteristic in GaAs", *IEEE Trans. Electron Devices*, vol. ED-17, p. 616, 1970.

[70] P.A Houston and A.G.R Evans, "Electron drift velocity in n-GaAs at high electric fields", *Solid-State Electronics*, vol. 20, pp. 197-204, 1977.

[71] W.T Maselink *et al*, "Electron velocity and negative differential mobility in AlGaAs/GaAs modulation doped heterostructures", Appl. Phys. Lett., vol. 51 (19), pp. 1533-1535, 1987.

[72] S.M. Goodnick *et al*, "Parallel implementation of a Monte Carlo simulation coupled to Maxwell's equations", *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, John Wiley and Sons, vol. 8, pp. 205-219, 1995.

[73] [Online]. Available at http://www.epm.ornl.gov/pvm/

[74] [Online]. Available at http://www.beowulf.org

[75] [Online]. Available at http://www.linux.org

[76] [Online]. Available at http://www.ltsp.org

[77] [Online]. Available at http://www.netlib.org/utk/papers/mpi-book/mpi-book.html

[78] J.W. Tully, "Monte Carlo simulation of a millimeter-wave Gunn-effect relaxation oscillator", IEEE Trans. Electron Devices", vol. 30, no.6, pp. 566-571, 1983.

[79] B.G. Bosch and R.W.H Engelman, *Gunn-effect Electronics*, Pitman Publishing, New York, 1975.

[80] J.E. Caroll, *Hot electron microwave generators*, Edward Arnold Publishers, London,

1970.

[81] R.R. van Zyl and W.J. Perold, "The Gunn-diode: Fundamentals and fabrication", presented at IEEE AP/MTTS-98, University of Cape Town, Cape Town, pp. 121-124, 1998.

[82] Y.P Teoh *et al*, "Monte Carlo modelling of multiple-transit-region Gunn diodes", *Semicond. Sci. Technol.*, Vol. 17, pp. 1090-1095, 2002.

[83] H Kroemer, *Proc IEEE*, vol. 52, p 1736, 1964.

[84] E. Alekseev and D. Pavlidis, "Large-signal microwave performance on GaN-based NDR diode oscillators", *Solid-State Electronic*, vol 44, no 6, pp. 941-947, 2000.

[85] Burgemeister, E. A., *Physica*, vol. B93, pp. 165-179, 1978.

Appendix A

"C" PROGRAMME LISTING

A1 MASTER NODE PROGRAMME: SYNOPSIS

lfmaster.c

This is the master node's main programme implementation of the leap-frog MC-PVM algorithm. It includes all the header files listed below.
For code listing, see p. A8.

m_var_new.h

Material parameters and variables used in "lfmaster.c" are declared.
For code listing, see p. A10.

writefile.h

This header file contains procedures to create data output files as the simulation progresses. These files include external terminal conditions (current, voltage), as well as internal electric field, charge, satellite occupation and temperature distributions. Data is written to text files for postprocessing in Matlab.
For code listing, see p. A12.

m_init_new.h

Initialisation of programme variables and parameters are listed in this header file.
For code listing, see p. A16.

m_pvm.h

This selection of procedures governs the communication between the master node and all the slave nodes.

For code listing, see p. A20.

field_solve.h

This header file contains the one-dimensional Poisson solver, used for determining the internal electric field distribution.
For code listing, see p. A24.

thermal.h

All procedures related to the determination of the internal temperature distributions are contained in this header file.
For code listing, see p. A24.

update2.h

Procedures related to the determining of terminal conditions (current and voltage), as well as internal field temperature and field distributions, at the end of every time step are listed in this header file.
For code listing, see p. A26.

A2 SLAVE NODE PROGRAMME: SYNOPSIS

lfslave.c

This is the slave node's main programme implementation of the leap-frog MC-PVM algorithm. It includes all the header files listed below.
For code listing, see p. A31.

svt_variable.h

Material parameters and variables used in "lfslave.c" are declared.
For code listing, see p. A33.

s_pvm.h

This selection of procedures governs the communication between the slave and master nodes.
For code listing, see p. A36.

s_param.h

This header file contains functions to determine all composition and temperature dependent material parameters.
For code listing, see p. A37.

scattable_xt_new.h

The scattering rates lookup table is generated. (See section 2.2.2.4 for details on table structure.)
For code listing, see p. A42.

s_init.h

All initialising procedures (initial electronic state and position, initialisation of programme variables and parameters) are listed in this header file. Nearest Grip Point allocation of charge is assumed.
For code listing, see p. A46.

randgen.h

This file contains the procedure to generate the random numbers used in the Monte Carlo algorithm (as described in Chapter 2).
For code listing, see p. A49.

dynamix_x.h

The electron's dynamics during free flight are determined.
For code listing, see p. A50.

boundary.h

The procedure in this file checks whether an electron has traversed a contact boundary.
For code listing, see p. A52.

scattering3.h

This file contains all procedures related to electron scattering, i.e. determining scattering rate, selecting a scattering event and determining the electron's momentum state after scattering event.
For code listing, see p. A53.

transfer.h

Electron parameters (position and state vector quantities) are transferred to unoccupied positions in these vectors that were left by deleted electrons to ensure efficient use of memory.
For code listing, see p. A58.

c_assign.h

This header file contains the procedures to assign charge to the position on the grid according to the nearest-grid-point scheme as well as inserting new particles to the simulation to ensure charge neutrality near the ohmic contacts.
For code listing, see p. A58.

vav.h

The average electron velocity distribution is determined.
For code listing, see p. A60.

sat_statistics.h

This file contains a procedure to determine statistics of the electrons' satellite occupation throughout the diode.
For code listing, see p. A61.

A3 PROCEDURES COMMON TO MASTER AND SLAVE NODES: SYNOPSIS

global.h

Global simulation parameters are declared.

For code listing, see p. A62.

g_param.h

This header file contains functions to determine all composition and temperature dependent material parameters that are used by both the master and slave nodes.

For code listing, see p. A63.

mathadd.h

Additional mathematical function used by the master and slave nodes are defined.

For code listing, see p. A64.

A1 MASTER NODE PROGRAMME: CODE LISTING

lfmaster.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <string.h>
#include "pvm3.h"
#include "mathadd.h"
#include "global.h"
#include "m_var_new.h"
#include "g_param.h"
#include "writefile.h"
#include "m_init_new.h"
#include "m_pvm.h"
#include "field_solve.h"
#include "thermal.h"
#include "update2.h"

main()
{
  int i,dum,dum2,file_i = 0,iter0;
  time_t starttime,fintime;
  realtype i_dummy,temp_v[NY+1],temp_vav[NY+1],itop_av_temp=0,ibot_av_temp=0;
  initpvm_alpha();
  init();
  write_var();
  printf("\ni_ar0 = %d  i_ar1 = %d  nar = %d",i_ar0,i_ar1,n_ar);
  printf("\n%d  %d  %d  %d",i1_con0,i1_con1,i2_con0,i2_con1);
  for (i = 0; i <= NY; i++) {
    temp_vav[i] = 0;
    vav[i] = 0;
  }
  iter = 0;
  iter0 = 0;
  printf("\nNumber of slaves = %d",NUMBER_OF_SLAVES);
  printf("\nx = %f", X_MAX);
  slavetot2slaves();
  qs2slaves();
  ndope2slaves();
  readfromslaves();
  distr_field_current();
  starttime = time(NULL);
  while (simtime <= TSIM) {
    iter++;
    if (iter%2 == 0) {
          printf("\nt = %.3fps",simtime*1e12);
          readfromslaves();
          distr_field_current();
```

```
            itop_av_temp += itop;
            ibot_av_temp += ibot;
            for (i = 0; i <= NY; i++) temp_vav[i] += vav[i];
            i_dummy = iext[1];
            if (CAVITY == 0) i_dummy = itop;
            printf("   V = %.2f",vdiode[1]);
            printf("   E0 = %.2f",e_fieldy[0]/1E6);
            printf("   sM = %.2f",s_max/1e14);
            printf("   tleft = %.0f",t_distr[0]);
    }
    if (iter%2 != 0) {
            e_fieldy[NY + 1] = 0;
            write2slaves();
    }
    dum = TSIM/TSTEP/OUTPUTS1;
    dum2 = TSIM/TSTEP/OUTPUTS2;
    if (iter%dum == 0) {
            file_i++;
            tofile("e",file_i,e_fieldy);
            tofile("e_av",file_i,efield_av);
            tofile("v_acc",file_i,v_acc_av);
            tofile("c",file_i,c_distr);
            write_qi("qi",file_i);
            write_t("t_distr",file_i);
            write_totals();
            for (i = 0; i <= NY; i++)
              if (c_distr[i] == 0) c_distr[i] = 1;
            for (i = 0; i <= NY; i++)
              temp_v[i] = valley_info[0][i]/c_distr[i];
            tofile("csat",file_i,temp_v);
            for (i = 0; i <= NY; i++)
              temp_v[i] = valley_info[1][i]/c_distr[i];
            tofile("lsat",file_i,temp_v);
            for (i = 0; i <= NY; i++)
              temp_v[i] = valley_info[2][i]/c_distr[i];
            tofile("xsat",file_i,temp_v);
            tofile("v",file_i,vav);
            tleft_write();
            ths_write();
    }
    if (iter%dum2 == 0) {
            for (i = 0; i <= NY; i++) vav_av[i] = 2*temp_vav[i]/(float)(iter-iter0);
            ibot_av = 2*ibot_av_temp/(float)(iter-iter0);
            itop_av = 2*itop_av_temp/(float)(iter-iter0);
            tofile("vav",file_i,vav_av);
            ip_write();
            iv_write();
            iter0 = iter;
            itop_av_temp = 0;
            ibot_av_temp = 0;
            for (i = 0; i <= NY; i++) temp_vav[i] = 0;
    }
    simtime += TSTEP/2;
}
fintime = time(NULL);
printf("\nTtot = %d",fintime - starttime);
```

```
  e_fieldy[NY + 1] = 1;  //terminate slaves
  write2slaves();
  pvm_exit();
  return 0;
}
```

m_var_new.h

```
#define PDD_AV_FACTOR 10
#define NDOMAIN 2
#define NT 100

const int
  N = 10,
  NVAV = 0,
  I_AV_FACTOR = 100,
  VAV_FACTOR = 30,
  OUTPUTS1 = 500,
  OUTPUTS2 = 2500,
  NEUMANN_HOSTS = 0,
  CAVITY = 1,  // = 1 for cavity, = 0 for constant Vdiode=VBIAS
  T_UPDATE = 1;  // = 1 for update, = 0 for constant temperature = TAMB

const double
  PI = 3.14159,
  VO = 0.0,
  VBIAS = 8,
  R_DC = 0.0,

  VB0 = 8,
  V1_0 = 5, V2_0 = 1, PHI0 = 320,

  VB1 = 8,   VB2 = 8,   VB3 = 8,
  V1_1 = 4.5,  V1_2 = 4.5,  V1_3 = 4.5,
  V2_1 = 1,  V2_2 = 1.2,  V2_3 = 1.4,
  PHI1 = 320, PHI2 = 320.0, PHI3 = 320.0,

  VB4 = 8,   VB5 = 8,   VB6 = 8,
  V1_4 = 5,  V1_5 = 5,  V1_6 = 5,
  V2_4 = 1,  V2_5 = 1.2,  V2_6 = 1.4,
  PHI4 = 320.0, PHI5 = 320.0, PHI6 = 320.0,

  VB7 = 8,   VB8 = 8,   VB9 = 8,
  V1_7 = 5.5,  V1_8 = 5.5,  V1_9 = 5.5,
  V2_7 = 1,  V2_8 = 1.2,  V2_9 = 1.4,
  PHI7 = 320.0, PHI8 = 320.0, PHI9 = 320.0,

  T0 = 0E-12,
  T1 = 50E-12,
  T2 = 150E-12,
  T3 = 250E-12,
  T4 = 350E-12,
  T5 = 450E-12,
```

```
 T6 = 550E-12,
 T7 = 650E-12,
 T8 = 750E-12,
 T9 = 850E-12,
 FREQ = 47E9,
 Q = -1.60218E-19,

 CCAP = 0.8E-12,
 LIND = 1.6E-12,
 RLOAD = 25,
 C_THERMAL = 55,
 KIHS = 319,  //Au
 L_IHS = 15e-6,
 KHS = 1250,  //diamond
 NR = 10/0.1e-6;

typedef double realtype;

typedef    char boolean;

int
 charge_ex1t,charge_ex2t,iter,eleccount,ensemble_tot,msgtype,mtid,stid,
 iv,total[20],accu,arrived,COUNTE_M,NUMBER_OF_SLAVES,stidv[20],i_th,i_v,i_ar0,i_ar1,n_ar,
 i1_con0,i1_con1,n1_con,i2_con0,i2_con1,n2_con,ip_count;

realtype
 v_source,idisp_av,idisp_ar,idisp_0,idisp_NY,ipartav,ipart,ipart_con1,ipart_con2,
 ipart_con_av,s_max,p1,p2,qs,doping,ey,simtime,ibot,itop,itop_av,
 ibot_av,old_bot_field,old_top_field,charge[NY + 10],e_fieldy[NY + 10],
 vdiode[2],idiode[2],old_poisson_field[NY+10],e_pred_field[NY+10],iext[2],hy,HY,vav[NY+1],vav_sum[NY+1],
 c_sum[NY+1],c_av[NY+1],c_distr[NY+1],vav_av[NY+1],doping_profile[NY+1],dielec[NY+1],net_charge[NY+1],
 jp_av,jp_sum,efield_sum[NY+1],efield_av[NY+1],j_particle[NY+1],k_t[NY+NY+4],a_t[NT+NT+4],c_t[NT+NT+4],
 d_t[NT+NT+4],w_t[NT+NT+4],thermal_y[NT+NT+4],ths,pddi_av[NY+1],pddi_sum[NY+1],v_acc_av[NY+1],
 thermal_t[NT+NT+4],t_distr[NY+1],pddi_v[NY+1],qi[NT+NT+4],p_param[4][NY+1],
 state_info_m[6][2][NY+1],valley_info[3][NY+1],e_exceed,ptest[NY+1],lhs[NY+1],rhs[NY+1],m_exceed,
 t_inc[11],VB[10],V1[10],V2[10],LAG[10],ip_dc_sum,ip_dc;

FILE *chandle1,*chandle2,*chandle3,*chandle4,*chandle5,*chandle6,*chandle7,*chandle8,*chandle9,
 *chandle10,*chandle11,*chandle12,*chandle13,*chandle14,*chandle15;
```

writefile.h

