2D SIMULATION OF ELECTRONIC TRANSPORT IN GaAs MESFET BY ENSEMBLE MONTE CARLO METHOD

A DISSERTATION
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
NANO SCIENCE AND TECHNOLOGY

Submitted by

Deepak Kumar Baghel

2K13/NST/08

Under the supervision of

Dr. Rinku Sharma
(Associate Professor)

DEPARTMENT OF APPLIED PHYSICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

2015

CERTIFICATE

This is to certify that the dissertation title **"2D simulation of electronic transport in GaAs MESFET by ensemble Monte Carlo method"** submitted by **Mr Deepak Kumar Baghel**, Roll. No. 2K13/NST/08, in partial fulfilment for the award of degree of Master of Technology in Nano Science and Technology at **Delhi Technological University, Delhi**, is a bonafide record of student's own work carried out by him under my supervision and guidance in the academic session 2014-15. The matter embodied in dissertation has not been submitted for the award of any other degree or certificate in this or any other university or institute.

Dr. Rinku Sharma
(Supervisor)
Associate Professor
Department of Applied Physics
DTU, New Delhi

Dr.S.C.Sharma
(HOD)
Professor
Department of Applied Physics
DTU, New Delhi

CANDIDATE DECLARATION

I, Deepak Kumar Baghel, Roll No.-2K13/NST/08 student of M. Tech. (NANO SCIENCE AND TECHNOLOGY), hereby declare that the dissertation/project titled **"2D Simulation of electronic transport in GaAs MESFET by ensemble Monte Carlo method"** under the supervision of Dr. Rinku Sharma, Department of Applied Physics, Delhi Technological University, in partial fulfilment of the requirement for the award of the degree of Master of Technology has not been submitted elsewhere for the award of any Degree.

( )

Place: Delhi
Date: 30.06.2015

Deepak Kumar Baghel
M.Tech (NST)
2K13/NST/08
Dept of Applied Physics
DTU, New Delhi

ACKNOWLEDGEMENT

I am indebted to my thesis supervisor **Dr.Rinku Sharma,** Department of Applied Physics, for her gracious encouragement and very valued constructive criticism that has driven me to carry the project successfully.

I am greatly thankful to **Dr.S.C.Sharma,** Head of Department (Appled Physics), entire faculty and staff of Applied Physics Department and friends for their, continuous support, encouragement and inspiration in the execution of this "**thesis"** work.

Finally I express my deep sense of gratitude to my parents who bestowed upon me their grace and were source of my inspiration and encouragement.

Deepak Kumar Baghel
M.Tech (NST)
2K13/NST/08
Dept. of Applied Physics
DTU, New Delhi

Table of contents

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

I have performed the simulation of the electronic transport of GaAs MESFET by ensemble Monte Carlo method using Archimedes package and GUI plot in 2-D space. MESFETs are usually constructed in compound semiconductor technologies lacking high quality surface passivation such as GaAs, InP, or SiC, and are faster but more expensive than silicon-based JFETs or MOSFETs. Production MESFETs are operated up to approximately 45 GHz and are commonly used for microwave frequency communications and radar. They are quite similar to a JFET in construction and terminology. The difference is that instead of using a p-n junction for a gate, a Schottky (metal-semiconductor) junction is used.

The Ensemble Monte Carlo method is the method that Archimedes uses to simulate and predict the behavior of a devices. Being the Monte Carlo very stable and reliable, Archimedes can be used to know the characteristics of a device even before this is built. The physics and geometry of a device is described simply by a script, which makes, in this sense, Archimedes a powerful tool for the simulation of quite general semiconductor devices.

CHAPTER 1

INTRODUCTION

The number of semiconductor materials for MESFET that are today studied and employed by the electronic industry is very large and continuously increasing, in particular after the introduction of semiconductor hetero structures. For space reasons, we use two most “popular” materials, namely silicon and gallium arsenide. Silicon is by far the most used material in semiconductor industry, both because of its large availability and because of the existence of a “natural oxide”, very suitable for the realization of electronic devices. Gallium arsenide, on the contrary, is much more convenient for optoelectronic applications, owing to its direct energy gap, appropriate for a transformation between electronic and optical energies. Furthermore, its small electron effective mass (0.067 compared to an average 0.295 in Si) provides a higher electron mobility [1].

1.1 GaAs MATERIAL PROPERTIES

GaAs is an III–V compound semiconductor composed of the element gallium (Ga) from column III and the element arsenic (As) from column V of the periodic table of the elements. GaAs was first created by Goldschmidt and reported in 1929, but the first reported electronic properties of III–V compounds as semiconductors did not appear until 1952[3].

The GaAs crystal is composed of two sub lattices, each face centered cubic (fcc) and offset with respect to each other by half the diagonal of the fcc cube. This crystal configuration is known as cubic sphalerite or zinc blende. Figure shows a unit cube for GaAs and Table provides a listing of some of the general material characteristics and properties [3].

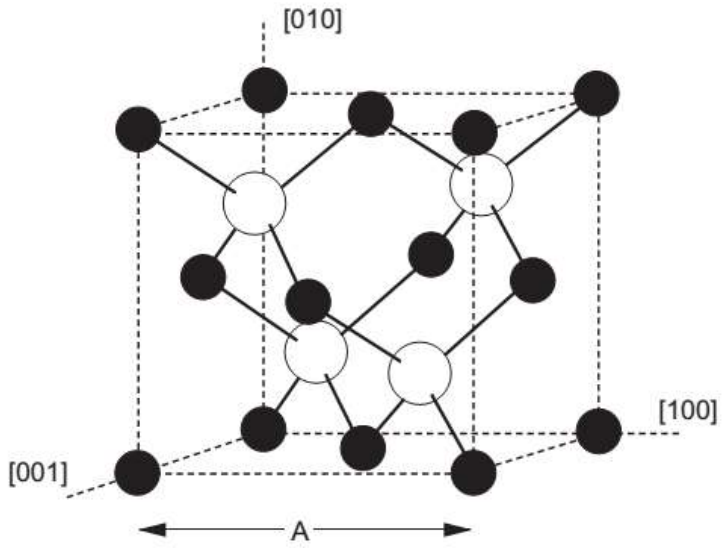

Fig 1.1 Unit cube of GaAs crystal lattice

Table 1.1 Properties of GaAs at room temperature

Property	Parameter
Crystal structure	Zinc blende
Lattice constant	5.65 Å
Density	5.32 g/cm^3
Atomic density	4.5×10^{22} atoms/cm^3
Molecular weight	144.64
Bulk modulus	7.55×10^{11} dyn/cm^2
Sheer modulus	3.26×10^{11} dyn/cm^2
Coefficient of thermal expansion	5.8×10^{-6} K^{-1}
Specific heat	0.327 J/g-K
Lattice thermal conductivity	0.55 W/cm-°C
Dielectric constant	12.85
Band gap	1.42 eV
Threshold field	3.3 kV/cm
Peak drift velocity	2.1×10^7 cm/s
Electron mobility (undoped)	8500 cm^2/V-s
Hole mobility (undoped)	400 cm^2/V-s
Melting point	1238°C

1.1.1 ENERGY BAND STRUCTURE

As a result of the laws of quantum mechanics, electrons in isolated atoms can have only certain discrete energy values. As these isolated atoms are brought together to form a crystal, the electrons become restricted not to single energy levels, but rather to ranges of allowed energies, or bands called the valance and conduction bands (Figure 1.2). These two bands are separated by an energy band gap, which is a very important characteristic of the semiconductor material. At zero kelvin, all the electrons are confined to the valance band and the material is a perfect insulator. Above zero kelvin, some electrons have sufficient thermal energy to make a transition to the conduction band where they are free to move and conduct current through the crystal. The probability of an electron having enough energy to make the transition is given by the Fermi distribution function. The Fermi level shown on Figure 1.2 is the energy level at which the probability function is equal to one half. For pure semiconductors, the Fermi level is approximately in the center of the band gap. Note, though, that no electron actually has an energy of *EF*, since they are not permitted to exist at energies in the band gap [3].