```
void write_var()
{
  int i;
  FILE *handle;

  handle = fopen("variable.txt","w+t");
  fprintf(handle,"%f",1E12*TSIM);
  fprintf(handle,"\n%f",1E15*TSTEP);
  for (i = 1; i <= 10; i++)
```

```
    fprintf(handle,"\n%f",1E12*t_inc[i]);
  for (i = 1; i <= 9; i++)
    fprintf(handle,"\n%f",VB[i]);
  for (i = 1; i <= 9; i++)
    fprintf(handle,"\n%f",V1[i]);
  for (i = 1; i <= 9; i++)
    fprintf(handle,"\n%f",V2[i]);
  for (i = 1; i <= 9; i++)
    fprintf(handle,"\n%f",LAG[i]);
  fprintf(handle,"\n%f",FREQ/1e9);
  fprintf(handle,"\n%d",OUTPUTS1);
  fprintf(handle,"\n%d",OUTPUTS2);
  fclose(handle);
}

//writes global array of field to disk
void tofile(char fileid[100], int ii, realtype vector[NY+1])
{
        int i;
        FILE *handle;
        char asci;
        char name[10],n1[10],n10[10],n100[10];
        asci= ii + 48 - (ii/10)*10;
        n1[0] = asci;
        n10[0] = ii/10 + 48 - (ii/100)*10;
        n100[0] = ii/100 + 48;
        strcpy(name,fileid);
        if (ii >= 100) strncat(name,n100,1);
        if (ii >= 10) strncat(name,n10,1);
        strncat(name,n1,1);
        strcat(name,".txt");
        handle = fopen(name,"w+t");
        fprintf(handle,"%.3f",simtime*1e12);
        for (i = 0; i <= NY; i++)
                fprintf(handle,"\n%f ",vector[i]);
        fclose(handle);
}


void write_qi(char fileid[100], int ii)
{
  int i;
  FILE *handle;
  char asci;
  char name[10],n1[10],n10[10],n100[10];
  asci= ii + 48 - (ii/10)*10;
  n1[0] = asci;
  n10[0] = ii/10 + 48 - (ii/100)*10;
  n100[0] = ii/100 + 48;
  strcpy(name,fileid);
  if (ii >= 100) strncat(name,n100,1);
  if (ii >= 10) strncat(name,n10,1);
  strncat(name,n1,1);
  strcat(name,".txt");
  handle = fopen(name,"w+t");
  fprintf(handle,"%.3f",simtime*1e12);
```

```
  for (i = 0; i <= NY; i++)
    fprintf(handle,"\n%f ",pddi_v[i]);
  fclose(handle);
}


void write_t(char fileid[100], int ii)
{
  int i;
  FILE *handle;
  char asci;
  char name[20],n1[20],n10[20],n100[20];
  asci= ii + 48 - (ii/10)*10;
  n1[0] = asci;
  n10[0] = ii/10 + 48 - (ii/100)*10;
  n100[0] = ii/100 + 48;
  strcpy(name,fileid);
  if (ii >= 100) strncat(name,n100,1);
  if (ii >= 10) strncat(name,n10,1);
  strncat(name,n1,1);
  strcat(name,".txt");
  handle = fopen(name,"w+t");
  fprintf(handle,"%.3f",simtime*1e12);
  for (i = 0; i <= NY; i++)
    fprintf(handle,"\n%f ",t_distr[i]);
  fclose(handle);
}

void write_totals() {
  int i;
  chandle1 = fopen("totals.txt","a+w+t");
  for (i = 1; i <= NUMBER_OF_SLAVES; i++) {
    fprintf(chandle1,"%d  ",total[i]);
  }
  fprintf(chandle1,"\n");
  fclose(chandle1);
}

void iv_write() {
  realtype temp_i;
  if (CAVITY == 1)
    temp_i = iext[1];
  if (CAVITY == 0)
    temp_i = idiode[0];
  chandle1 = fopen("iv.txt","a+w+t");
  fprintf(chandle1,"\n%.3f  %f  %f",simtime*1e12,temp_i,vdiode[1]);
  fclose(chandle1);
}

void ip_write() {
  realtype temp_i;
  chandle4 = fopen("iptop.txt","a+w+t");
  chandle5 = fopen("ipbot.txt","a+w+t");
  chandle6 = fopen("ipart.txt","a+w+t");
  chandle7 = fopen("ipart_av.txt","a+w+t");
  chandle8 = fopen("idisp_av.txt","a+w+t");
```

```
  chandle9 = fopen("ip_dc.txt","a+w+t");
  chandle10 = fopen("idisp_0.txt","a+w+t");
  chandle11 = fopen("idisp_NY.txt","a+w+t");
  chandle12 = fopen("idisp_ar.txt","a+w+t");
  chandle13 = fopen("ipart_con1.txt","a+w+t");
  chandle14 = fopen("ipart_con2.txt","a+w+t");
  chandle15 = fopen("ipart_con_av.txt","a+w+t");
  fprintf(chandle4,"\n%.3f   %f",simtime*1e12,itop);
  fprintf(chandle5,"\n%.3f   %f",simtime*1e12,ibot);
  fprintf(chandle6,"\n%.3f   %f",simtime*1e12,ipart);
  fprintf(chandle7,"\n%.3f   %f",simtime*1e12,ipartav);
  fprintf(chandle8,"\n%.3f   %f",simtime*1e12,idisp_av);
  fprintf(chandle9,"\n%.3f   %f",simtime*1e12,ip_dc);
  fprintf(chandle10,"\n%.3f   %f",simtime*1e12,idisp_0);
  fprintf(chandle11,"\n%.3f   %f",simtime*1e12,idisp_NY);
  fprintf(chandle12,"\n%.3f   %f",simtime*1e12,idisp_ar);
  fprintf(chandle13,"\n%3f    %f",simtime*1e12,ipart_con1);
  fprintf(chandle14,"\n%3f    %f",simtime*1e12,ipart_con2);
  fprintf(chandle15,"\n%3f    %f",simtime*1e12,ipart_con_av);
  fclose(chandle4);
  fclose(chandle5);
  fclose(chandle6);
  fclose(chandle7);
  fclose(chandle8);
  fclose(chandle9);
  fclose(chandle10);
  fclose(chandle11);
  fclose(chandle12);
  fclose(chandle13);
  fclose(chandle14);
  fclose(chandle15);
}


void tleft_write() {
  chandle1 = fopen("tleft.txt","a+w+t");
  fprintf(chandle1,"\n%.3f   %f",simtime*1e12,t_distr[0]);
  fclose(chandle1);
}

void ths_write() {
  chandle1 = fopen("ths.txt","a+w+t");
  fprintf(chandle1,"\n%.3f   %f",simtime*1e12,ths);
  fclose(chandle1);
}

void iv_write_() {
  chandle2 = fopen("current.txt","a+w+t");
  chandle1 = fopen("voltage.txt","a+w+t");
  fprintf(chandle1,"\n%d   %f",iter,vdiode[1]);
  fprintf(chandle2,"\n%d   %f",iter,iext[1]);
  fclose(chandle1);
  fclose(chandle2);
}
```

m_init_new.h

```
void r_init_() {
        int i,j=0,ii,iii,particles_per_gridpoint;

        for (ii = 1; ii <= (NY-1); ii++) {
                particles_per_gridpoint = Q*doping_profile[ii]*LX*LZ*LY/NY/qs;
                //particles_per_gridpoint = Q*doping_profile[ii]*LX*LZ*LY/NY/qs/NUMBER_OF_SLAVES;
                printf("%d  ",particles_per_gridpoint);
                for (i = 1; i <= particles_per_gridpoint; i++) {
                        iii = i + j;
                }
                j += particles_per_gridpoint;
        }
        particles_per_gridpoint = Q*doping_profile[0]*LX*LZ*LY/NY/qs/NUMBER_OF_SLAVES/2;
        for (i = 1; i <= particles_per_gridpoint; i++) {
                iii = i + j;
        }
        printf("%d  ",particles_per_gridpoint);
        j += particles_per_gridpoint;
        particles_per_gridpoint = Q*doping_profile[NY]*LX*LZ*LY/NY/qs/NUMBER_OF_SLAVES/2;
        for (i = 1; i <= particles_per_gridpoint; i++) {
                iii = i + j;
        }
                printf("%d  ",particles_per_gridpoint);
}

// initialise
void init()
{
  int i,ii;
  realtype dum,temp_y,x,temp_dope[NY+1];
  COUNTE_M = TOTAL_M/NUMBER_OF_SLAVES;
  srand(64);
  hy = LY/NY;
  HY = hy;
  simtime = 0;
  i_ar0 = abs(dround((LY1+LY2+LY3)/HY));
  i_ar1 = abs(dround((LY-LY11)/HY));
  n_ar = i_ar1 - i_ar0 + 1;
  i1_con0 = dround((LY1*0.1)/HY);
  i1_con1 = dround((LY1*0.9)/HY);
  i2_con0 = dround((LY - 0.9*LY11)/HY);
  i2_con1 = dround((LY - 0.1*LY11)/HY);
  n1_con = i1_con1 - i1_con0 + 1;
  n2_con = i2_con1 - i2_con0 + 1;


  ip_count = 0;
  s_max = 0;
  VB[0] = VB0; VB[1] = VB1; VB[2] = VB2; VB[3] = VB3; VB[4] = VB4;
  VB[5] = VB5; VB[6] = VB6; VB[7] = VB7; VB[8] = VB8; VB[9] = VB9;

  V1[0] = V1_0; V1[1] = V1_1; V1[2] = V1_2; V1[3] = V1_3; V1[4] = V1_4;
```

```
  V1[5] = V1_5; V1[6] = V1_6; V1[7] = V1_7; V1[8] = V1_8; V1[9] = V1_9;

  V2[0] = V2_0; V2[1] = V2_1; V2[2] = V2_2; V2[3] = V2_3; V2[4] = V2_4;
  V2[5] = V2_5; V2[6] = V2_6; V2[7] = V2_7; V2[8] = V2_8; V2[9] = V2_9;

  t_inc[0] = T0; t_inc[1] = T1; t_inc[2] = T2; t_inc[3] = T3; t_inc[4] = T4;
  t_inc[5] = T5; t_inc[6] = T6; t_inc[7] = T7; t_inc[8] = T8; t_inc[9] = T9;
  t_inc[10] = TSIM;

  LAG[0] = PHI0; LAG[1] = PHI1; LAG[2] = PHI2; LAG[3] = PHI3; LAG[4] = PHI4;
  LAG[5] = PHI5; LAG[6] = PHI6; LAG[7] = PHI7; LAG[8] = PHI8; LAG[9] = PHI9;

  i_v = 0;

  for (ii = 0; ii <= NY; ii++) {
    temp_y = ((realtype) ii)*LY/NY;
    x = x_fact(temp_y);
    if (temp_y <= (1 - G1_FACTOR)*LY1)
          doping_profile[ii] = NDOPE1;
    if (temp_y <= LY1 && temp_y > (1-G1_FACTOR)*LY1)
          doping_profile[ii] = NDOPE1*exp(1/G1_FACTOR*(temp_y-(1-G1_FACTOR)*LY1)/LY1*log(NDOPE1_FACTOR));
    if (temp_y <= (LY1+LY2) && temp_y > LY1)
          doping_profile[ii] = NDOPE2;
    if (temp_y <= (LY1+LY2+LY3) && temp_y > (LY1+LY2))
          doping_profile[ii] = NDOPE3;
    if (temp_y <= (LY1+LY2+LY3+LY4) && temp_y > (LY1+LY2+LY3))
          doping_profile[ii] = NDOPE4;
    if (temp_y <= (LY1+LY2+LY3+LY4+(1-G_FACTOR)*LY5) && temp_y > (LY1+LY2+LY3+LY4))
          doping_profile[ii] = NDOPE5;
    if (temp_y <= (LY1+LY2+LY3+LY4+LY5) && temp_y > (LY1+LY2+LY3+LY4+(1-G_FACTOR)*LY5))
          doping_profile[ii] =
NDOPE5*exp(1/G_FACTOR*(temp_y-LY1-LY2-LY3-LY4-(1-G_FACTOR)*LY5)/LY5*log(NDOPE_FACTOR));
                  if (temp_y <= (LY1+LY2+LY3+LY4+LY5+LY6) && temp_y > (LY1+LY2+LY3+LY4+LY5))
          doping_profile[ii] = NDOPE6;
    if (temp_y <= (LY1+LY2+LY3+LY4+LY5+LY6+LY7)  && temp_y > (LY1+LY2+LY3+LY4+LY5+LY6))
          doping_profile[ii] = NDOPE7;
    if (temp_y <= (LY1+LY2+LY3+LY4+LY5+LY6+LY7+LY8) && temp_y > (LY1+LY2+LY3+LY4+LY5+LY6+LY7))
          doping_profile[ii] = NDOPE8;
    if (temp_y <= (LY1+LY2+LY3+LY4+LY5+LY6+LY7+LY8+LY9) && temp_y > (LY1+LY2+LY3+LY4+LY5+LY6+LY7+LY8))
          doping_profile[ii] = NDOPE9;
    if (temp_y <= (LY1+LY2+LY3+LY4+LY5+LY6+LY7+LY8+LY9+(1-G_FACTOR)*LY10) &&
                              temp_y > (LY1+LY2+LY3+LY4+LY5+LY6+LY7+LY8+LY9))
          doping_profile[ii] = NDOPE10;
    if (temp_y <= (LY1+LY2+LY3+LY4+LY5+LY6+LY7+LY8+LY9+LY10) &&
           temp_y > (LY1+LY2+LY3+LY4+LY5+LY6+LY7+LY8+LY9+(1-G_FACTOR)*LY10))
                              doping_profile[ii] =
NDOPE10*exp(1/G_FACTOR*(temp_y-LY1-LY2-LY3-LY4-LY5-LY6-LY7-LY8-LY9-(1-G_FACTOR)*LY10)
                                                                  /LY10*log(NDOPE_FACTOR));
    if (temp_y <= LY && temp_y > (LY1+LY2+LY3+LY4+LY5+LY6+LY7+LY8+LY9+LY10))
                              doping_profile[ii] = NDOPE11;
    dielec[ii] = dielec_s(x,T_INIT);
  }

  qs = 0;
  dum = Q*LX*LZ*HY/(QSF*TOTAL_M);
  for (i = 1; i <= (NY-1); i++)
```

```
  qs += doping_profile[i];
qs += doping_profile[0]/2;
qs += doping_profile[NY]/2;
qs = dum*qs;

for (i = 0; i <= NY; i++)
  temp_dope[i] = log(doping_profile[i])/log(10);
tofile("nprofile",0,temp_dope);
tofile("nprofile",1,doping_profile);
tofile("dielec",0,dielec);
p1 = 0;
p2 = 0;
if ((H1_FINISH - H1_START) != 0)
  p1 = X_MAX1/(H1_FINISH - H1_START);
if((H2_FINISH - H2_START) != 0)
  p2 = X_MAX2/(H2_FINISH - H2_START);
printf("\np1 = %f  p2 = %f",p1,p2);
//initialize poisson solver parameters
for (i = 1; i <= (NY-1); i++) {
   temp_y = ((realtype) i)*LY/NY;
   x = x_fact(temp_y);
   p_param[0][i] = -3.12*EPS0*p_fact(temp_y)/2/HY - EPS0*dielec[i]/HY/HY;
   p_param[1][i] = 2*EPS0*dielec[i]/HY/HY;
   p_param[2][i] = 3.12*EPS0*p_fact(temp_y)/2/HY - EPS0*dielec[i]/HY/HY;
}
p_param[3][1] = p_param[2][1]/p_param[1][1];
for (i = 2; i <= (NY-2); i++)
  p_param[3][i] = p_param[2][i]/(p_param[1][i] - p_param[0][i]*p_param[3][i-1]);
          p_param[3][NY-1] = p_param[1][NY-1]/p_param[0][NY-1];


for (ii = 0; ii <= NY; ii++) {
   pddi_v[ii] = 0;
}
thermal_y[0] = 0;
i = NDOMAIN*(NT + 1) + 1;
thermal_y[i] = LY + L_ADD;
          if (NDOMAIN == 1) {
            for (i = 0; i <= NT; i++)
              thermal_y[i+1] = LY1 + ((realtype) i)*(LY - LY11 - LY1)/(realtype)NT;
          }
if (NDOMAIN == 2) {
   for (i = 0; i <= NT; i++) {
           thermal_y[i+1] = LY1 + ((realtype) i)*(LY2+LY3+LY4+LY5)/(realtype)NT;
           thermal_y[i+NT+2] = LY1+LY2+LY3+LY4+LY5+LY6 +
            ((realtype) i)*(LY7+LY8+LY9+LY10)/(realtype)NT;
   }
}
for (i = 1; i <= (NDOMAIN*(NT + 1) + 1); i ++)
  k_t[i] = C_THERMAL;
for (i = 1; i <= (NDOMAIN*(NT + 1) + 1); i ++)
  a_t[i] = k_t[i]/(thermal_y[i] - thermal_y[i-1]);
for (i = 1; i <= (NDOMAIN*(NT + 1)); i ++)
  c_t[i] = -(a_t[i] + a_t[i+1])/a_t[i+1];
for (i = 1; i <= (NDOMAIN*(NT + 1)); i ++)
  d_t[i] = a_t[i]/a_t[i+1];
```

```
  i = NDOMAIN*(NT + 1);
            w_t[i] = d_t[i]/c_t[i];
  for (i = (NDOMAIN*(NT + 1) - 1); i >= 1; i --)
    w_t[i] = d_t[i]/(c_t[i] - w_t[i+1]);
  w_t[0] = -1;
  for (i = 0; i <= NY; i++) {
     t_distr[i] = T_INIT;
     c_sum[i] = 0;
     vav_sum[i] = 0;
  }
  ibot = 0;
  itop = 0;
  vdiode[0] = 0;
  vdiode[1] = 0;
  idiode[0] = itop;
  idiode[1] = itop;
  iext[0] = 0;
  iext[1] = 0;
  itop_av = 0;
  ibot_av = 0;
  i_th = 0;
  jp_sum = 0;
  r_init_();
  printf("\ncounte = %d",COUNTE_M);
}
```

m_pvm.h

```
void initpvm_alpha()
{
  FILE *infile;
  char inputline[128];
  int nhost,narch,numt,i,tempstid;
  int tempstidv[100];
  char slavename[100],name[100];
  struct pvmhostinfo *hostp[100];
  // enrolls master into PVM
  mtid = pvm_mytid();
  pvm_config(&nhost,&narch,hostp);
  NUMBER_OF_SLAVES = nhost-1;
            printf("Number of slaves = %d\n",NUMBER_OF_SLAVES);
  infile = fopen("hostfile", "r");
  if (infile == NULL) {
    printf("Unable to open hostfile.\n");
    exit(1);
  }
  //start up slaves
  for (i = 1; i <= (NUMBER_OF_SLAVES); i++) {
            if (fgets(inputline, 128, infile) == NULL) {
                      i = NUMBER_OF_SLAVES+1;
                      continue;
            };
            printf("Starting slave on %s", inputline);
```

```
            fflush(stdout);
            numt = pvm_spawn(PROG_SLAVE, (char**)0, 0, inputline, 1, &tempstid);
            tempstidv[i] = tempstid;
            fflush(stdout);
            if ( numt < 1 ) {
                  printf("Trouble spawning slave %d. Aborting.\n",i);
                  fflush(stdout);
                  pvm_kill(tempstidv[i]);
                  pvm_exit();
                  exit(0);
            }
            stidv[i] = tempstidv[i];
  }
          fclose(infile);
}


void writearrays(double tempfield[NY + 10], double temptemp[NY+1], int temptid)
{
  int send_err;
  pvm_initsend(PvmDataDefault);
  pvm_pkdouble(tempfield,NY + 2,1);
  send_err = pvm_send(temptid, CMSGTYPE);
  if (send_err < 0 )
   printf("\nCould not send field to slave %d",temptid);
  pvm_initsend(PvmDataDefault);
  pvm_pkdouble(temptemp,NY + 1,1);
  send_err = pvm_send(temptid, TMSGTYPE);
  if (send_err < 0 )
   printf("\nCould not send temperature to slave %d",temptid);
}


void writearray(double temptemp[NY+1], int temptid)
{
  int send_err;
  pvm_initsend(PvmDataDefault);
  pvm_pkdouble(temptemp,NY + 1,1);
  send_err = pvm_send(temptid, NMSGTYPE);
  if (send_err < 0 )
   printf("\nCould not send temperature to slave %d",temptid);
}


void slavetot2slaves()
{
  int tempnumber = NUMBER_OF_SLAVES,i;

  for (i = 1; i <= NUMBER_OF_SLAVES; i++) {
    pvm_initsend(PvmDataDefault);
    pvm_pkint(&tempnumber,1,1);
    pvm_send(stidv[i], SMSGTYPE);
  }
}

void qs2slaves()
```

```
{
  int i;

  for (i = 1; i <= NUMBER_OF_SLAVES; i++) {
    pvm_initsend(PvmDataDefault);
    pvm_pkdouble(&qs,1,1);
    pvm_send(stidv[i], SMSGTYPE);
  }
}

void ndope2slaves()
{
  int j,tempnumber = NUMBER_OF_SLAVES;
  for (j = 1; j <= tempnumber; j++)
   writearray(doping_profile,stidv[j]);
}

void write2slaves()
{
  int j,tempnumber = NUMBER_OF_SLAVES;
  for (j = 1; j <= tempnumber; j++)
   writearrays(e_fieldy,t_distr,stidv[j]);
}


void readfromslaves()
{
  int tempnumber,i,j,received,tempslavenumber;
  realtype tempcharge[NY + 10],tempv[NY+1],temp_s=0,emax=0;

  accu = 0;
  charge_ex1t = 0;
  charge_ex2t = 0;
  e_exceed = 0;
  m_exceed = 0;
  for (i = 0; i <= NY; i++) {
            charge[i] = 0;
            vav[i] = 0;
            valley_info[0][i] = 0;
    valley_info[1][i] = 0;
    valley_info[2][i] = 0;
  }
  while (accu < NUMBER_OF_SLAVES) {
    received = pvm_recv(-1,CMSGTYPE);
    if (received >= 0) {
          pvm_upkdouble(tempcharge,NY + 6,1);
          stid = tempcharge[NY + 1];
          if (tempcharge[0] == 1208) {
            printf("\n ****************************");
          }
          pvm_freebuf(received);
          for (i = 0; i <= NY; i++)
           charge[i] += tempcharge[i];
          charge_ex1t += tempcharge[NY + 3];
          charge_ex2t += tempcharge[NY + 4];
          temp_s = tempcharge[NY + 5];
```

```
            //if (temp_s > s_max)
            s_max = temp_s;
            for (i = 1; i <= NUMBER_OF_SLAVES; i++)
              if (stid == stidv[i]) tempslavenumber = i;
            total[tempslavenumber] = tempcharge[NY + 2];
            accu++;
    }
    else printf("\nerror when reading from slaves");
  }
  accu = 0;
  while (accu < NUMBER_OF_SLAVES) {
    received = pvm_recv(-1,VMSGTYPE);
    if (received >= 0) {
            pvm_upkdouble(tempv,NY+1,1);
            for (i = 0; i <= NY; i++) vav[i] += tempv[i];
            accu++;
    }
    else printf("\nerror when reading from slaves");
  }
  for (i = 0; i <= NY; i++) vav[i] = vav[i]/NUMBER_OF_SLAVES;
  accu = 0;
  while (accu < NUMBER_OF_SLAVES) {
    received = pvm_recv(-1,VCMSGTYPE);
    if (received >= 0) {
            pvm_upkdouble(tempv,NY+1,1);
            for (i = 0; i <= NY; i++)
              valley_info[0][i] += tempv[i];
            accu++;
    }
    else printf("\nerror when reading from slaves");
  }
  accu = 0;
  while (accu < NUMBER_OF_SLAVES) {
    received = pvm_recv(-1,VLMSGTYPE);
    if (received >= 0) {
            pvm_upkdouble(tempv,NY+1,1);
            for (i = 0; i <= NY; i++)
              valley_info[1][i] += tempv[i];
            accu++;
    }
    else printf("\nerror when reading from slaves");
  }
  accu = 0;
  while (accu < NUMBER_OF_SLAVES) {
    received = pvm_recv(-1,VXMSGTYPE);
    if (received >= 0) {
            pvm_upkdouble(tempv,NY+1,1);
            for (i = 0; i <= NY; i++)
              valley_info[2][i] += tempv[i];
            accu++;
    }
    else printf("\nerror when reading from slaves");
  }
}
```

field_solve.h

```
void solve_poisson() {
        float g[NY],nd;
        int i;

        //calculate net charge density
        for (i = 1; i <= (NY-1); i++) {
                nd = doping_profile[i];
                charge[i] = (qs*charge[i]/(LX*HY*LZ) - Q*nd);
        }

        g[1] = (charge[1] - p_param[0][1]*VO)/p_param[1][1];
        for (i = 2; i <= (NY-2); i++)
                g[i] = (charge[i] - p_param[0][i]*g[i-1])/(p_param[1][i] -
                                                p_param[0][i]*p_param[3][i-1]);
        g[NY-1] = (charge[NY-1] - p_param[2][NY-1]*vdiode[1])/p_param[0][NY-1];

        //calculate potential and store in charge[]
        charge[NY-1] = (g[NY-1] - g[NY-2])/(p_param[3][NY-1] - p_param[3][NY-2]);
        for (i = (NY-2); i >= 1; i--)
                charge[i] = g[i] - p_param[3][i]*charge[i+1];
        charge[0] = VO;   //VO = 0
        charge[NY] = vdiode[1];
}

void e_field()
{
  int j;
  for (j = 1; j <= NY - 1; j++) {
    e_fieldy[j]  = (charge[j-1] - charge[j+1])/(2*HY);
  }
  e_fieldy[0]  = e_fieldy[1];
  e_fieldy[NY] = e_fieldy[NY-1];
}
```

thermal.h

```
void j_e_av() {
  int i;
  i_th++;
  for (i = 0; i <= NY; i++) {
    vav_sum[i] += vav_av[i];
    c_sum[i] += c_distr[i];
    j_particle[i] = -qs*vav_sum[i]*c_sum[i]/LX/LZ/HY/i_th/i_th;
  }
  for (i = 0; i <= NY; i++) {
    efield_sum[i] += e_fieldy[i];
    efield_av[i] = efield_sum[i]/i_th;
    v_acc_av[i] = vav_sum[i]/i_th;
  }
```

```
}

void pddi() {
  int i,k,i0,i1;
  for (i = 0; i <= NY; i++)
    pddi_v[i] = -j_particle[i]*efield_av[i];
  for (i = 1; i <= (NDOMAIN*(NT + 1) + 1); i++) {
    qi[i] = 0;
    i0 = thermal_y[i-1]*NY/LY;
    i1 = thermal_y[i]*NY/LY;
    if (i1 > NY)
          i1 = NY;
    //take note that pddi_v[PDD_AV_FACTOR][NY] is not included in coding: minimal effect!
    for (k = i0; k < i1; k++) {
          qi[i] += pddi_v[k]/(i1-i0);
    }
  }
}

void pddi_av_calc() {
  int i,k,i0,i1;

  i_th++;
  for (i = 0; i <= NY; i++) {
    //j_particle[i] = qs*vav[i]*c_distr[i]/LX/LZ/HY;
    pddi_sum[i] += qs*vav[i]*c_distr[i]/LX/LZ/HY*e_fieldy[i];
    pddi_av[i] = pddi_sum[i]/i_th;
    pddi_v[i] = pddi_av[i];
  }
  for (i = 1; i <= (NDOMAIN*(NT + 1) + 1); i++) {
    qi[i] = 0;
    i0 = thermal_y[i-1]*NY/LY;
    i1 = thermal_y[i]*NY/LY;
    //take note that pddi_v[PDD_AV_FACTOR][NY] is not included in coding...minimal effect anyway!
    for (k = i0; k < i1; k++) {
          qi[i] += pddi_av[k]/(i1-i0);
    }
  }
}

void temp_calc() {
  int i,ii,i0,i1;
  realtype b_t[NT+NT+4],e_t[NT+NT+4],g_t[NT+NT+4],y_temp,dt=0,dummy;

  //calculate heatsink temperature
  for (i = 1; i <= (NY-1); i++)
    dt += pddi_v[i];
  dt += pddi_v[0]/2;
  dt += pddi_v[NY]/2;
  dt = dt*HY*(LX/sqrt(PI)/KHS + L_IHS/KIHS);
  //dt = ip_dc/LX/LZ*VBIAS*(LX/sqrt(PI)/KHS + L_IHS/KIHS);
  ths = TAMB_0 + dt;
  //calculate temperature profile
  for (i = 1; i <= (NDOMAIN*(NT+1)); i++)
    b_t[i] = (qi[i]*(-thermal_y[i] + thermal_y[i-1]) + qi[i+1]*(-thermal_y[i+1] + thermal_y[i]))/2;
  for (i = (NDOMAIN*(NT + 1)); i >= 1; i--)
```