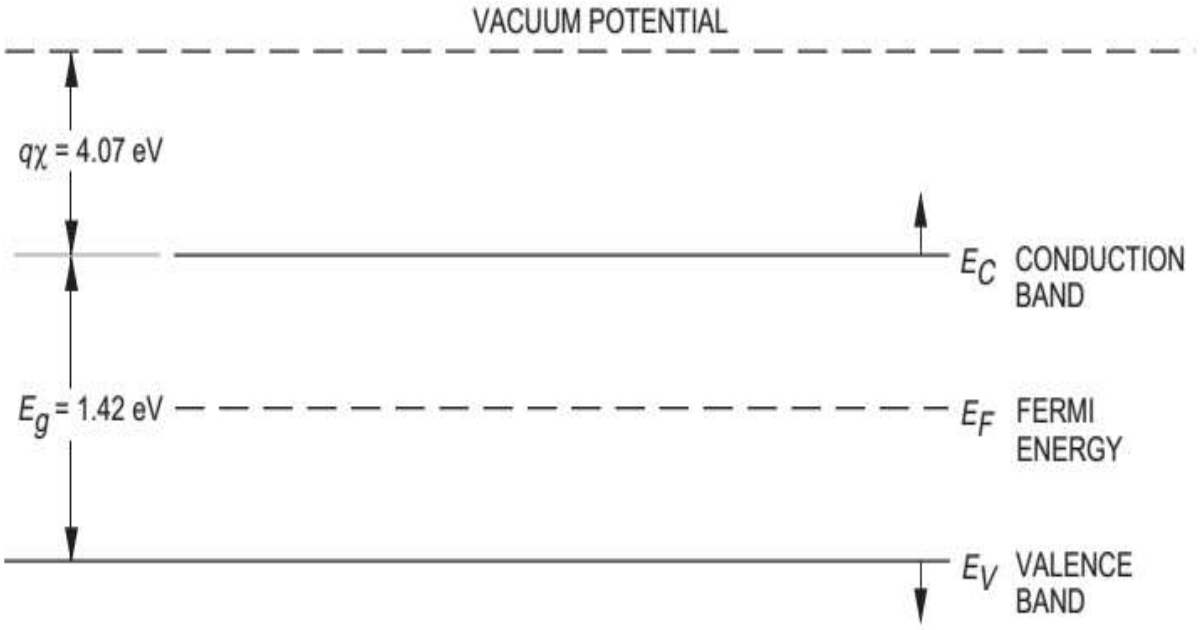

Fig 1.2 Energy band diagram of GaAs

The amount of energy required for an electron to move from the valance band to the conduction band (energy band gap) depends on the temperature, the semiconductor material, and the material's purity and doping profile. For undoped GaAs, the energy band gap at room temperature is 1.42 eV. The energy band diagram is usually referenced to a potential called the vacuum potential. The electron affinity, $q\chi$, is the energy required to remove an electron from the bottom of the conduction band to the vacuum potential. For GaAs, $q\chi$ is approximately 4.07 eV.

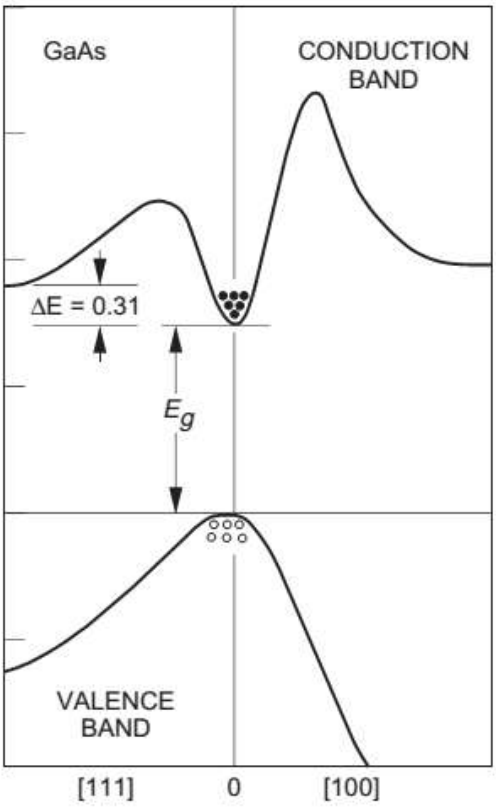

Fig 1.3 Energy band structure of GaAs

GaAs is a direct band gap semiconductor, which means that the minimum of the conduction band is directly over the maximum of the valance band (Figure 1.3). Transitions between the valance band and the conduction band require only a change in energy, and no change in momentum, unlike indirect band-gap semiconductors such as silicon (Si). This property makes GaAs a very useful material for the manufacture of light emitting diodes and semiconductor lasers, since a photon is emitted when an electron changes energy levels from the conduction band to the valance band [2] [25] .

1.1.2 MOBILITY AND DRIFT VELOCITY

GaAs has several advantages over silicon for operation in the microwave region—primarily, higher mobility and saturated drift velocity and the capability to produce devices on a semi-insulating substrate.

In a semiconductor, when a carrier (an electron) is subjected to an electric field, it will experience a force (F = –*q*E) and will be accelerated along the field. During the time between collisions with other carrier ions and the semiconductor lattice, the carrier will achieve a velocity that is a function of the electric field strength. This velocity is defined as the drift velocity (*v*). From the conservation of momentum, it can be shown that the drift velocity (*v*) is proportional to the applied electric field (Figure 1.4) and can be expressed as

$$v = -\left(\frac{q\tau_c}{m^*}\right)\mathbf{E} \tag{1.1}$$

The proportionality factor depends on the mean free time between collisions (τc) and the electron effective mass (*m**). The proportionality factor is called the electron mobility (μ) in units of cm2/V-s.

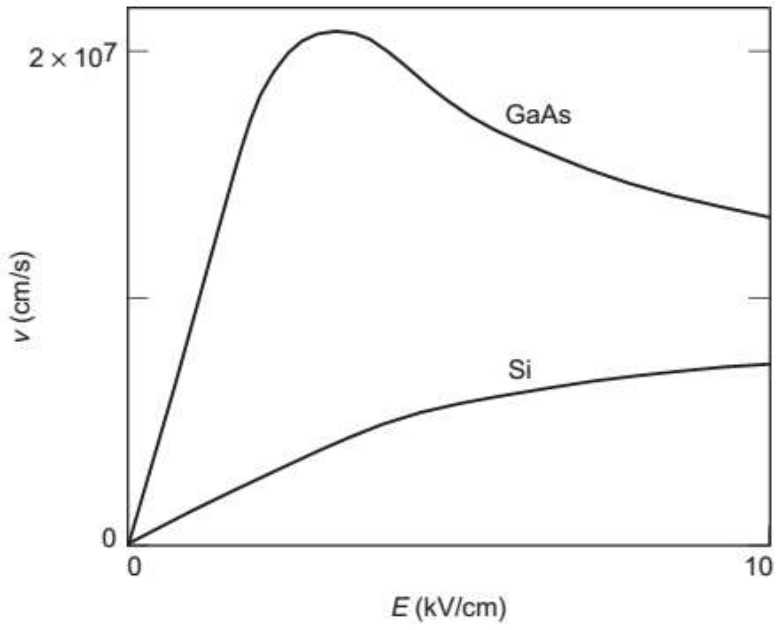

Fig 1.4 drift velocity of electron in Si and GaAs as a function of E field

Mobility is an important parameter for carrier transport because it describes how strongly the motion of an electron is influenced by an applied electric field. From the equation (1.1) above, it is evident that mobility is related directly to the mean free time between collisions, which in turn is determined primarily by lattice scattering and impurity scattering. Lattice scattering, which is a result of thermal vibrations of the lattice, increases with temperature and becomes dominant at high temperatures; therefore, the mobility decreases with increasing temperature. Impurity scattering on the other hand, which is a result of the movement of a carrier past an ionized dopant impurity, becomes less significant at higher temperatures .Although the peak mobility of GaAs in the linear region can be as much as six times greater than that of silicon (Si) at typical field strengths, the advantage of GaAs may be only as much as a factor of two. This still translates to the fact that GaAs devices can work at significantly higher frequencies than Si. The exact increase in the speed of operation depends on factors such as the circuit capacitance and the electric field regime in which the device operates [3].

1.1.3 SEMI-INSULATING GaAs

The importance of semi-insulating GaAs is based on the fact that devices made of it by direct ion implantation are self-isolating, so that it is ideally suited to integrated circuit fabrication. Moreover, the semi-insulating substrate provides greatly reduced parasitic capacitances, thus faster devices, and allows for integration and the implementation of monolithic microwave integrated circuits (MMIC)

Semi-insulating GaAs must meet the following requirements to provide semiconductor quality material:

(1) Thermal stability during epitaxial growth or anneal of ion-implanted active layer.

(2) Absence of undesirable substrate active layer interface effects, such as back-gating and light sensitivity.

(3) No degradation of active layer properties by outdiffusion of impurities from substrate during thermal processing.

(4) Lowest possible density of crystalline defects, such as dislocations, stacking faults, and precipitates.

1.2 GaAs MESFET

The MESFET is a form of semiconductor technology which is very similar to a junction FET or JFET. As the name of the MESFET indicates, it has a metal contact directly onto the silicon, and this forms a Schottky barrier diode junction. The material that is used can be silicon or other forms of semiconductor. However the material that is most widely used is gallium arsenide GaAs. Gallium arsenide is normally chosen because of the very superior electron mobility it provides that enables superior high frequency operation to be achieved. The substrate for the semiconductor device is semi-insulating for low parasitic capacitance, and then the active layer is deposited epitaxially. The resulting channel is typically less than 0.2 microns thick [35].

The doping profile is normally non-uniform in a direction perpendicular to the gate. This makes for a device which has good linearity and low noise. Most devices are required for high speed operation, and therefore an n-channel is used because electrons have a much greater mobility than holes that would be present in a p-channel. The gate contacts can be made from a variety of materials including Aluminum, a Titanium-Platinum-Gold layered structure, Platinum itself, or Tungsten. These provide a high barrier height and this in turn reduces the leakage current. This is particularly important for enhancement mode devices which require a forward biased junction. The gate length to depth ratio is an important as this determines a number of the performance parameters. Typically it is kept at around four as there is a trade-off between parasitic, speed, and short channel effects. The source and drain regions are formed by ion-implantation. The drain contacts for GaAs MESFETs are normally AuGe - a Gold-Germanium alloy [8].

There are two main structures that are used for MESFETs:

1. Non self-aligned source and drain:

For this form of MESFET, the gate is placed on a section of the channel. The gate contact does not cover the whole of the length of the channel. This arises because the source and drain contacts are normally formed before the gate.

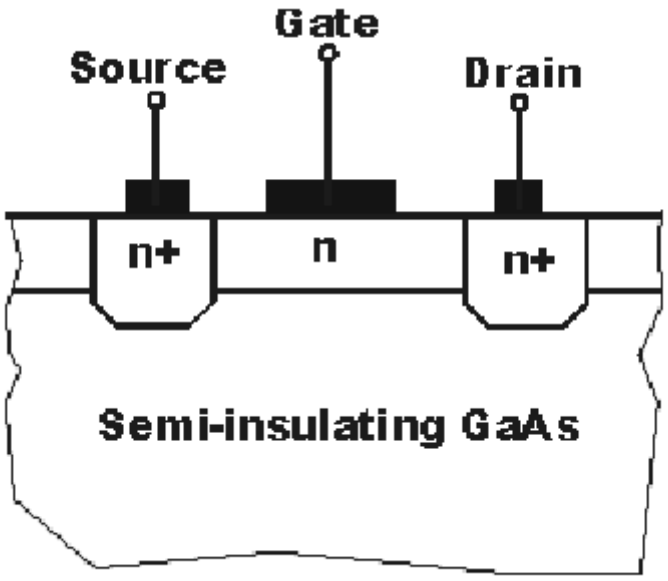

Fig1.5 Non self-aligned MESFET structure

2. self-aligned source and drain:

This form of structure reduces the length of the channel and the gate contact covers the whole length. This can be done because the gate is formed first, but in order that the annealing process required after the formation of the source and drain areas by ion implantation, the gate contact must be able to withstand the high temperatures and this results in the use of a limited number of materials being suitable [35].

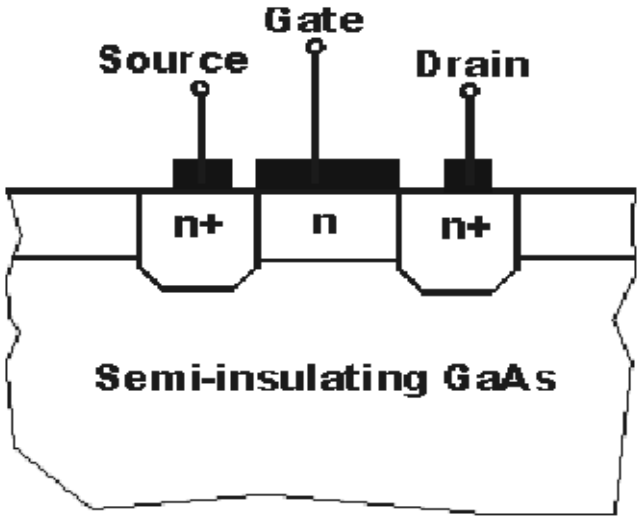

Fig1.6 self-aligned MESFET structure

1.3 ARCHIMEDES PACKAGE OVERVIEW

The GNU package Archimedes is a 2D Quantum Monte Carlo simulator for semiconductor devices. At the present time it can simulate the transport of electrons and holes in Silicon, Gallium Arsenide, Germanium, InSb, AlSb, AlAs, InP, GaP and the two compounds AlxInxSb and AlxIn(1−x)Sb,(actually the purpose is reaching the possibility of simulating a quite big range of materials belonging to the cubic group IV of the diamond structure and to the III-V semiconductors of the zinc blende structure along with all the heterostructure possible) [32].