```
    e_t[i] = b_t[i]/a_t[i+1];
  i = NDOMAIN*(NT + 1);
  g_t[i] = (e_t[i] - ths)/c_t[i];
  for (i = (NDOMAIN*(NT + 1) - 1); i >= 1; i--)
    g_t[i] = (e_t[i] - g_t[i+1])/(c_t[i] - w_t[i+1]);
  g_t[0] = -qi[1]*thermal_y[1]*thermal_y[1]/2/k_t[1];
  thermal_t[0] = (g_t[1] - g_t[0])/(w_t[1] - w_t[0]);
  for (i = 1; i <= (NDOMAIN*(NT + 1)); i++)
    thermal_t[i] = g_t[i] - w_t[i]*thermal_t[i-1];
  ii = NDOMAIN*(NT + 1) + 1;
  thermal_t[ii] = ths;
  ii = 1;
  for (i = 1; i <= NY; i++) {
    y_temp = (float)i*LY/(float)NY;
    if ((y_temp <= thermal_y[ii]) && (y_temp > thermal_y[ii-1]))
          t_distr[i] = thermal_t[ii-1] + (y_temp - thermal_y[ii-1])*(thermal_t[ii] - thermal_t[ii-1])/
          (thermal_y[ii] - thermal_y[ii-1]);
    if (y_temp >= thermal_y[ii]) {
          t_distr[i] = thermal_t[ii];
          ii = ii + 1;
    }
  }
  t_distr[0] = thermal_t[0];
}
```

update.h

```
//particle current at contact 1 (anode)
realtype current1_0()
{
  return(-qs*(charge_ex1t)/TSTEP);
}

//particle current at contact 2 (cathode)
realtype current2_0()
{
  return(qs*(charge_ex2t)/TSTEP);
}

//particle current at cathode contact based on charge x velocity
realtype current1_contact()
{
  int i;
  float dum1 = 0,dum2 = 0;
  dum1 = qs/n1_con/HY;
  for (i = i1_con0; i <= i1_con1; i++)
    dum2 += vav[i]*c_distr[i];
  return(dum1*dum2);
}

//particle current at anode contact based on charge x velocity
realtype current2_contact()
{
  int i;
```

```
  float dum1 = 0,dum2 = 0;
  dum1 = qs/n2_con/HY;
  for (i = i2_con0; i <= i2_con1; i++)
    dum2 += vav[i]*c_distr[i];
  return(dum1*dum2);
}

//particle current at cathode contact based on charge x average velocity
realtype current2_contact_av()
{
  int i;
  float dum1 = 0,dum2 = 0;
  dum1 = qs/n2_con/HY;
  for (i = i2_con0; i <= i2_con1; i++)
    dum2 += vav_av[i]*c_distr[i];
  return(dum1*dum2);
}

//displacement current based on averaged field difference across cathode contact region
realtype current_displ_av()
{
  int i;
  float dum1 = 0;
  for (i = i2_con0; i <= i2_con1; i++)
    dum1 += (e_fieldy[i] - old_poisson_field[i])*dielec_s(0,t_distr[i]);
  dum1 = dum1/n2_con;
  return(dum1*LX*LZ*EPS0/TSTEP);
}

//displacement current based on averaged field difference across active region
realtype current_displ_ar()
{
  int i;
  float dum1 = 0;
  for (i = i_ar0; i <= i_ar0; i++)
    dum1 += (e_fieldy[i] - old_poisson_field[i])*dielec_s(0,t_distr[i]);
  return(dum1*LX*LZ*EPS0/TSTEP);
}

//displacement current at cathode contact
realtype current_displ_0()
{
  int i;
  float dum1 = 0;
  for (i = 0; i <= 0; i++)
    dum1 += (e_fieldy[i] - old_poisson_field[i])*dielec_s(0,t_distr[i]);
  return(dum1*LX*LZ*EPS0/TSTEP);
}

//displacement current at anode contact
realtype current_displ_NY()
{
  int i;
  float dum1 = 0;
  for (i = NY; i <= NY; i++)
    dum1 += (e_fieldy[i] - old_poisson_field[i])*dielec_s(0,t_distr[i]);
```

```
  return(dum1*LX*LZ*EPS0/TSTEP);
}

//calculate new parameters for next TSTEP-update
void new_volt_current()
{
  vdiode[0] = vdiode[1];
  iext[0] = iext[1];
  vdiode[1] = TSTEP/CCAP*(iext[0] - idiode[0]) + vdiode[0];
  iext[1] = iext[0] - (vdiode[1] - vdiode[0])/RLOAD - TSTEP*(vdiode[1] - VBIAS)/LIND;
}

//calculate new parameters for next TSTEP-update
void vi_voltage_driven()
{
  int temp_i_v = i_v + 1;
  if (simtime > t_inc[temp_i_v])
    i_v++;
  vdiode[0] = vdiode[1];
  iext[0] = iext[1];
  iext[1] = -qs*(charge_ex1t)/TSTEP;
  v_source = V1[i_v]*sin(2*PI*FREQ*simtime) + V2[i_v]*sin(4*PI*FREQ*simtime + LAG[i_v]*PI/180) + VB[i_v];
  vdiode[1] = v_source - R_DC*ipart_con2;
  iext[1] = idiode[0];   //this is the current that is written to iv.txt

}

//update current and field distributions
void distr_field_current()
{
  int i,tempnumber = NUMBER_OF_SLAVES;

  for (i = 0; i <= NY; i++) {
    c_distr[i] = charge[i];
  }
  ensemble_tot = 0;
  for (i = 1; i <= tempnumber; i++)
    ensemble_tot = ensemble_tot + total[i];
  ibot  = -current1_0();
  itop  = -current2_0();
  ipart_con1 = -current1_contact();
  ipart_con2 = -current2_contact();
  ipart_con_av = (ipart_con1 + ipart_con2)/2;
  ipartav = -current2_contact_av();
  ipart = ipart_con2;
  idiode[0] = ipartav;
  idisp_av = -current_displ_av();
  idisp_ar = -current_displ_ar();
  idisp_0 = -current_displ_0();
  idisp_NY = -current_displ_NY();

  if (CAVITY == 0) vdiode[1] = VBIAS - R_DC*ipart_con_av;

  if (CAVITY == 1) {
                        if (iter > 2) {
                          //new_volt_current();
```

```
                    vi_voltage_driven();
                    for (i = 0; i <= NY; i++)
                      e_pred_field[i] = 1.5*e_fieldy[i] - 0.5*old_poisson_field[i];  //not necessary...see m_pvm
                  }
          }
          for (i = 0; i <= NY; i++)
                  old_poisson_field[i] = e_fieldy[i];
          old_top_field = old_poisson_field[NY]; //value at previous charge receive
  old_bot_field = old_poisson_field[0];
  solve_poisson();
  e_field();
  if (simtime > 20e-12) {
            if (iter%PDD_AV_FACTOR == 0) {
              ip_count++;
              ip_dc_sum += ipart;
              ip_dc = ip_dc_sum/ip_count;
              j_e_av();
              pddi();
              if (T_UPDATE == 1)
                      temp_calc();
              if (T_UPDATE == 0) {
                      for (i = 0; i <= NY; i++)
                        t_distr[i] = TAMB;
              }
            }
          }
          for (i = 0; i <= NY; i++)
                              if (t_distr[i] > 600) t_distr[i] = 600;
}
```

A2 SLAVE NODE PROGRAMME: CODE LISTING

lfslave.c

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <math.h>
#include "pvm3.h"
#include "mathadd.h"
#include "global.h"
#include "svt_variable.h"
#include "s_pvm.h"
#include "s_param.h"
#include "g_param.h"
#include "scattable_xt_new.h"
#include "s_init.h"
#include "randgen.h"
#include "dynamix_x.h"
#include "boundary.h"
#include "scattering3.h"
#include "transfer.h"
#include "c_assign.h"
#include "vav.h"
#include "sat_statistics.h"

main()
{
  int valley,i;
  initpvm();
  slavetotreceive();
  qsreceive();
  ndopereceive();
  init_const();
  initiate();
  e_exceed = 0;
  m_exceed = 0;
  vav_init();
  scattable();
  charge_assign_();
  write2master();
  do {
    teller++;
    if (teller%2 != 0) {
          readfrommaster();
          for (i = 0; i <= NY; i++) {
            e_old_3[2][i] = charge[i];
            e_old[1][i] = charge[i];
            if (teller == 1)
              e_old[0][i] = e_old[1][i];
            if (teller <= 3) {
              e_old_3[1][i] = e_old_3[2][i];
              e_old_3[0][i] = e_old_3[1][i];
```

```
            }
        }
    }
    if (teller%2 == 0) {
        if (charge[NY + 1] == 0) {
          sat_stat();
          write2master();
          vav_init();
          e_exceed = 0;
          m_exceed = 0;
        }
        exc1 = 0;
        exc2 = 0;
        for (i = 0; i <= NY; i++) {
          charge[i] = 2*e_old[1][i] - e_old[0][i];
          e_old[0][i] = e_old[1][i];
          e_old_3[0][i] = e_old_3[1][i];
          e_old_3[1][i] = e_old_3[2][i];
        }
    }
    excount1 = 0;
    excount2 = 0;
    excount = 0;
    extinct_flag = 0;
    for (eleccount = 1; eleccount <= ensemble_tot; eleccount++) {
        tf_acc = 0;
        excount = excount1 + excount2;
        e_inc = eleccount - excount;
        if (extinct_flag == 1) {
          point = ensemble_tot - (excount - 1);
          transfer_to_old(point,e_inc);
        }
        extinct_flag = 0;
        do {
          randomgenerate();
          if (tf_over[e_inc] > 0) {
            tf = tf_over[e_inc];
            tf_over[e_inc] = 0;
          }
          else {
            tf = -log(randfree)/SCAT_TOT;
          }
          tf_acc = tf_acc + tf;
          if (tf_acc > TSTEP/2) {
            tf_over[e_inc] = tf_acc - TSTEP/2;
            tf = tf - tf_over[e_inc];
          }
          calc_dynamix();
          boundary_test_0();
          valley = sat[e_inc];
          if (extinct_flag == 0) {
            if (tf_acc > TSTEP/2) {
                    flag[valley][11] = 1;
            }
            else {
                    scatter_rates();
```

```
                        selfscatter();
                }
                if (flag[valley][11] == 1) {
                        update_selfscatter();
                }
                else if (flag[valley][11] == 0) {
                        scatter_choice();
                        update_scatter();
                }
              }
            } while (tf_acc <= TSTEP/2 && extinct_flag == 0);
    }
    ensemble_tot -= excount;
    if (teller%2 != 0) {
            charge_assign_();
            vav_calc();
    }
    exc1 += excount1;
    exc2 += excount2;
    if (ry[E_TOT_MAX_S] == 1208) charge[0] = 1208;
    ry[E_TOT_MAX_S] = 0;
  } while (teller > 0);
  return 0;
}
```

svt_variable.h

```
#define TLENGTH 10000
#define X_INCR 5

const int N = 10;

const double
KF = 1,
PI = 3.14159,
Q = -1.60218E-19,
H_ = 1.05458E-34,
MD_C = 0.067*9.1095E-31,
MD_L = 0.29*9.1095e-31,
MD_X = 0.45*9.1095e-31,
MD_XCOEF_C = 0.083*9.1095E-31,
MD_XCOEF_L = 0.040*9.1095E-31,
MD_XCOEF_X = -0.07*9.1095E-31,
MC_C = 0.067*9.1095E-31,
MC_L = 0.29*9.1095e-31,
MC_X = 0.45*9.1095e-31,
MC_XCOEF_C = 0.083*9.1095E-31,
MC_XCOEF_L = 0.10*9.1095E-31,
MC_XCOEF_X = -0.14*9.1095E-31,
ZCL = 4,
ZLC = 1,
ZCX = 3,
```

```
ZXC = 1,
ZLX = 3,
ZLL = 3,
ZXX = 2,
ZXL = 4,
DACC = 7*1.60218E-19,
DACL = 7*1.60218E-19,
DACX = 7*1.60218E-19,
DLL = 1.1*16.0218E-9,
DCL = 1.1*16.0218E-9,
DCX = 1.1*16.0218E-9,
DLX = 1.1*16.0218E-9,
DXX = 1.1*16.0218E-9,
GAP2 = 0.447*1.60218E-19,
GAP1 = 0.284*1.60218E-19,
GAPVC = 1.72*1.602E-19,
GAPVL = 2.05*1.602E-19,
GAPVX = 2.17*1.602e-19,
KB  = 1.38066E-23,
VS  = 5240,
RHO = 5360,
WOP = 5.37E13,
WCL = 4.6E13,
WCX = 4.6E13,
WLL = 4.41E13,
WXX = 4.6E13,
WLX = 4.6E13,
WE = 4.558E13,
WN = 4.558e13,
Z  = 1,
SCAT_TOT = 7.5E14,
ALPHA1 = 1.2*0.56/1.602e-19,  //Kim,Hess
ALPHAL = 0.4/1.602e-19,  //Kim,Hess
ALPHAX = 0.6/1.602e-19,  //Kim,Hess
E_AFF_FACTOR = 0.65*1.247*1.602E-19,
E_MAX = 3E-19;

typedef                 double                  realtype;

typedef                         char                            boolean;

realtype
m_0,n_op,n_eq,n_neq,c_op,konst_impu,energy,tf_acc,k0_mag,k1_mag,hwo,hwe,hwn,randfree,
randscat,randang1,randang2,rand_tr,tf,ey,doping,hy,HY,energy_max,charge[NY+10],e_old_3[3][NY+1],e_old[2][NY+1],
qs,c_ac,c_cl,c_lc,c_cx,c_xc,c_lx,c_inter,c_impu,c_impu_new,HK_2,H_K,H2_2,H_2,
k0x[E_TOT_MAX_S+1],k0y[E_TOT_MAX_S+1],k0z[E_TOT_MAX_S+1],k1x[E_TOT_MAX_S+1],k1y[E_TOT_MAX_S+1],
k1z[E_TOT_MAX_S+1],ry[E_TOT_MAX_S+1],tf_over[E_TOT_MAX_S+1],rate[15],engy[15],t_distr[NY+1],
scat01[TLENGTH + 1][X_INCR+1][T_INCR+1],scat02[TLENGTH + 1][X_INCR+1][T_INCR+1],
scat03[TLENGTH + 1][X_INCR+1][T_INCR+1],scat04[TLENGTH + 1][X_INCR+1][T_INCR+1],
scat05[TLENGTH + 1][X_INCR+1][T_INCR+1],scat06[TLENGTH + 1][X_INCR+1][T_INCR+1],
scat07[TLENGTH + 1][X_INCR+1][T_INCR+1],scat08[TLENGTH + 1][X_INCR+1][T_INCR+1],
scat09[TLENGTH + 1][X_INCR+1][T_INCR+1],scat010[TLENGTH + 1][X_INCR+1][T_INCR+1],
scat11[TLENGTH + 1][X_INCR+1][T_INCR+1],scat12[TLENGTH + 1][X_INCR+1][T_INCR+1],
scat13[TLENGTH + 1][X_INCR+1][T_INCR+1],scat14[TLENGTH + 1][X_INCR+1][T_INCR+1],
scat15[TLENGTH + 1][X_INCR+1][T_INCR+1],scat16[TLENGTH + 1][X_INCR+1][T_INCR+1],
scat17[TLENGTH + 1][X_INCR+1][T_INCR+1],scat18[TLENGTH + 1][X_INCR+1][T_INCR+1],
```

```
scat19[TLENGTH + 1][X_INCR+1][T_INCR+1],scat110[TLENGTH + 1][X_INCR+1][T_INCR+1],
scat21[TLENGTH + 1][X_INCR+1][T_INCR+1],scat22[TLENGTH + 1][X_INCR+1][T_INCR+1],
scat23[TLENGTH + 1][X_INCR+1][T_INCR+1],scat24[TLENGTH + 1][X_INCR+1][T_INCR+1],
scat25[TLENGTH + 1][X_INCR+1][T_INCR+1],scat26[TLENGTH + 1][X_INCR+1][T_INCR+1],
scat27[TLENGTH + 1][X_INCR+1][T_INCR+1],scat28[TLENGTH + 1][X_INCR+1][T_INCR+1],
scat29[TLENGTH + 1][X_INCR+1][T_INCR+1],scat210[TLENGTH + 1][X_INCR+1][T_INCR+1],
delta_e[3][11][X_INCR+1][T_INCR+1],alpha[3],dmass[3],cmass[3],dmass_xcoef[3],
vav[NY+1],vav_av[NY+1],state_info[2][NY+1],valley_info[3][NY+1],vavcount[NY+1],scatrate,
doping_profile[NY+1],p1,p2,e_aff_factor[3],temperature,energy_max,s_max;

int
qadd1[6],qadd2[6],excount,excount1,excount2,exc1,exc2,eleccount,flagno,stid,
mtid,ensemble_tot,e_inc,point,sat[E_TOT_MAX_S+1],sat1[3][11],
extinct_flag,flag[3][12],COUNTE_S,NUMBER_OF_SLAVES,oldvalley,e_exceed,m_exceed;

long int teller;
```