Archimedes uses the well-known ensemble Monte Carlo method for the simulations. It can simulate both the transient and the steady state solution (even if the transient can be quite noisy, due to the statistical approach). The particles dynamics is coupled to the electrostatic potential by means of the simulation of a non-stationary Poisson equation. This last equation is simulated by a simple, but very robust, finite difference method. In this present version of **Archimedes** we can choose the physics of the various contacts present on the device. So, for example, you can decide if an edge (or a part of it) is an insulator, or a Schottky contact or even an Ohmic one. In addition, the quantum effects are taken into account by means of the recent effective potential method, which is starting to be used by the academic community, as you can see from scientific papers [24].

Furthermore we can simulate a simplified MEP (Maximum Entropy Principle) model which is very useful for making Archimedes faster than the precedent release. Now we can simulate even a fixed constant magnetic field and/or the self-consistent magnetic field by means of the Faraday's equation. This is a quite rare feature in semiconductor simulator that Archimedes is already able to implement.

All the particles in this code have a 'statistical weight' which is made a piecewise-function of the position. We can choose the number of particle used in the simulation, even if this last will vary during the simulation, but it is not allowed to be more than 10 million. If we want a bigger number we have to change it in the code (modifying the definition of NPMAXIMUM in the file"archimedes.c" and recompiling it). I have chosen to not dynamically allocate the memory because the number of particles in the devices can vary very rapidly (depending on the device

structure, obviously) and this can enormously tax the velocity of the simulation, which is very undesirable in a Monte Carlo simulation.

These are only some keywords (or commands) we can use in **Archimedes** to describe the geometry and the physical characteristics of a simulated device. As we will see, they are simple to understand and very general. So it is easy to define a device with quite general characteristics [32].

CHAPTER 2

Physical Models Used in Archimedes

Describing the physical models used in Archimedes, It is very important to understand this chapter in order to fully exploit the possibility offered by this code. Obviously, everything here is a brief review of what we can find in papers and books on the Monte Carlo subject. We will, also, describe the simplified MEP (Maximum Entropy Principle) model, which is a simplified version of the MEP model. The professors of the Department of Mathematics and Computer Sciences of the University of Catania who came up with this model were A.M.Anile and V.Romano[]. The accuracy of this model is very high when compared to the other hydrodynamical models of its kind.

The coupling of MEP model and Monte Carlo method have been used in order to obtain very accurate simulation results in very short running times.

2.1 THE SEMI CLASSICAL APPROACH

In semi classical approach we are discussing about a well-defined group of approximations.

1. **Quantum size effects**. Firstly, the dimensions of the device simulated have to be such that the envelope wavelength of the carriers (in our case electrons or holes) are negligible with respect to the characteristic length of the device. In that case, the particles can be described by wave-packets well-localized in the phase-space. In this case, we can consider the particles as ”billiard balls”.

2. **Slow Physical Phenomena**. The dynamics of electrons or holes are significantly slow with respect to the dynamics of the electric and/or magnetic field. So we have worked in a physical context in which it is justified to use electrostatic rather than full set of Maxwell’s equations. Thus we have simulated only the Poisson equation, neglecting the potential retardation effects and the coupling with the photons.

3. **The effective mass approximation**. It is well-known, both from the quantum theory of matter and from physical experiments, that a particle moving in a periodic potential, as the potential experienced by a particle in a lattice, can be described as a free particle with a mass lightly smaller than the original one. Then if an electron move in a semiconductor lattice, its mass will be smaller by a well-defined factor. This is the approximation have been used in order to take into account the effects of the lattice on the particles.

4. **The scattering events**. The scattering are considered as semi classical, i.e. they are obtained from quantum theory of scatterings, but the scattering events are considered instantaneous, uncorrelated and localized in space and time.

5. **The Many Body effects**. In our simulations, we have neglected the Pauli principle as a result of which all the particles in the simulation have interaction with each other. Even if it is evident, from experiments, that in the real world and for enough diluted doping concentrations, the electrons don't interact with other electrons, hence there are no collisions between electrons.

2.2 THE QUANTUM EFFECTS

Concerning the quantum effects, while we consider the particles as semi classical objects, we want to have the possibility of simulating relevant quantum effects in recently manufactured semiconductor devices (like the diode tunnel, or the nanoscopic MOSFETs). This is something hard to achieve, because we need an accurate solution of the Wigner equation to simulate these effects correctly. Unfortunately, the solution of the Wigner equation is a very difficult challenge, both from the point of view of numerical analysis and the point of view of computer resources, because it is an integro-differential equation, with a non-local term for the potential, which is very difficult to solve numerically, even in the one dimensional case and the solution of such an equation is a function of the phase-space variables and time, which means that it is an enormously expensive solution from the point of view of computer memory. So, we have to use another approach than the Wigner equation one [14]. For this purpose, recently, a new interpretation of quantum mechanics has been presented which is, at least at first order, equivalent to the Wigner quantum approach (and to the density gradient approach): this approach is known as @sampthe effective potential method. While in the Wigner, but even in

the Schrodinger, quantum theory we consider the particles as wave-like objects (with very strange and unphysical properties, like negative probability, or non-locality...), in the effective potential we keep on considering particles as well-positioned particles in the phase-space, which is actually what we experience in the real world. So, instead of giving a new definition of particles we redefine the electrostatic potential. In order to do it we compute the classical electrostatic potential by means of the classical and widely used Poisson equation

$$\nabla \cdot [\varrho(x)\nabla\phi cl(x, t)] = -q[ND(x) - NA(x) - n(x, t) + p(x, t)] \quad (2.1)$$

where ∇ is the gradient operator, ϱ the material dielectric constant, ND and NA the donor and acceptor densities respectively, q the elementary charge, n and p the electron and hole densities respectively. Then we transform the precedent obtained classical potential in a quantum one in the following fashion

$$\phi_{quant} = \frac{1}{\sqrt{2\pi}a}\int_{\mathbb{R}^n} \phi_{cl}(\mathbf{x}+\xi, t)exp(-\frac{\xi^2}{2a^2})d\xi \quad (2.2)$$

where n is the dimension of the spatial space (here $n = 2$), $a = \frac{\hbar}{\sqrt{8m^* k_B T_L}}$, being ~ the Planck constant divided by 2π, m∗ is the precedent discussed effective mass, k_B the Boltzmann constant and T_L the lattice temperature.

2.3 THE PARTICLE DYNAMICS

The purpose of this simulation is to solve the Boltzmann or the Wigner equation including the most accurate physical models, i.e. one of the following two equations (depending on including or not the quantum effects).

$$\frac{\partial f}{\partial t} + \frac{1}{\hbar}\nabla_{\mathbf{k}}\mathcal{E}\cdot\nabla_{\mathbf{x}}f - \frac{q}{\hbar}\mathbf{E}\cdot\nabla_{\mathbf{k}}f = \mathcal{C}[f](\mathbf{k},\mathbf{x},t) \quad (2.3)$$

$$\frac{\partial w}{\partial t} + \frac{\hbar\mathbf{k}}{m^*}\nabla_{\mathbf{x}}w\cdot\nabla_{\mathbf{x}}w - \frac{1}{2\pi\hbar}\int_{\mathbb{R}} d\mathbf{k}' V^W(\mathbf{k}-\mathbf{k}',\mathbf{x})w(\mathbf{k}',\mathbf{x},t) = \mathcal{C}^W[w](\mathbf{k},\mathbf{x},t) \quad (2.4)$$

$$V^W(\mathbf{k},\mathbf{x}) = i\int_{\mathbb{R}} d\eta exp(i\eta\cdot\mathbf{k})(V(\mathbf{k},\mathbf{x}+\frac{\eta}{2}) - V(\mathbf{k},\mathbf{x}-\frac{\eta}{2})) \quad (2.5)$$

where $f = f(k,x,t)$ is the Boltzmann probability density function, $w = w(k,x,t)$ the Wigner probability density function, k the particle pseudo-wave vector, x the position vector, ı the imaginary unity, $V(x,t)$ the classical electrostatic potential, $E = -\nabla\varphi$ the classical electric field, $E = E(k)$ the energy band relation. The operators $C[f]$ and $C^W[w]$ are the collision kernel for the Boltzmann and the Wigner equation respectively. Let us note that both the collision terms are numerically very difficult to simulate (as we will see in the mathematical expression of them) and it has a not very mathematically clear expression (at the present time) for the Wigner equation. These equations have to be simulated in order to get accurate and predictive results [6] [7].

In this section we report the basic models used in our simulations, in order to compute the solution of the two precedent equations.

2.3.1 The Band Structure

It is well-known from the crystallography that crystals can be described in terms of Bravais lattices, which means, physically, that the crystal lattice can be though as a periodic potential made of ions. The quantum mechanical dynamics of an electron in a periodic potential can be described by the following well-known Bloch's theorem. Theorem.

Let us consider an electron whose motion is governed by the potential VL generated by the ions located at the points of the crystal lattice L. The Schrodinger equation is

$$H\psi = E\psi \tag{2.6}$$

With the Hamiltonian H given by

$$H = -\frac{\hbar^2}{2m}\nabla^2 - qV_L$$

Then, this theorem states that the bounded Eigen states have the following form

$$\psi(\mathbf{x}) = \exp(\imath\mathbf{k}\cdot\mathbf{x})\mathbf{u_k}(\mathbf{x}) \tag{2.7}$$

And

$$u_k(\mathbf{x} + \mathbf{X}) = \mathbf{u_k}(\mathbf{x}) \tag{2.8}$$

With x belonging to L. Furthermore, it is possible to prove the existence of an infinite sequence of Eigen pairs of solutions

$$E_l(k),\ u_{k,l}$$

With *l* belonging to the non-negative integers set N. The function $E = E_l(\mathbf{k})$ describes the *l*-th energy band of the crystal.

The energy band of crystals can be obtained at the cost of intensive numerical calculations by the quantum theory of solids. However, in order to describe electron and hole transport, for most applications, a simplified description is adopted which is based on simple analytical models. These are the effective mass approximation and the Kane dispersion relation, which are used in this simulations. In the approximation of the Kane dispersion relation, which takes into account the non parabolicity at high energy, the energy still depends only on the modulus of the pseudo-wave vector, but we have the following relation

$$\mathcal{E}(k)[1+\alpha\mathcal{E}(k)] = \frac{\hbar^2 k^2}{2m^*} \tag{2.9}$$

Where α is the non-parabolicity parameter.

The Kane dispersion relation is the best choice if we consider the accuracy of the electron energy and velocity along with velocity of computation [4].

2.3.2 The Drift Process

An electron moving in a crystal lattice moves just like a free electron, but with a change of mass. We have used the classical equations of motion in order to describe the motion of electrons and holes in a semiconductor device [6]. Thus Hamilton formalism have been used to get the electron equations of motion which are given below:

$$\frac{dx}{dt} = \frac{1}{h}\nabla_k H \tag{2.10}$$

$$\frac{dk}{dt} = -\frac{1}{h}\nabla_x H \tag{2.11}$$

Where *H* is the Hamiltonian of the system, i.e.

$$H = E(k) + V(x)$$

Finally, by using the Kane dispersion relation, the expression for the electron velocity is given below:

$$\mathbf{v} = \frac{\hbar \mathbf{k}}{m^*} \frac{1}{\sqrt{1 + 4\alpha \frac{\hbar^2 k^2}{2m^*}}} \tag{2.12}$$

2.4 INITIAL CONDITIONS

In this paragraph, we explain how we have specified the initial conditions for the super-particles. Concerning the spatial distribution, this is trivially done according to the donor (resp. acceptor) profile density specified by the user in the input file for the electrons (resp. holes). Concerning the distribution in the pseudo-wave vector space, things are a little bit more complex. We have to specify an initial particle distribution in the k-space. This is done in the following way. We have considered all the particles at the initial time of the simulation, nearly the thermal equilibrium, which means that the energy of a particle reads

$$\mathcal{E}(k) = -\frac{3}{2} k_B T_L ln(r) \tag{2.13}$$

Where r is a random number between 0 and 1.

Once we have specified the energy of the electrons, then we have chosen the pseudo-wave vectors of all particles which is done by the following algorithm.

1. The modulus of the pseudo-wave vector is computed from the Kane dispersion relation, which is given by the below expression

$$k = \frac{\sqrt{2m^*\mathcal{E}(k)[1+\alpha\mathcal{E}(k)]}}{\hbar} \tag{2.14}$$

2. Then we have generated two random numbers between 0 and 1, say θ and ϕ.

3. The three component of the pseudo-wave vector is computed as

$$kx = k \sin\theta \cos\phi \tag{2.15}$$

$$Ky = k \sin\theta \sin\phi \tag{2.16}$$

$$Kz = k \sin\theta \tag{2.17}$$

2.5 CONTACTS AND BOUNDARIES

We have divided the contacts into three categories i.e. Insulator boundary, Ohmic and Schottky contacts. We have imagined a contact or a boundary as a line on an edge of the device. There is no limitation in using the number of contacts.

2.5.1 Insulator Boundaries

This kind of boundary is also called as a “mirror boundary” in which a particle when interact with such a contact will be simply reflected by this one. This is necessary to simulate the insulator boundaries of a device. We can also apply a non-zero potential on a boundary even if this is an insulator edge.

2.5.2 Ohmic Contacts

Ohmic contacts are open contacts in which particles can either go out or enter in the device through it, but hold the neutrality charge condition, i.e. the charge have to be assigned constant on it. Thus Ohmic contacts behave as electron reservoirs from which the particles can go out from the device.

2.5.3 Schottky Contacts

Schottky contacts are also open contacts but they don't have an electron reservoir, which means that electrons (or particles) can go out through it, but they are only absorbing contacts, which means neutrality charge condition have to hold on it [15].

2.6 THE SCATTERING PROCESS

1. Self-Scattering-This scattering in used to find the flight time. It is need to compute this scattering accurately, as it influences all process during the simulation. Let us report, how the self-scattering is introduced in the simulation. If the various scatterings read

$$W_i(E(k))$$

for i = 1,2, ..., N, where N is the number of the scatterings taken into account in the simulation, then we define the following variable Γ as follows

$$\Gamma = \sum_{i=0}^{N} \mathcal{W}_i(\mathcal{E}(k)) \tag{2.18}$$

Then the free flight τ of a particle will read

$$\tau = -\frac{ln(r)}{\Gamma} \tag{2.19}$$

Where *r* is a random number between 0 and 1. The factor Γ will be used to determine when the self-scattering occurs which has been discussed below.