s_pvm.h

```
void initpvm()
{
        stid = pvm_mytid();
        mtid = pvm_parent();
}

void write2master() {
        int i,send_err;
        realtype temp_v[NY+1];
        charge[NY + 5] = (float) s_max;
        charge[NY + 4] = (float) exc2;
        charge[NY + 3] = (float) exc1;
        charge[NY + 2] = (float) ensemble_tot;
        charge[NY + 1] = (float) stid;
        pvm_initsend(PvmDataDefault);
        pvm_pkdouble(charge,NY + 6,1);
        send_err = pvm_send(mtid,CMSGTYPE);
        if (send_err < 0 )
                printf("\nCould not send position vector to master");
        pvm_initsend(PvmDataDefault);
        pvm_pkdouble(vav,NY + 1,1);
        send_err = pvm_send(mtid, VMSGTYPE);
        if (send_err < 0 )
                printf("\nCould not send position vector to master");
        for (i = 0; i <= NY; i++)
                temp_v[i] = valley_info[0][i];
        pvm_initsend(PvmDataDefault);
        pvm_pkdouble(temp_v,NY + 1,1);
        send_err = pvm_send(mtid, VCMSGTYPE);
        if (send_err < 0 )
                printf("\nCould not send position vector to master");
        for (i = 0; i <= NY; i++)
                temp_v[i] = valley_info[1][i];
```

```
        pvm_initsend(PvmDataDefault);
        pvm_pkdouble(temp_v,NY + 1,1);
        send_err = pvm_send(mtid, VLMSGTYPE);
        if (send_err < 0 )
                printf("\nCould not send position vector to master");

        for (i = 0; i <= NY; i++)
                temp_v[i] = valley_info[2][i];
        pvm_initsend(PvmDataDefault);
        pvm_pkdouble(temp_v,NY + 1,1);
        send_err = pvm_send(mtid, VXMSGTYPE);
        if (send_err < 0 )
                printf("\nCould not send position vector to master");
}

void readfrommaster() {
        pvm_recv(mtid,CMSGTYPE);
        pvm_upkdouble(charge,NY + 2,1);
        pvm_recv(mtid,TMSGTYPE);
        pvm_upkdouble(t_distr,NY + 1,1);
}

void slavetotreceive() {
        pvm_recv(mtid,SMSGTYPE);
        pvm_upkint(&NUMBER_OF_SLAVES,1,1);
}

void qsreceive() {
        pvm_recv(mtid,SMSGTYPE);
        pvm_upkdouble(&qs,1,1);
}

void ndopereceive() {
        pvm_recv(mtid,NMSGTYPE);
        pvm_upkdouble(doping_profile,NY + 1,1);
}
```

s_param.h

```
realtype md_xt(int valley, realtype xf, realtype T) {
  if (xf > 0) {
    if (valley == 0)
        return(MD_C - 1.85e-5*(T - 300)*9.1095E-31 + MD_XCOEF_C*xf);
    if (valley == 1)
        return(MD_L + MD_XCOEF_L*xf);
    if (valley == 2)
        return(MD_X + MD_XCOEF_X*xf);
  }
  else {
    if (valley == 0)
        return(MD_C - 1.85e-5*(T - 300)*9.1095E-31);
    if (valley == 1)
        return(MD_L);
```

```
    if (valley == 2)
                                        return(MD_X);
  }
}


realtype mc_xt(int valley, realtype xf, realtype T) {
  if (xf > 0) {
    if (valley == 0)
          return(MC_C - 1.85e-5*(T - 300)*9.1095E-31 + MD_XCOEF_C*xf);
    if (valley == 1)
          return(MC_L + MD_XCOEF_L*xf);
    if (valley == 2)
          return(MC_X + MD_XCOEF_X*xf);
  }
  else {
    if (valley == 0)
          return(MC_C - 1.85e-5*(T - 300)*9.1095E-31);
    if (valley == 1)
          return(MC_L);
    if (valley == 2)
          return(MC_X);
  }
}

realtype vs(float x) {
  return(VS);
}

realtype rho(float x) {
  return(RHO - 1600*x);
}

realtype dacc(float x) {
  return(DACC - 0.50*1.60218E-19*x);
}

realtype dacl(float x) {
  return(DACL + 0.32*1.60218E-19*x);
}

realtype dacx(float x) {
  return(DACX + 0.20*1.60218E-19*x);
}

realtype dcl(float x) {
  return(DCL);
}

realtype dcx(float x) {
  return(DCX);
}

realtype dlx(float x) {
  return(DLX);
}
```

```
realtype dxx(float x) {
  return(DXX);
}

realtype dll(float x) {
  return(DLL);
}

realtype alpha1(float x) {
  realtype result;
  if (ALPHA1 == 0) result = 0;
  else result = ALPHA1 - 0.94*x/1.60218E-19;
  return(result);
}

realtype alphal(float x) {
  realtype result;
  if (ALPHAL == 0) result = 0;
  else result = ALPHAL - 0.038*x/1.60218E-19;
  return(result);
}

realtype alphax(float x) {
  realtype result;
  if (ALPHAX == 0) result = 0;
  else result = ALPHAX;
  return(result);
}

realtype x_alpha(int valley, float x) {
  float result;
  if (valley == 0) {
    if (ALPHA1 == 0) result = 0;
    else result = ALPHA1 - 0.94*x/1.60218E-19;
  }
  if (valley == 1) {
    if (ALPHAL == 0) result = 0;
    else result = ALPHAL - 0.038*x/1.60218E-19;
  }
  if (valley == 2) {
    if (ALPHAX == 0) result = 0;
    else result = ALPHAX;
  }
  return(result);
}

realtype x0_alpha(int val) {
  float result;
  if (val == 0)
    result = ALPHA1;
  if (val == 1 )
    result = ALPHAL;
  if (val == 2)
    result = ALPHAX;
  return(result);
}
```

```
realtype wop(float x) {
  return(WOP);
}

realtype wcl(float x) {
  return(WCL);
}

realtype wcx(float x) {
  return(WCX);
}

realtype wll(float x) {
  return(WLL);
}

realtype wxx(float x) {
  return(WXX);
}

realtype wlx(float x) {
  return(WLX);
}

realtype gap1(realtype x, realtype T) {
  return(GAP1 - 5.7e-5*(T - 300)*1.60218e-19 - 1.60218E-19*0.605*x);
}

realtype gap2(realtype x, realtype T) {
  return(GAP2 + 7.1e-5*(T - 300)*1.60218e-19 - 1.60218E-19*(1.122*x - 0.143*x*x));
}

realtype dEcC(float x) {
  return(0.65*1.247*1.60218E-19*x);
}

realtype dEcL(float x) {
  realtype temp_de;
  temp_de = dEcC(x) - 0.605*x*1.60218E-19;
  return(temp_de);
}

realtype dEcX(float x) {
  realtype temp_de;
  temp_de = dEcC(x) + (-1.122*x + 0.143*x*x)*1.60218E-19;
  return(temp_de);
}

realtype dEc_h(int valley, float x) {
  float result;
  if (valley == 0)
    result = dEcC(x);
  if (valley == 1)
    result = dEcL(x);
  if (valley == 2)
    result = dEcX(x);
```

```
  return(result);
}

realtype tr(realtype egy1, realtype egy2, realtype m1, realtype m2, int valley) {
  realtype k1m,k2m,t = 0,temp_engy;

  if (valley == 0) {
    k1m = sqrt(2*egy1*m1)/H_;
    k2m = sqrt(2*egy2*m2)/H_;
    t = 4*k1m*k2m/(k1m + k2m)/(k1m + k2m);
    if (rand_tr < t)
          return(1);
    else return(0);
  }
  if (valley == 1) {
    k1m = sqrt(2*egy1*m1)/H_;
    k2m = sqrt(2*egy2*m2)/H_;
    t = 4*k1m*k2m/(k1m + k2m)/(k1m + k2m);
    if (rand_tr < t)
          return(1);
    else return(0);
  }
  if (valley == 2) {
    if (egy2 < 0)
          return(0);
    else {
          k1m = sqrt(2*egy1*m1)/H_;
          k2m = sqrt(2*egy2*m2)/H_;
          t = 4*k1m*k2m/(k1m + k2m)/(k1m + k2m);
          if (rand_tr < t)
            return(1);
          else return(0);
    }
  }
}
```