2. Elastic Acoustic Phonon Scattering. From quantum mechanics, applying the Fermi's golden rule and some other approximations, it is possible to show that the probability that an electron with a starting pseudo-wave vector k scatters with an elastic acoustic phonon and having a final pseudo-wave vector k', is

$$S(\mathbf{k}, \mathbf{k}') = \frac{\pi \Xi^2 k_B T_L}{\hbar c_L \Omega} \frac{k}{q_w \mathcal{E}(k)} \delta\left(\frac{q_w}{2k} \pm \cos\theta'\right) \tag{2.20}$$

Where Ξ is a proportionality constant called deformation potential, c_L the elastic constant of the material, ϑ' the polar angle between the two vectors k and k′, q_w the modulus of the phonon wave vector and Ω the volume of the crystal. Now integrating on k′ one can easily obtain the probability that an electron of energy E scatters with an acoustic phonon. This last reads

$$\mathcal{W}(\mathbf{k}) = \frac{2\pi\Xi^2 k_B T_L}{\hbar c_L} N(\mathcal{E}_k) \tag{2.21}$$

Where *N* (E) is the density of states and reads

$$N(\mathcal{E}(k)) = \frac{(2m^*)^{\frac{3}{2}}\sqrt{\mathcal{E}(k)}}{4\pi^2\hbar^3} \tag{2.22}$$

3. Non-Polar Optical Phonon Scattering. Concerning the non-polar optical phonon, following the same rules as before we get the two probabilities

$$S(\mathbf{k},\mathbf{k}') = \frac{\pi D_{opt}^2}{\rho\omega_0\Omega}(n_0 + \frac{1}{2} \mp \frac{1}{2})\delta(\frac{\hbar^2 q_w^2}{2m^*} \pm \frac{\hbar^2 k q_w \cos\theta'}{m^*} \mp \hbar\omega_0) \tag{2.23}$$

$$\mathcal{W}(\mathbf{k}) = \frac{\pi D_{opt}^2}{\rho\omega_0}(n_0 + \frac{1}{2} \mp \frac{1}{2})N(\mathcal{E}(k) \pm \hbar\omega_0) \tag{2.24}$$

Where D_{opt} is the optical deformation potential constant, ω_0 the phonon angular frequency, n_0 a value almost equal to the intrinsic density of the material.

2.6.1 The Choice of the Scattering

The choice of the scattering is quite simple. First of all, we select randomly a scattering process and after this has been done, we compute the particle state after the scattering event. To this purpose we define the following functions

$$\Lambda_i(\mathcal{E}) = \frac{\sum_{j=1}^{n} \mathcal{W}_j(\mathcal{E})}{\Gamma} \tag{2.25}$$

For *i* = 1,2,...,*N* where *N* is, as before, the number of scattering taken into account during the simulation. A scattering mechanism is, then, chosen generating a number *r* lying between 0 and 1 and doing the following comparison

$$\Lambda_{i-1}(\mathcal{E}) < r <= \Lambda_i(\mathcal{E}) \tag{2.26}$$

For a particle with energy $\mathcal{E}$.

2.7 THE SIMPLIFIED MEP MODEL

From the vast literature, it is known that Monte Carlo method is the best way for obtaining very accurate simulations for the transport of electrons in semiconductor devices. Although this method is highly accurate, but simulation time is very high. Thus we have the possibility of coupling MEP and Monte Carlo in order to make Monte Carlo method faster.

The MEP model is a very advanced hydro dynamical model for both electrons and holes in Silicon devices. Since we need simple initial conditions for the Monte Carlo method, so we have used the simplified version. The MEP model is based on the closure of the semi classical Boltzmann equation by means of the maximum entropy principle [9] [14].

Using the relaxation time approximation (for only the moments and not for the energy moment) and using the so-called Liotta-Mascali distribution function which has the following form

$$f = (\frac{3\hbar^2}{4\pi m^* W})(\frac{3}{2})ne^{-\frac{3\epsilon}{2W}} + \frac{27\hbar^3}{32\pi\sqrt{2m^*}}\frac{n}{W}\epsilon^{-\frac{1}{2}}e^{-frac3\epsilon 2W}u^i v_i \tag{2.27}$$

we get the following hydro dynamical model for electrons, which we will call the Simplified MEP model.

$$\frac{\partial n}{\partial t} + \frac{\partial n u^i}{\partial x^i} = C_n \tag{2.28}$$

$$\frac{nu^j}{\partial t} + \frac{nK_B T^{ij}}{\partial x^i} + qE^j\frac{n}{m^*} = -\frac{nu^i}{\tau_p} \tag{2.29}$$

$$\frac{\partial nW}{\partial t} + \frac{\partial nS^i}{\partial x^i} + qE_i n u^i = -n\frac{W - W_0}{\tau_W} \tag{2.30}$$

Where τ_W is a function of the electrons energy. This function is computed numerically and reads:

$$\tau_W(W) = -\frac{W - \frac{3}{2}K_B T_L}{-7.24 \times 10^{10} W^5 + 5.13 \times 10^{11} W^4 - 1.36 \times 10^{12} W^3 + 3.69 \times 10^{11} W^2 - 3.07 \times 10^{12} W} \tag{2.31}$$

Furthermore, we have the following relations:

$$K_B T^{ij} = \frac{2}{3}\frac{W}{m^*}\delta^{ij} \tag{2.32}$$

$$S^i = \frac{4}{3} W u^i \tag{2.33}$$

For the moment relaxation time we have the following relations which are taken from the Baccarani model:

$$\tau_p = \frac{k_{\tau_p}}{T} \tag{2.34}$$

Where

$$k_{\tau_p} = \frac{m^* \mu_0 T_L}{q} \tag{2.35}$$

With μ_0 the low field mobility and T_L the lattice temperature. It is very easy to see how to adapt everything to Silicon heavy holes, so we do not report the Simplified MEP model for them.

2.7.1 Coupling Simplified MEP model and Monte Carlo method

In this section we show, in a cursory fashion, how it is possible to use the MEP simulation results to obtain faster Monte Carlo simulation.

The method is surprisingly simple and gives very fast and accurate results. First of all, we simulate a device by means of the simplified MEP model. When the simulation reaches the stationary solution, we save it and use it as a starting point for the Monte Carlo simulation. It uses the electron density, the potential, and what is the most important, the electron energy as a starting point. Concerning the energy as a starting point, it is very easy but, surprisingly, works very well. When we start the Monte Carlo simulation, we usually assign an electron energy which is proportional to $K_B T_L$, i.e. related to the lattice energy. Now, the only thing to do is to assign to the electrons the energy present in the cell *i,j* which has been computed by means of simplified MEP model. Then we assign the same potential and the same density computed previously by MEP [16].

CHAPTER 3

LINKING BETWEEN MONTE CARLO AND POISSON

3.1 INTRODUCTION

The most correct and predictive tool for the simulation of an electron gas in solid state matter coupled to its electrostatic potential, should be the Schrodinger-Poisson system. Even if some works have been done on this topic, it remains a very difficult.

Important problems still remains in the application of the boundary conditions for the Schrodinger equation, and it is very difficult, and for some process still impossible, to take into account all the relevant scattering events. Furthermore, solving Schrodinger-Poisson system is a very difficult task also from the numerical point of view, since the Schrodinger wave-function to be solved is a function in a $3N_e$ space, where N_e is the number of electrons simulated in the device. A solution like this, in realistic devices, is certainly a daunting task from the point of view of computer memory. This is why we have used the Monte Carlo method for the simulation.