scattable_xt_new

```
void constants(float x, float T)
{
  c_impu = 8*EPS0*KB/H_/H_/Q/Q;
  c_ac = sqrt(2)*KB/(PI*H_*H_*H_*H_*vs(x)*vs(x)*rho(x));
  c_op = Q*Q*wop(x)*(1/dielec_h(x,T) - 1/dielec_s(x,T))/(sqrt(2)*H_*4*PI*EPS0);
  c_inter = 1/(sqrt(2)*PI*rho(x)*H_*H_*H_);
}

realtype acoustic(realtype mass,realtype egy, realtype alpha, realtype dac) {
  return(c_ac*dac*dac*mass*sqrt(mass)*sqrt(egy*(1 + alpha*egy))*
          ((1 + alpha*egy)*(1 + alpha*egy) + (alpha*egy)*(alpha*egy)/3)/
          (1 + 2*alpha*egy));
}

realtype polaroptic_a(realtype mass,realtype egy, realtype alpha, float x) {
```

```
  realtype egy_ = egy + H_*wop(x),Af,Bf,Cf,Ff,gamma = egy*(1 + alpha*egy),
    gamma_ = egy_*(1 + alpha*egy_);
  if (gamma == 0) gamma = 1e-30;
  Af = 2*(1 + alpha*egy)*(1 + alpha*egy_) + alpha*(gamma + gamma_);
  Af = Af*Af;
  Bf = -2*alpha*sqrt(gamma)*sqrt(gamma_)*
    (4*(1 + alpha*egy)*(1 + alpha*egy_) + alpha*(gamma + gamma_));
  Cf = 4*(1 + alpha*egy)*(1 + alpha*egy_)*(1 + 2*alpha*egy)*
    (1 + 2*alpha*egy_);
  Ff = log(fabs((sqrt(gamma) + sqrt(gamma_))/(sqrt(gamma) - sqrt(gamma_))));
  Ff = (Ff*Af + Bf)/Cf;
  return(c_op*sqrt(mass)*(1 + 2*alpha*egy_)*Ff/sqrt(gamma));
}

realtype polaroptic_e(realtype mass,realtype egy, realtype alpha, float x) {
  realtype egy_ = egy - H_*wop(x),Af,Bf,Cf,Ff,gamma = egy*(1 + alpha*egy),
    gamma_ = egy_*(1 + alpha*egy_);
  if (egy_ < 0)
    return(0);
  else {
    Af = 2*(1 + alpha*egy)*(1 + alpha*egy_) + alpha*(gamma + gamma_);
    Af = Af*Af;
    Bf = -2*alpha*sqrt(gamma)*sqrt(gamma_)*
          (4*(1 + alpha*egy)*(1 + alpha*egy_) + alpha*(gamma + gamma_));
    Cf = 4*(1 + alpha*egy)*(1 + alpha*egy_)*(1 + 2*alpha*egy)*
          (1 + 2*alpha*egy_);
    Ff = log(fabs((sqrt(gamma) + sqrt(gamma_))/(sqrt(gamma) - sqrt(gamma_))));
    Ff = (Ff*Af + Bf)/Cf;
    return(c_op*sqrt(mass)*(1 + 2*alpha*egy_)*Ff/sqrt(gamma));
  }
}

realtype inter(realtype egy,realtype gapi,realtype gapj, realtype ai, realtype aj,
                realtype wij,realtype zij,realtype dij, realtype mj, realtype sign) {
  realtype egy_ = egy + sign*H_*wij - (gapj - gapi),f1,f2,F,
    gamma_ = egy_*(1 + aj*egy_);
  if (egy_ < 0 ) return(0);
  else {
    f1 = sqrt(gamma_);
    f2 = 1/(1 + 2*aj*egy_);
    F = f1*f2;
    return(zij*dij*dij*c_inter*mj*sqrt(mj)*F*(1 + ai*egy + aj*egy_ + 2/3*ai*aj*egy*egy_)/wij);
  }
}

realtype inter_old(realtype egy,realtype gapi,realtype gapj, realtype ai, realtype aj,
                realtype wij,realtype zij,realtype dij, realtype mj, realtype sign) {
  realtype egy_ = egy + sign*H_*wij - (gapj - gapi),f1,f2,F,
    gamma_ = egy_*(1 + aj*egy_);
  if (egy_ < 0 ) return(0);
  else {
    f1 = (1 + ai*egy)/(1 + 2*ai*egy);
    f2 = (1 + aj*egy_)/(1 + 2*aj*egy_);
    F = f1*f2;
    return(zij*dij*dij*c_inter*mj*sqrt(mj)*F*sqrt(gamma_)*(1 + 2*aj*egy_)/wij);
  }
```

```
}

//BH approach
realtype impurity_old(realtype mass, realtype egy, realtype alpha) {
  realtype dum;
  dum = sqrt(2)*mass*sqrt(mass)*KB*KB/PI/H_/H_/H_/H_;
  return(dum*sqrt(egy*(1 + alpha*egy))*(1 + 2*alpha*egy));
}

//CW approach
realtype impurity_cw(realtype mass, realtype egy, realtype alpha) {
  realtype dum;
  dum = exp(2.0/3.0*log(3.0/4.0/PI));
  dum = dum*PI*sqrt(2/mass)*egy*egy*(1 + 2*alpha*egy)/(egy*(1 + alpha*egy))/sqrt(egy*(1 + alpha*egy));
  return(dum);
}

scattable()
{
  int i,ii,iii;
  realtype x,T,mass0,mass1,mass2;
  for (ii = 0; ii <= X_INCR; ii++) {
    x = X_MAX*(float)ii/(float)X_INCR;
    for (iii = 0; iii <= T_INCR; iii++) {
          T = T_MIN + (T_MAX - T_MIN)*(float)iii/(float)T_INCR;
          constants(x,T);
          for (i = 0; i <= TLENGTH; i++) {
            energy = E_MAX*(float)i/TLENGTH;

            //scatter in central satellite
            mass0 = md_xt(0,x,T); //MD_C + MD_XCOEF_C*x;
            mass1 = md_xt(1,x,T); //MD_L + MD_XCOEF_L*x;
            mass2 = md_xt(2,x,T); //MD_X + MD_XCOEF_X*x;
            scat01[i][ii][iii] = acoustic(mass0,energy,alpha1(x),dacc(x));
            scat02[i][ii][iii] = polaroptic_a(mass0,energy,alpha1(x),x);
            scat03[i][ii][iii] = polaroptic_e(mass0,energy,alpha1(x),x);
            scat04[i][ii][iii] = inter_old(energy,0,gap1(x,T),alpha1(x),alphal(x),wcl(x),ZCL,dcl(x),mass1,1);
            scat05[i][ii][iii] = inter_old(energy,0,gap1(x,T),alpha1(x),alphal(x),wcl(x),ZCL,dcl(x),mass1,-1);
            scat06[i][ii][iii] = inter_old(energy,0,gap2(x,T),alpha1(x),alphax(x),wcx(x),ZCX,dcx(x),mass2,1);
            scat07[i][ii][iii] = inter_old(energy,0,gap2(x,T),alpha1(x),alphax(x),wcx(x),ZCX,dcx(x),mass2,-1);
            scat08[i][ii][iii] = 0; //impossible
            scat09[i][ii][iii] = 0; //impossible
            scat010[i][ii][iii] = impurity_old(mass0,energy,alpha1(x));
            delta_e[0][1][ii][iii] = 0;
            delta_e[0][2][ii][iii] = H_*wop(x);
            delta_e[0][3][ii][iii] = -H_*wop(x);
            delta_e[0][4][ii][iii] = H_*wcl(x) - gap1(x,T);
            delta_e[0][5][ii][iii] = -H_*wcl(x) - gap1(x,T);
            delta_e[0][6][ii][iii] = H_*wcx(x) - gap2(x,T);
            delta_e[0][7][ii][iii] = -H_*wcx(x) - gap2(x,T);
            delta_e[0][8][ii][iii] = 0;  //impossible scattering
            delta_e[0][9][ii][iii] = 0;  //impossible scattering
            delta_e[0][10][ii][iii] = 0;
            sat1[0][1] = 0;
            sat1[0][2] = 0;
            sat1[0][3] = 0;
```

```
sat1[0][4] = 1;
sat1[0][5] = 1;
sat1[0][6] = 2;
sat1[0][7] = 2;
sat1[0][8] = 0;  //impossible
sat1[0][9] = 0;  //impossible
sat1[0][10] = 0;

//scatter in L-valley
scat11[i][ii][iii] = acoustic(mass1,energy,alphal(x),dacl(x));
scat12[i][ii][iii] = polaroptic_a(mass1,energy,alphal(x),x);
scat13[i][ii][iii] = polaroptic_e(mass1,energy,alphal(x),x);
//scat12[i][ii][iii] = polaroptic_a(mass1,energy,0,x);
//scat13[i][ii][iii] = polaroptic_e(mass1,energy,0,x);
scat14[i][ii][iii] = inter_old(energy,gap1(x,T),0,alphal(x),alpha1(x),wcl(x),ZLC,dcl(x),mass0,1);
scat15[i][ii][iii] = inter_old(energy,gap1(x,T),0,alphal(x),alpha1(x),wcl(x),ZLC,dcl(x),mass0,-1);
scat16[i][ii][iii] = inter_old(energy,gap1(x,T),gap2(x,T),alphal(x),alphax(x),wlx(x),ZLX,dlx(x),mass2,1);
scat17[i][ii][iii] = inter_old(energy,gap1(x,T),gap2(x,T),alphal(x),alphax(x),wlx(x),ZLX,dlx(x),mass2,-1);
scat18[i][ii][iii] = inter_old(energy,gap1(x,T),gap1(x,T),alphal(x),alphal(x),wll(x),ZLL,dll(x),mass1,1);
scat19[i][ii][iii] = inter_old(energy,gap1(x,T),gap1(x,T),alphal(x),alphal(x),wll(x),ZLL,dll(x),mass1,-1);
scat110[i][ii][iii] = impurity_old(mass1,energy,alphal(x));
delta_e[1][1][ii][iii] = 0;
delta_e[1][2][ii][iii] = H_*wop(x);
delta_e[1][3][ii][iii] = -H_*wop(x);
delta_e[1][4][ii][iii] = H_*wcl(x) + gap1(x,T);
delta_e[1][5][ii][iii] = -H_*wcl(x) + gap1(x,T);
delta_e[1][6][ii][iii] = H_*wlx(x) + gap1(x,T) - gap2(x,T);
delta_e[1][7][ii][iii] = -H_*wlx(x) + gap1(x,T) - gap2(x,T);
delta_e[1][8][ii][iii] = H_*wll(x);
delta_e[1][9][ii][iii] = -H_*wll(x);
delta_e[1][10][ii][iii] = 0;
sat1[1][1] = 1;
sat1[1][2] = 1;
sat1[1][3] = 1;
sat1[1][4] = 0;
sat1[1][5] = 0;
sat1[1][6] = 2;
sat1[1][7] = 2;
sat1[1][8] = 1;
sat1[1][9] = 1;
sat1[1][10] = 1;

//scatter in X-valley
scat21[i][ii][iii] = acoustic(mass2,energy,alphax(x),dacx(x));
scat22[i][ii][iii] = polaroptic_a(mass2,energy,alphax(x),x);
scat23[i][ii][iii] = polaroptic_e(mass2,energy,alphax(x),x);
scat24[i][ii][iii] = inter_old(energy,gap2(x,T),0,alphax(x),alpha1(x),wcx(x),ZXC,dcx(x),mass0,1);
scat25[i][ii][iii] = inter_old(energy,gap2(x,T),0,alphax(x),alpha1(x),wcx(x),ZXC,dcx(x),mass0,-1);
scat26[i][ii][iii] = inter_old(energy,gap2(x,T),gap1(x,T),alphax(x),alphal(x),wlx(x),ZXL,dlx(x),mass1,1);
scat27[i][ii][iii] = inter_old(energy,gap2(x,T),gap1(x,T),alphax(x),alphal(x),wlx(x),ZXL,dlx(x),mass1,-1);
scat28[i][ii][iii] = inter_old(energy,gap2(x,T),gap2(x,T),alphax(x),alphax(x),wxx(x),ZXX,dxx(x),mass2,1);
scat29[i][ii][iii] = inter_old(energy,gap2(x,T),gap2(x,T),alphax(x),alphax(x),wxx(x),ZXX,dxx(x),mass2,-1);
scat210[i][ii][iii] = impurity_old(mass2,energy,alphax(x));
delta_e[2][1][ii][iii] = 0;
delta_e[2][2][ii][iii] = H_*wop(x);
delta_e[2][3][ii][iii] = -H_*wop(x);
```

```
            delta_e[2][4][ii][iii] = H_*wcx(x) + gap2(x,T);
            delta_e[2][5][ii][iii] = -H_*wcx(x) + gap2(x,T);
            delta_e[2][6][ii][iii] = H_*wlx(x) + gap2(x,T) - gap1(x,T);
            delta_e[2][7][ii][iii] = -H_*wlx(x) + gap2(x,T) - gap1(x,T);
            delta_e[2][8][ii][iii] = H_*wxx(x);
            delta_e[2][9][ii][iii] = -H_*wxx(x);
            delta_e[2][10][ii][iii] = 0;
            sat1[2][1] = 2;
            sat1[2][2] = 2;
            sat1[2][3] = 2;
            sat1[2][4] = 0;
            sat1[2][5] = 0;
            sat1[2][6] = 1;
            sat1[2][7] = 1;
            sat1[2][8] = 2;
            sat1[2][9] = 2;
            sat1[2][10] = 2;
          }
    }
  }
}
```

s_init .h

```
void r_init_() {
  int i,j=0,ii,iii,particles_per_gridpoint;

  for (ii = 1; ii <= (NY-1); ii++) {
    particles_per_gridpoint = Q*doping_profile[ii]*LX*LZ*LY/NY/qs/NUMBER_OF_SLAVES;
    for (i = 1; i <= particles_per_gridpoint; i++) {
          iii = i + j;
          //ry[iii] = (2*((realtype) rand())/RAND_MAX - 1)*(float)ii*LY/NY;
          //if (ry[iii] < 0) ry[i] = abs(ry[i]);
          //if (ry[iii] > LY) ry[i] = LY;
          ry[iii] = (float)ii*LY/NY;
    }
    j += particles_per_gridpoint;
  }
  particles_per_gridpoint = Q*doping_profile[0]*LX*LZ*LY/NY/qs/NUMBER_OF_SLAVES/2;
  for (i = 1; i <= particles_per_gridpoint; i++) {
    iii = i + j;
    ry[iii] = 0;
  }
  j += particles_per_gridpoint;
  particles_per_gridpoint = Q*doping_profile[NY]*LX*LZ*LY/NY/qs/NUMBER_OF_SLAVES/2;
  for (i = 1; i <= particles_per_gridpoint; i++) {
    iii = i + j;
    ry[iii] = LY;
  }
}
```

```
void k_init_gauss()
{
  int i,ii;
  realtype dum1,dum2;
  float nfloat;

  dum2 = sqrt(md_xt(0,0,T_INIT)*T_INIT*KB)/H_;
  nfloat = (float) N;
  for (i = 1; i <= ensemble_tot; i++) {
    dum1 = 0;
    for (ii = 1;ii <= N;ii++)
          dum1 += ((realtype) rand())/RAND_MAX;
    k0x[i] = KF*dum2*(dum1 - nfloat*0.5)/sqrt(nfloat/12);
    dum1 = 0;
    for (ii = 1;ii <= N;ii++)
          dum1 += ((realtype) rand())/RAND_MAX;
    k0y[i] = KF*dum2*(dum1 - nfloat*0.5)/sqrt(nfloat/12);
    dum1 = 0;
    for (ii = 1;ii <= N;ii++)
          dum1 += ((realtype) rand())/RAND_MAX;
    k0z[i] = KF*dum2*(dum1 - nfloat*0.5)/sqrt(nfloat/12);
  }
}

void k_init_gauss_single(int boundary, int particle_number)
{
  int ii;
  realtype dum1,dum2;
  float nfloat;

  if (boundary == 1)
    temperature = t_distr[0];
  if (boundary == 3)
    temperature = t_distr[NY];
  dum2 = sqrt(MD_C*temperature*KB)/H_;
  nfloat = (float) N;
  dum1 = 0;
  for (ii = 1;ii <= N;ii++)
    dum1 += ((realtype) rand())/RAND_MAX;
  k0x[particle_number] = KF*dum2*(dum1 - nfloat*0.5)/sqrt(nfloat/12);
  dum1 = 0;
  for (ii = 1;ii <= N;ii++)
    dum1 += ((realtype) rand())/RAND_MAX;
  k0y[particle_number] = 1*KF*(-boundary + 2)*fabs(dum2*(dum1 - nfloat*0.5)/sqrt(nfloat/12));  //important factor 0.5
  dum1 = 0;
  for (ii = 1;ii <= N;ii++)
    dum1 += ((realtype) rand())/RAND_MAX;
  k0z[particle_number] = KF*dum2*(dum1 - nfloat*0.5)/sqrt(nfloat/12);
}

void init_const()
{
  int i;
  time_t t;

  srand(stid);
```

```
  teller = 0;
  eleccount = 1;
  excount  = 0;
  excount1 = 0;
  excount2 = 0;
  exc1 = 0;
  exc2 = 0;
  tf_acc = 0;
  hy = LY/NY;
  HY = hy;
  H_K = H_/KB;
  H2_2 = H_*H_/2;
  H_2 = H_/sqrt(2);
  COUNTE_S = TOTAL_S;
  alpha[0] = ALPHA1;
  alpha[1] = ALPHAL;
  alpha[2] = ALPHAX;
  dmass[0] = MD_C;
  dmass[1] = MD_L;
  dmass[2] = MD_X;
  dmass_xcoef[0] = MD_XCOEF_C;
  dmass_xcoef[1] = MD_XCOEF_L;
  dmass_xcoef[2] = MD_XCOEF_X;
  konst_impu = 8*DIELEC_S*EPS0*KB/(Q*Q*H_*H_);
  p1 = 0;
  p2 = 0;
  if ((H1_FINISH - H1_START) != 0)
    p1 = X_MAX1/(H1_FINISH - H1_START);
  if((H2_FINISH - H2_START) != 0)
    p2 = X_MAX2/(H2_FINISH - H2_START);
  e_aff_factor[0] = E_AFF_FACTOR;
  e_aff_factor[1] = E_AFF_FACTOR - 1.60218E-19*0.605;
  e_aff_factor[2] = E_AFF_FACTOR - 1.60218E-19*1.122;
  s_max = 0;
}



void initiate()
{
  int i;
  realtype temp_y;

  ensemble_tot = dround(QSF*COUNTE_S);

  for (i = 1; i <= COUNTE_S; i++) {
    tf_over[i] = 0.0;
    sat[i] = 0;
    k1x[i] = 0;
    k1y[i] = 0;
    k1z[i] = 0;
    ry[i] = 0;
  }
  for (i = 0; i <= NY; i++) {
    t_distr[i] = TAMB_0;
    vav_av[i] = 0;
```

```
    state_info[0][i] = 0;
    state_info[1][i] = 0;
  }
  k_init_gauss();
  r_init_();
}

void re_init_mass() {
  dmass[0] = MD_C;
  dmass[1] = MD_L;
  dmass[2] = MD_X;
  dmass_xcoef[0] = MD_XCOEF_C;
  dmass_xcoef[1] = MD_XCOEF_L;
  dmass_xcoef[2] = MD_XCOEF_X;
}
```