In Monte Carlo electron gas simulations, it is very necessary to solve correctly both the dynamics of the particles and the computation of the electric field raising from the electron-hole distribution and, eventually, from applied potentials. This is because the charge transport in semiconductor devices is strongly dependent on the electric field, so, if the mentioned electric field is not correctly coupled to the charge dynamics, and correctly computed, all the simulation will be of no utility. How the Monte Carlo simulations and the Poisson equation are coupled are explained in this chapter.

3.2 THE CLOUD-IN-A-CELL ALGORITHM

Since the number of particles in a simulation is quite limited, if compared to the number of particles in a real semiconductor device, noise will always be present in the solutions [22] [23].

That is the reason to use an advanced algorithm in order to avoid, as the best as possible, this noise, instead of simply counting the number of particles in the cells of the simulation.

The brief description of the cloud-in-cell method is given below:

Let us consider a finite difference mesh with the nodes located at (x_i,y_j). Let us denote by Δx and Δy the constant spatial step in the x-direction and y-direction. Then, if we denote by (x,y) the point coordinates in which one wants to compute the density charge, with $x_i < x < x_{i+1}$ and $y_i < y < y_{i+1}$, we compute the density in the following way

$$n_{i,j} = \frac{S_{i,j}}{A_{i,j}^2}(x_{i+1} - x)(y_{j+1} - y) \quad (3.1)$$

$$n_{i+1,j} = \frac{S_{i+1,j}}{A_{i+1,j}^2}(x - x_i)(y_{j+1} - y) \quad (3.2)$$

$$n_{i,j+1} = \frac{S_{i,j+1}}{A_{i,j+1}^2}(x_{i+1} - x)(y_{j+1} - y) \quad (3.3)$$

$$n_{i+1,j+1} = \frac{S_{i+1,j+1}}{A_{i+1,j+1}^2}(x - x_i)(y - y_j) \quad (3.4)$$

Where $n_{i,j}$ is the density located at (x_i,y_j), $S_{i,j}$ the statistical weight of the particles located at (x_i,y_j) and $A_{i,j} = \Delta x_i \Delta y_j$. Methods do exist that avoid the problems of self-forces but they are necessary only when the grid is not regular and when we deal with hetero structures.

3.3 THE STATIONARY POISSON EQUATION

In semiconductor devices, the potential retardation effects are completely negligible so

1. We have neglected the computation of the magnetic field.

2. We can adopt the stationary description of the electric potential, i.e. we select and calculate only the Poisson equation among the set of Maxwell's equations.

The Poisson equation is given below:

$$\nabla \cdot [\epsilon(\mathbf{x})\nabla\phi(\mathbf{x},t)] = -q[N_D(\mathbf{x}) - N_A(\mathbf{x}) - n(\mathbf{x},t) + p(\mathbf{x},t)] \tag{3.5}$$

Actually, if we have a two-dimensional regular finite-difference grid, the discretization of the Poisson will give an algebraic system to solve, which is quite complicated to solve [22] [23].

3.4 THE NON-STATIONARY POISSON EQUATION

This equation is very easy to implement and solve with the numerical schemes. The NSP equation is given below:

$$\frac{1}{k_S}\frac{\partial\phi}{\partial t} + \nabla \cdot [\epsilon(\mathbf{x})\nabla\phi(\mathbf{x},t)] = -q[N_D(\mathbf{x}) - N_A(\mathbf{x}) - n(\mathbf{x},t) + p(\mathbf{x},t)] \tag{3.6}$$

Where k_S is a constant for giving the right dimensions of the term $\frac{d\emptyset}{dt}$ and the other variables have the usual meaning. The solution of this equation will converge in time, to the solution of the classical Poisson equation described in the precedent paragraph, whatever are the initial potential conditions and with the same boundary conditions. So, if we have a numerical solver for this equation, it would be very easy to get the solution of the classical Poisson equation, simply getting the solution of the NSP equation for very big final time.

Actually it is trivial to develop and implement a numerical solver for NSP. In fact, in the context of finite difference, such a numerical scheme can be obtained applying finite-difference approximations of derivatives to the NSP equation. This is what we have discussed in the following paragraph [22] [23].

3.4.1 Numerical Resolution of the NSP Equation

In the context of finite-difference approximations, we can trivially write

$$\frac{\partial\phi}{\partial t}(x_i, y_j) = \frac{\phi_{i,j}^{n+1} - \phi_{i,j}^{n}}{\Delta t} \tag{3.7}$$

$$\nabla^2\phi(x_i, y_j, t^n) = \frac{\phi^n_{i+1,j} - 2\phi^n_{i,j} + \phi^n_{i-1,j}}{\Delta x^2} + \frac{\phi^n_{i,j+1} - 2\phi^n_{i,j} + \phi^n_{i,j-1}}{\Delta y^2} \tag{3.8}$$

Where $\varphi^n_{i,j}$ is the potential computed at time $t^n = t_i + n\Delta t$, in the point (x_i,y_j).Applying these approximations to the NSP equation, we have got the following numerical scheme

$$\phi^{n+1}_{i,j} = \phi^n_{i,j} + \Delta t(-\epsilon_{i,j}(\frac{\phi^n_{i+1,j} - 2\phi^n_{i,j} + \phi^n_{i-1,j}}{\Delta x^2} + \frac{\phi^n_{i,j+1} - 2\phi^n_{i,j} + \phi^n_{i,j-1}}{\Delta y^2}) - q[N_{Di,j} - N_{Ai,j} - n^n_{i,j} + p^n_{i,j}]) \tag{3.9}$$

Note that the above scheme is valid only for the homogeneous case.

As we can see, once we have the initial conditions and the boundary conditions, it is very easy to implement this equation in the simulator.

3.5 ELECTRIC FIELD CALCULATION

The electric field is easily computed once we have the solution of the static Poisson equation or the NSP equation. The definition of the electric field is as follows

$$\mathbf{E}(x, y) = -\nabla\phi(x, y) \tag{3.10}$$

So, in the context of finite-difference approximations, the electric field is computed in the various cells of the grid as follows

$$E_{xi,j} = -\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x} \tag{3.11}$$

$$E_{yi,j} = -\frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y} \tag{3.12}$$

These simple expressions are accurate and robust.

CHAPTER 4

ARCHIMEDES INSTRUCTIONS SET

In order to make a simulation of a new general device, in **Archimedes**, we needs to describe this device by means of ASCII scripts using keywords belonging to Archimedes meta-language. This meta-language is very generic, so it gives the possibility of defining semiconductor devices of quite general structures. Furthermore, since the keywords are very simple to understand (and remember), it is possible to define devices in short amounts of time and change them with small modifications in the input ASCII file, in case of, for example, optimization process or similar[24].

Thus, in this chapter, we describe the syntax of all the commands actually implemented in **Archimedes**. The definition of a new device is done by means of a user defined ASCII input file, which is processed by **Archimedes**. This is done typing the following command line in a shell

```
# archimedes filename.extension
```

Where"filename.extension" is the name of the ASCII file in which the user have defined the device to be simulated. Some extra options are possible with Archimedes. For this type in a shell window

```
# archimedes --help
```

VERY IMPORTANT REMARK: Archimedes is case sensitive so every command have to be written in capitals, otherwise it will not understand the command. Furthermore, all the unity of measure are taken from the international M.K.S.C. system.