randgen.h

```
void randomgenerate()
{
  randfree = fabs(((realtype) rand())/RAND_MAX);
  if (randfree == 0)
    randfree = 1e-17;
  randscat = fabs(SCAT_TOT*((realtype) rand())/RAND_MAX);
  if (randscat == 0)
    randscat = SCAT_TOT*1e-15;
  randang1 = fabs(((realtype) rand())/RAND_MAX);
  if (randang1 == 0)
    randang1 = 1e-17;
  randang2 = fabs(((realtype) rand())/RAND_MAX);
  if (randang2 == 0)
    randang2 = 1e-17;
  rand_tr = fabs(((realtype) rand())/RAND_MAX);
  if (rand_tr == 0)
    rand_tr = 1e-17;
}
```

dynamix_x.h

```
void calc_dynamix() {
  int ii,valley = sat[e_inc];
  realtype en1,en0,enm,en1_temp,ytemp,x,temp_alpha,temp_alpha2,transmit,x2,pp,m_d,m_c,m_c1,di,
    vtemp,kmy;
  x = x_fact(ry[e_inc]);
  pp = p_fact(ry[e_inc]);

  if (x == 0) {
    ii = abs(dround(ry[e_inc]/HY));
    if (ii > NY) ii = NY;
    if (ii < 0) ii = 0;
```

```
    temperature = t_distr[ii];
    temp_alpha = x_alpha(valley,x);
    m_d = md_xt(valley,x,temperature);
    m_c = m_d;
    m_c1 = m_c;
    ey = charge[ii];
    k1y[e_inc] = Q*tf*ey/H_ + k0y[e_inc];
    k1x[e_inc] = k0x[e_inc];
    k1z[e_inc] = k0z[e_inc];
    kmy = (k0y[e_inc] + k1y[e_inc])/2;
    enm = (k0x[e_inc]*k0x[e_inc]+kmy*kmy+k0z[e_inc]*k0z[e_inc])*H2_2/m_d;
    if (temp_alpha != 0) enm = fabs((sqrt(1 + 4*temp_alpha*enm) - 1)/2/temp_alpha);
    m_c = m_c*(1 + 2*temp_alpha*enm);
    ytemp = ry[e_inc];
    ry[e_inc]  = tf*H_*kmy/m_c + ry[e_inc];
    if (ytemp > H1_FINISH && ry[e_inc] < H1_FINISH) {
            if (H1_FINISH > H1_START) {
              ry[e_inc] = 2*H1_FINISH - ry[e_inc];
              k1y[e_inc] = fabs(k1y[e_inc]);
            }
    }
    if (ytemp > H2_FINISH && ry[e_inc] < H2_FINISH) {
            if (H2_FINISH > H2_START) {
              ry[e_inc] = 2*H2_FINISH - ry[e_inc];
              k1y[e_inc] = fabs(k1y[e_inc]);
            }
    }
}
    else {
    ii = abs(dround(ry[e_inc]/HY));
    if (ii > NY) ii = NY;
    if (ii < 0) ii = 0;
    temperature = t_distr[ii];
    temp_alpha = x_alpha(valley,x);
    m_d = md_xt(valley,x,temperature);
    m_c = m_d;
    m_c1 = m_c;
    ey = charge[ii];

    en0 = (k0x[e_inc]*k0x[e_inc]+k0y[e_inc]*k0y[e_inc]+k0z[e_inc]*k0z[e_inc])*H2_2/m_d;
    m_c = m_c*sqrt(1 + 4*temp_alpha*en0);
    if (temp_alpha != 0) en0 = (sqrt(1 + 4*temp_alpha*en0) - 1)/2/temp_alpha;
    k1y[e_inc] = tf*(Q*ey  - e_aff_factor[valley]*pp)/H_ + k0y[e_inc];
    k1x[e_inc] = k0x[e_inc];
    k1z[e_inc] = k0z[e_inc];
    ytemp = ry[e_inc];
    kmy = (k0y[e_inc] + k1y[e_inc])/2;
    enm = (k0x[e_inc]*k0x[e_inc]+kmy*kmy+k0z[e_inc]*k0z[e_inc])*H2_2/m_d;
    if (temp_alpha != 0) enm = fabs((sqrt(1 + 4*temp_alpha*enm) - 1)/2/temp_alpha);
    m_c = m_c*(1 + 2*temp_alpha*enm);
    ytemp = ry[e_inc];
    ry[e_inc]  = tf*H_*kmy/m_c + ry[e_inc];
    x2 = x_fact(ry[e_inc]);
    if (ytemp < H1_FINISH && ry[e_inc] > H1_FINISH) {
            en1 = en0 + dEc_h(valley,x);
            transmit = tr(en0,en1,m_c,dmass[valley],valley);
```

```
            if (en1 < 0)
              transmit = 0;
            if (transmit == 0) {
              ry[e_inc] = ytemp;
              k1y[e_inc] = k0y[e_inc];
              en1 = en0;
            }
            else {
              temp_alpha = x_alpha(valley,0);
              m_d = md_xt(valley,0,temperature);
              k1x[e_inc] = k1x[e_inc];
              k1z[e_inc] = k1z[e_inc];
              k1y[e_inc] = sqrt(fabs(2*m_d*en1*(1 + temp_alpha*en1)/H_/H_  - k1x[e_inc]*k1x[e_inc] - k1z[e_inc]*k1z[e_inc]));
            }
    }
    else if (ytemp < H2_FINISH && ry[e_inc] > H2_FINISH) {
            en1 = en0 + dEc_h(valley,x);
            transmit = tr(en0,en1,m_c,dmass[valley],valley);
            if (en1 < 0)
              transmit = 0;
            if (transmit == 0) {
              ry[e_inc] = ytemp;
              k1y[e_inc] = k0y[e_inc];
              en1 = en0;
            }
            else {
              temp_alpha = x_alpha(valley,0);
              m_d = md_xt(valley,x,temperature);
              k1x[e_inc] = k1x[e_inc];
              k1z[e_inc] = k1z[e_inc];
              k1y[e_inc] = sqrt(fabs(2*m_d*en1*(1 + temp_alpha*en1)/H_/H_  - k1x[e_inc]*k1x[e_inc] - k1z[e_inc]*k1z[e_inc]));
            }
    }
    else en1 = en1_temp;
    en1 = fabs(en1);
    kmy = (k0y[e_inc] + k1y[e_inc])/2;
    enm = (k0x[e_inc]*k0x[e_inc]+kmy*kmy+k0z[e_inc]*k0z[e_inc])*H2_2/m_d;
  }
  ii = abs(dround(ry[e_inc]/HY));
  vav[ii] += H_*(k1y[e_inc] + k0y[e_inc])/2/(1 + 2*temp_alpha*enm)/m_c1;
  vavcount[ii] += 1;
}
```

boundary .h

```
void boundary_test_0()
{
  if (ry[e_inc] <= (0*LY)) {
    excount1++;
    extinct_flag = 1;
  }
  if (ry[e_inc] >= (1 - 0)*LY) {
    excount2++;
```

```
    extinct_flag = 1;
  }
  excount = excount1 + excount2;
}

void boundary_test_NB()
{
  realtype r_nb = (float) (NB-1),
   r_ny = (float) NY;
  if (ry[e_inc] <= r_nb*LY/r_ny) {
    excount1++;
    extinct_flag = 1;
  }
  if (ry[e_inc] >= (1 - r_nb/r_ny)*LY) {
    excount2++;
    extinct_flag = 1;
  }
  excount = excount1 + excount2;
}
```

scattering3.h

```
void scatter_rates()
{
  int i,ii,iii,index,valley;
  realtype dummy,temp_alpha,x,m_d,m_c,n_neq1,n_neq2;
  valley = sat[e_inc];
  x = x_fact(ry[e_inc]);
  temp_alpha = x_alpha(valley,x);

  ii = 0;
  if (X_MAX != 0)
    ii = dround(X_INCR*x/X_MAX);
  i = dround(ry[e_inc]/HY);
  if (i > NY) i = NY;
  if (i < 0 ) i = 0;
  doping = doping_profile[i];
  temperature = t_distr[i];
  iii = dround(T_INCR*(temperature - T_MIN)/(T_MAX - T_MIN));
  if (iii > T_INCR) iii = T_INCR;
  if (iii < 0) iii = 0;
  m_d = md_xt(valley,x,temperature);
  m_0 = m_d;
  n_op = 1/(exp(H_K*wop(x)/temperature) - 1);
  if (valley == 0) {
    n_neq1 = 1/(exp(H_K*wcl(x)/temperature) - 1);
    n_neq2 = 1/(exp(H_K*wcx(x)/temperature) - 1);
    n_eq = 1e-15;
  }
  if (valley == 1) {
    n_neq1 = 1/(exp(H_K*wcl(x)/temperature) - 1);
    n_neq2 = 1/(exp(H_K*wlx(x)/temperature) - 1);
```

```
    n_eq = 1/(exp(H_K*wll(x)/temperature) - 1);
  }
  if (valley == 2) {
    n_neq1 = 1/(exp(H_K*wcx(x)/temperature) - 1);
    n_neq2 = 1/(exp(H_K*wlx(x)/temperature) - 1);
    n_eq = 1/(exp(H_K*wxx(x)/temperature) - 1);
  }
  energy = (k1x[e_inc]*k1x[e_inc] + k1y[e_inc]*k1y[e_inc] +
            k1z[e_inc]*k1z[e_inc])*H2_2/m_d;
  if (temp_alpha != 0)
   energy = (sqrt(1 + 4*temp_alpha*energy) - 1)/2/temp_alpha;
  energy = fabs(energy);
  i = dround(TLENGTH*energy/E_MAX);
  if (i > TLENGTH) {
    e_exceed++;
    i = TLENGTH-1;
    ry[E_TOT_MAX_S] = 1208;
  }
  if (valley == 0) {
    rate[1] = scat01[i][ii][iii];
    rate[2] = scat02[i][ii][iii];
    rate[3] = scat03[i][ii][iii];
    rate[4] = scat04[i][ii][iii];
    rate[5] = scat05[i][ii][iii];
    rate[6] = scat06[i][ii][iii];
    rate[7] = scat07[i][ii][iii];
    rate[8] = scat08[i][ii][iii];
    rate[9] = scat09[i][ii][iii];
  }
  if (valley == 1) {
    rate[1] = scat11[i][ii][iii];
    rate[2] = scat12[i][ii][iii];
    rate[3] = scat13[i][ii][iii];
    rate[4] = scat14[i][ii][iii];
    rate[5] = scat15[i][ii][iii];
    rate[6] = scat16[i][ii][iii];
    rate[7] = scat17[i][ii][iii];
    rate[8] = scat18[i][ii][iii];
    rate[9] = scat19[i][ii][iii];
  }
  if (valley == 2) {
    rate[1] = scat21[i][ii][iii];
    rate[2] = scat22[i][ii][iii];
    rate[3] = scat23[i][ii][iii];
    rate[4] = scat24[i][ii][iii];
    rate[5] = scat25[i][ii][iii];
    rate[6] = scat26[i][ii][iii];
    rate[7] = scat27[i][ii][iii];
    rate[8] = scat28[i][ii][iii];
    rate[9] = scat29[i][ii][iii];
  }
  rate[1] = temperature*rate[1];
  rate[2] = n_op*rate[2];
  rate[3] = (n_op + 1)*rate[3];
  rate[4] = n_neq1*rate[4];
  rate[5] = (n_neq1 + 1)*rate[5];
```

```
  rate[6] = n_neq2*rate[6];
  rate[7] = (n_neq2 + 1)*rate[7];
  rate[8] = n_eq*rate[8];
  rate[9] = (n_eq + 1)*rate[9];
  if (doping == 0) rate[10] = 0;
  else {
    dummy = doping + temperature*c_impu*dielec_s(x,temperature)*m_d*energy*(1 + temp_alpha*energy);
    if (valley == 0)
            {
              rate[10] = temperature*temperature*scat010[i][ii][iii]/dummy;
            }
    if (valley == 1)
            rate[10] = temperature*temperature*scat110[i][ii][iii]/dummy;
    if (valley == 2)
            rate[10] = temperature*temperature*scat210[i][ii][iii]/dummy;
  }

  if (rate[10] > 1e14) rate[10] = 1e14;
}

//will selfscatter take place?
void selfscatter()
{
  float temp_alpha,x;
  int i,valley;

  valley = sat[e_inc];
  scatrate = 0;
  for (i = 1; i <= 10; i++)
    scatrate += rate[i];
  if (s_max < scatrate)
    s_max = scatrate;
  if (scatrate < randscat)
    flag[valley][11] = 1;   //selfscatter true
  else
    flag[valley][11] = 0;  //selfscatter false
  if (SCAT_TOT < scatrate) {     //check passed to master
    ry[E_TOT_MAX_S] = 1208;
    scatrate = 0.1*SCAT_TOT;
  }
}

//choice of real scattering mechanism
void scatter_choice()
{
  int i;
  realtype sumrate;
  int valley = sat[e_inc];
  for (i = 1;i <= 10;i++)
    flag[valley][i] = 0;
  sumrate = 0;
  i = 1;
  while (randscat > sumrate)
    sumrate += rate[i++];
  i--;
  flag[valley][i] = 1;
```

```
  flagno = i;
  oldvalley = sat[e_inc];  //current satellite
  sat[e_inc] = sat1[valley][flagno];  //satellite after scatter
}

void update_selfscatter()
{
  k0x[e_inc] = k1x[e_inc];
  k0y[e_inc] = k1y[e_inc];
  k0z[e_inc] = k1z[e_inc];
}

void update_scatter()
{
  realtype dum1,dum2,dum3,dum4,phi,f,cos_teta,sin_teta,cos_alfa,sin_alfa,
    cos_beta,sin_beta,cos_phi,sin_phi,engy1,x,temp_alpha,m_d,a1,a2,a,b,c,d,teta,Cmax,f1,
    Pteta0, Pteta1,eb,b_cw;
  int valley,i,ii,iii;

  valley = sat[e_inc];  //new valley
  x = x_fact(ry[e_inc]);

  i = dround(ry[e_inc]/HY);
  if (i > NY) i = NY;
  if (i < 0 ) i = 0;
  temperature = t_distr[i];
  iii = dround(T_INCR*(temperature - T_MIN)/(T_MAX - T_MIN));
  if (iii > T_INCR) iii = T_INCR;
  if (iii < 0) iii = 0;
  m_d = md_xt(valley,x,temperature);
  ii = 0;
  if (X_MAX != 0)
    ii = dround(X_INCR*x/X_MAX);

  temp_alpha = x_alpha(valley,x);
  engy1 = fabs(energy + delta_e[oldvalley][flagno][ii][iii]);

  // acoustic, non-polar optical phonon (intervalley)
  if (flagno >= 4 && flagno <= 9 || flagno == 1) {
    phi     = 2*PI*randang1;
    cos_teta = 1 - 2*randang2;
    sin_teta = sqrt(fabs(1 - cos_teta*cos_teta));
    k0_mag   = sqrt(m_d*engy1*(1 + temp_alpha*engy1))/H_2;
    k0x[e_inc] = k0_mag*cos(phi)*sin_teta;
    k0y[e_inc] = k0_mag*sin(phi)*sin_teta;
    k0z[e_inc] = k0_mag*cos_teta;
  }

  //polar optical phonon
  if (flagno == 2 || flagno == 3) {
    phi   = 2*PI*randang1;
    dum1  = sqrt(energy*engy1);
    dum2  = (sqrt(energy) - sqrt(engy1))*(sqrt(energy) - sqrt(engy1));
    f     = 2*dum1/dum2;
    dum3  = exp(randang2*log(1 + f + f));
    cos_teta = (1 + f - dum3)/f;
```

```
    sin_teta = sqrt(fabs(1 - cos_teta*cos_teta));
    cos_phi  = cos(phi);
    sin_phi  = sin(phi);
    k0_mag   = sqrt((m_d*engy1*(1 + temp_alpha*engy1)))/H_2;
    k1_mag   = sqrt((m_0*energy*(1 + temp_alpha*energy)))/H_2;
    dum4     = sqrt(k1x[e_inc]*k1x[e_inc] + k1y[e_inc]*k1y[e_inc]);
    sin_alfa = dum4/k1_mag;
    cos_alfa = k1z[e_inc]/k1_mag;
    cos_beta = k1y[e_inc]/dum4;
    sin_beta = k1x[e_inc]/dum4;
    k0x[e_inc] = k0_mag*(cos_beta*sin_teta*cos_phi +
                                    cos_alfa*sin_beta*sin_teta*sin_phi +
                                    sin_alfa*sin_beta*cos_teta);
    k0y[e_inc] = k0_mag*(-sin_beta*sin_teta*cos_phi +
                                    cos_alfa*cos_beta*sin_teta*sin_phi +
                                    sin_alfa*cos_beta*cos_teta);
    k0z[e_inc] = k0_mag*(-sin_alfa*sin_teta*sin_phi +
                                    cos_alfa*cos_teta);
  }

 // impurity
  if (flagno == 10) {
    phi     = 2*PI*randang1;

    //BH approach
    cos_teta = 1 - 2*randang2/(1 + (1 - randang2)*
                                        dielec_s(x,temperature)*c_impu*temperature*m_d*energy*(1 + temp_alpha*energy)/doping);


    sin_teta = sqrt(fabs(1 - cos_teta*cos_teta));
    if (sin_teta < 0)
            charge[0] =  1208;
    cos_phi  = cos(phi);
    sin_phi  = sin(phi);
    k0_mag   = sqrt(m_d*engy1*(1 + temp_alpha*engy1))/H_2;
    k1_mag   = k0_mag;
    dum4     = sqrt(k1x[e_inc]*k1x[e_inc] + k1y[e_inc]*k1y[e_inc]);
    sin_alfa = dum4/k1_mag;
    cos_alfa = k1z[e_inc]/k1_mag;
    cos_beta = k1y[e_inc]/dum4;
    sin_beta = k1x[e_inc]/dum4;
    k0x[e_inc] = k0_mag*(cos_beta*sin_teta*cos_phi +
                                    cos_alfa*sin_beta*sin_teta*sin_phi +
                                    sin_alfa*sin_beta*cos_teta);
    k0y[e_inc] = k0_mag*(-sin_beta*sin_teta*cos_phi +
                                    cos_alfa*cos_beta*sin_teta*sin_phi +
                                    sin_alfa*cos_beta*cos_teta);
    k0z[e_inc] = k0_mag*(-sin_alfa*sin_teta*sin_phi +
                                    cos_alfa*cos_teta);
  }
}
```

transfer.h

```
void transfer_to_old(int txfrom, int txto)
{
  tf_over[txto] = tf_over[txfrom];
  sat[txto]    = sat[txfrom];
  k0x[txto]    = k0x[txfrom];
  k0y[txto]    = k0y[txfrom];
  k0z[txto]    = k0z[txfrom];
  k1x[txto]    = k1x[txfrom];
  k1y[txto]    = k1y[txfrom];
  k1z[txto]    = k1z[txfrom];
  ry[txto]     = ry[txfrom];
}
```

c_assign.h

```
void new_particle(int boundary,int y_gridpoint,int particle_number)
{
  int sign;
  realtype dummy;
  dummy = ((realtype) rand())/RAND_MAX;
  sign = 2*(dround(dummy)) - 1;

  if (boundary == 1) {
    ry[particle_number] = fabs((y_gridpoint + sign*(realtype) rand()/RAND_MAX/2)*HY);
    k_init_gauss_single(1,particle_number);
  }
  else if (boundary == 3) {
    if (y_gridpoint == NY) sign = -abs(sign);
    ry[particle_number] = fabs((y_gridpoint + sign*(realtype) rand()/RAND_MAX/2)*HY);
    if (ry[particle_number] > LY) ry[particle_number] = LY - HY/2;
    k_init_gauss_single(3,particle_number);
  }
  tf_over[particle_number] = 0;
  sat[particle_number] = 0;
}

void charge_assign_()
{
  realtype dum1[11],dum2[11],nd;
  int sign,i,j,i_ex = 0,i_index,p_index,tempnumber = NUMBER_OF_SLAVES;

  dum1[0] = fabs(Q/qs)*(NDOPE1*LX*HY*LZ)/tempnumber/2;
  dum2[NB-1] = fabs(Q/qs)*(NDOPE11*LX*HY*LZ)/tempnumber/2;
  for (i = 1; i <= (NB - 1); i++)
   {
          dum1[i] =  fabs(Q/qs)*(NDOPE1*LX*HY*LZ)/tempnumber;
          dum2[i-1] =  fabs(Q/qs)*(NDOPE11*LX*HY*LZ)/tempnumber;
   }
```

```
for (i = 0; i <= NY; i++)
  charge[i] = 0;

//removal of electrons
for (i = 1; i <= ensemble_tot; i++) {
   i_index = i - i_ex;
   j = dround(ry[i_index]/HY);
   charge[j]++;
   if (j <= (NB-1)) {
            if (charge[j] > (dum1[j])) {
              p_index = ensemble_tot - i_ex;
              transfer_to_old(p_index,i_index);
              i_ex++;
              excount1++;
              charge[j]--;
            }
   }
   else if (j >= (NY - (NB-1))) {
            if (charge[j] > (dum2[j - 1 - NY + NB])) {
              p_index = ensemble_tot - i_ex;
              transfer_to_old(p_index,i_index);
              i_ex++;
              excount2++;
              charge[j]--;
            }
   }
}
ensemble_tot -= i_ex;

//renewal of electrons
for (j = 0; j <= (NB-1); j++) {
   while (charge[j] < (dum1[j])) {
            ensemble_tot++;
            new_particle(1,j,ensemble_tot);
            charge[j]++;
            excount1--;
   }
}
for (j = (NY - (NB-1)); j <= NY; j++) {
   while (charge[j] < (dum2[j - 1 - NY + NB])) {
            ensemble_tot++;
            new_particle(3,j,ensemble_tot);
            charge[j]++;
            excount2--;
   }
}
}
```

vav.h

```
void vav_init()
{
  int i;
  for (i = 0; i <= NY; i++) {
```

```
    vav[i] = 0;
    vavcount[i] = 0;
  }
}

void vav_calc()
{
  int i;
  if (vavcount[0] == 0) vavcount[0] = 1e50;
  vav[0] = vav[0]/vavcount[0];
  for (i = 1; i <= NY; i++) {
    if (vavcount[i] != 0) vav[i] = vav[i]/vavcount[i];
    else vav[i] = vav[i-1];
  }
  for (i = 0; i <= NY; i++)
   vav_av[i] = (vav_av[i]+vav_av[i])/2;
}

void smooth_vav()
{
  int i;
  realtype temp_y,dndy,dn,dy,n1,n0,check,y1,y0,nr;

  for (i = 1; i <= NY; i++)
   vav[i] = (vav[i] + vav[i-1])/2;
}
```

sat_statistics.h

```
void sat_stat()
{
  int i,ii,iii;
  for (i = 0; i <= NY; i++) {
    valley_info[0][i] = 0;
    valley_info[1][i] = 0;
                        valley_info[2][i] = 0;
  }
  for (i = 0; i <= ensemble_tot; i++) {
    ii = sat[i];
    iii = abs(dround(ry[i]/(LY/NY)));
    valley_info[ii][iii] += 1;
  }
}
```

A3 COMMON PROCEDURES: CODE LISTING

global.h

```
#define NY 1500
#define MX 1
#define T_INCR 5
#define E_TOT_MAX_M 70000
#define E_TOT_MAX_S 70000
#define LY1 0.5E-6
#define LY2 0.05E-6
#define LY3 0.01E-6
#define LY4 0.2E-6
#define LY5 1.4E-6
#define LY6 0.5E-6
#define LY7 0.05E-6
#define LY8 0.01E-6
#define LY9 0.2E-6
#define LY10 1.4E-6
#define LY11 1E-6
#define L_ADD 1e-6

const int
  NB = 1,
  TOTAL_M = E_TOT_MAX_M,
  TOTAL_S = E_TOT_MAX_S,
  CMSGTYPE = 1, //msgtype of charge
  VMSGTYPE = 2, //msgtype of velocity
  EMSGTYPE = 3, //msgtype of energy
  SMSGTYPE = 4, //msgtype of initial totals
  TMSGTYPE = 5, //msgtype of temperature
  ST0MSGTYPE = 6, //msgtype of state info: kinetic energy
  ST1MSGTYPE = 7, //msgtype of state info: valley
  VCMSGTYPE = 8,
  VLMSGTYPE = 9,
  VXMSGTYPE = 10,
  NMSGTYPE = 11;

const double
  QSF = 0.9,
  DIELEC_H = 10.82,
  DIELEC_S = 12.53,
  EPS0 = 8.854E-12,
  X_MAX1 = 0.3,
  X_MAX2 = 0.3,
  X_MAX = 0.3, //always maximum of the above
  LX=90e-6, LZ=90e-6,
  H1_START = LY1, H1_FINISH = LY1 + LY2,
  H2_START  = LY1+LY2+LY3+LY4+LY5+LY6,
  H2_FINISH = LY1+LY2+LY3+LY4+LY5+LY6+LY7,
  LY = LY1+LY2+LY3+LY4+LY5+LY6+LY7+LY8+LY9+LY10+LY11,
  NDOPE1 = 1.50e23,NDOPE2 = 0E21,NDOPE3 = 1E24,NDOPE4 = 0E22,
  NDOPE5 = 1.2E22,NDOPE6 = 1.75E23,NDOPE7 = 0e22,NDOPE8 = 1E24,
```

```
 NDOPE9 = 0E21,NDOPE10 = 1.2E22,NDOPE11 = 1.50E23,
 NDOPE_FACTOR = 1.54,
 G_FACTOR = 0.25,
 NDOPE1_FACTOR = 0.999,
 G1_FACTOR = 0.999,
 TAMB = 300,
 TAMB_0 = 300,
 T_INIT = 450,
 T_MIN = 350,
 T_MAX = 550,
 LASTKEY = 12.08,
 TSTEP = 5E-15,
 TSIM = 950E-12;
#define PROG_SLAVE "lfslave"
```

g_param.h

```
realtype x_fact(float ly) {
   float temp_x = 0;
            if (ly <= H1_FINISH && ly > H1_START)
                                temp_x = X_MAX1*(ly - H1_START)/(H1_FINISH - H1_START);
            if (ly <= H2_FINISH && ly > H2_START)
                                temp_x = X_MAX2*(ly - H2_START)/(H2_FINISH - H2_START);
            return(temp_x);
}

realtype x_fact_old(float ly) {
   float temp_x = 0;
            if (ly <= H1_FINISH && ly > H1_START)
                                temp_x = X_MAX1*(ly - H1_START)/(H1_FINISH - H1_START);
            if (ly <= H2_FINISH && ly > H2_START)
                                temp_x = X_MAX2*(ly - H2_START)/(H2_FINISH - H2_START);
            return(temp_x);
}

realtype p_fact(float ly) {
   float temp_p = 0;
            if (ly <= H1_FINISH && ly > H1_START)
                                temp_p = p1;
            if (ly <= H2_FINISH && ly > H2_START)
                                temp_p = p2;
            return(temp_p);
}

realtype dielec_s(realtype x, realtype T) {
            return(DIELEC_S*(1 + 1.2e-4*T) - 3.12*x);
}


realtype dielec_h(realtype x, realtype T) {
            return(DIELEC_H*(1 + 9e-5*T) - 2.73*x);
}
```

mathadd.h

```
int dround(double xx)
{
        int yy;
        yy = (int)xx;
        if (yy + 0.5 <= xx) yy = yy + 1;
        return yy;
}
```