4.1 ACCEPTOR DENSITY

In the definition of a new device, even if the holes are considered as fixed in **Archimedes**, we have to specify the acceptor density, i.e. the spatial distribution of the acceptors in the device. This is done in order to solve correctly the Poisson equation. In

fact, this last equation needs both the donor and acceptor distribution, so it is necessary to specify them. If we do not specify any constant value or distribution of the acceptors, **Archimedes** will consider that the acceptor distribution is constant on all the device and it is equal to the intrinsic density as default. Furthermore, if we specify the value of the acceptor distribution only on a part of the device, the resistant part will be considered equal to the intrinsic density.

Let us see, now, how to specify the acceptor distribution on a device. In **Archimedes** a sub-domain (but also the entire device) on which we want to specify an acceptor value for the spatial distribution is just a simple rectangle. This means that we can specify the value of the acceptor density on a rectangle (the entire device or a part of it), specifying only five numbers i.e.

1. The x-coordinate value of the left-bottom vertex of the rectangle. Let us denote it by x_{min}
2. The y-coordinate value of the left-bottom vertex of the rectangle. Let us denote it by y_{min}
3. The x-coordinate value of the right-upper vertex of the rectangle. Let us denote it by x_{max}
4. The y-coordinate value of the right-upper vertex of the rectangle. Let us denote it by y_{max}
5. The value of the acceptor density on the rectangular sub-domain. Let us denote it by N_A

Then we will have the acceptor density N_A on the rectangle $[x_{min},x_{max}] \times [y_{min},y_{max}]$.

An example will clarify everything

```
# acceptor spatial distribution on the rectangle [0.0,1.0e-6]x[0.0,0.1e-6]
# acceptor density on this rectangle is equal to 1.e20

ACCEPTORDENSITY 0.       0.    1.0e-6    0.1e-6    1.e20
```

4.2 CIMP

Since the impurity scattering can be very relevant in the GaAs material, it is implemented in **Archimedes**. In order to specify the density of impurity, we type

```
CIMP 1.e23
```

4.3 COMMENTS

Like in every computer language, comments are very important for the clarity of a code. We can make our own comments by simply preceding them by the # symbol. So, for example, the following rows of an ASCII file processed by **Archimedes** will be interpreted as comments.

```
# this is comment
# MATERIAL X 0.0 1.0e-6   Y 0.0 0.1e-6   SILICON
# even if the precedent row contains a command
# this will never processed and it will be consider
# simply a comment
```

Pay attention to the fact that everything after the # symbol is a comment even if this is a command usually recognized by Archimedes.

4.4 CONTACT

When a new device is defined, we needs to specify where the edges of insulator, where there are Ohmic contacts and where the Schottky contacts are positioned. Even, we can have the need of applying a potential on an insulator edge (for simulation purposes). All this definitions are possible by the use of only one command, i.e. **CONTACT**. Let us describe the syntax of this command.

This command can be described as follows

```
CONTACT  place  init_pos  fin_pos  kind  pote  dens
```

Where place can be one of the following choice

1. **UP**. This in invoked when the contact have to be placed on the upper edge of the device.
2. **DOWN**. This in invoked when the contact have to be placed on the bottom edge of the device.
3. **RIGHT**. This in invoked when the contact have to be placed on the right edge of the device.
4. **LEFT**. This in invoked when the contact have to be placed on the left edge of the device.

Furthermore, init_pos is the initial position of the contact, fin_pos is the final position of the same contact. The choice kind can be one of the following

1. **INSULATOR**. This is invoked in the case the contact is of insulator type. In this case, the contact will be “reflective mirror” for the particles, i.e. the particles cannot go out or inside the device through that contact.

2. **OHMIC**. This is invoked in the case the contact is of Ohmic type. This kind of contact can be considered as a gate through which the particles can go out. Furthermore, it can be considered as a particle reservoir from which particles can go into the device.

3. **SCHOTTKY**. This is invoked in the case the contact is of Schottky type. This kind of contact is the same as the Ohmic one with the exception that this is not a particle reservoir, so this contact is only an absorbing one.
The choice pot is the potential which is applied to this contact. In the case the edge, or a part of it, is of insulator type and there is no potential applied there, then it have to be put to 0.

The choice **dens** is the density of the particle reservoir, so it has to be specified only in the Ohmic contact case.

When an edge, or part of it, will not be specified by us, it will be considered as of insulator type with zero potential applied, as default [24].

If we want to specify that the upper edge is of insulator type, with no applied potential, in a diode with x-direction length 1.0 micron, than we have to write

```
CONTACT    0.0    1.0e-6    INSULATOR    0.0
```

If we want to specify that the left edge is of Ohmic type, with zero applied potential and $\frac{1.e23}{m^3}$, in a diode with y-direction length 0.1 micron, than we have to write

```
CONTACT    0.0    0.1e-6    OHMIC    0.0    1.e23
```

Finally, if we want to specify that a part of the upper edge is of Schottky type, with −0.8 Volts applied potential starting from 0.2 × 10−6 to 0.4 × 10−6, in a MESFET, than the we have to write

```
CONTACT    0.2e-6    0.4e-6    SCHOTTKY    -0.8
```

4.5 DONORDENSITY

In the definition of a new device, we have to specify the donor density, i.e. the spatial distribution of the donors in the device. This is done in order to solve correctly the Poisson equation. In fact, this last equation needs both the donor and acceptor distribution, so it is necessary to specify them. If we do not specify any constant value or distribution of the donors, **Archimedes** will consider that the donor distribution is constant on all the device and it is equal to the intrinsic density as default.

Furthermore, if we specify the value of the donor distribution only on a part of the device, the restart part will be considered equal to the intrinsic density.

Let us see, now, how to specify the donor distribution on a device. In **Archimedes** a sub-domain (but also the entire device) on which we want to specify a donor value for the spatial distribution is just a simple rectangle. This means that we can specify the value of the donor density on a rectangle (the entire device or a part of it), specifying only five numbers i.e.

1. The x-coordinate value of the left-bottom vertex of the rectangle. Let us denote it by xmin
2. The y-coordinate value of the left-bottom vertex of the rectangle. Let us denote it by ymin
3. The x-coordinate value of the right-upper vertex of the rectangle. Let us denote it by xmax
4. The y-coordinate value of the right-upper vertex of the rectangle. Let us denote it by ymax
5. The value of the donor density on the rectangular sub-domain. Let us denote it by NA

then we will have the donor density NA on the rectangle [xmin, xmax] × [ymin, ymax] [24].

An example will clarify everything.

```
# donor spatial distribution on the rectangle [0.0,1.0e-6]x[0.0,0.1e-6]
# donor density on this rectangle is equal to 1.e20

DONORDENSITY 0.       0.    1.0e-6    0.1e-6    1.e20
```

4.6 LEID

This is a very powerful command. This command is invoked when we want to obtain Monte Carlo simulations in a very fast way. Let us suppose that we have simulated a device by means of the simplified MEP model. Then we can use the results obtained by this model as a starting point for the Monte Carlo simulation. All it has to be done is to save the electron density, the electron energy and the potential in files named respectively

```
density_start.xyz
energy_start.xyz
potential_start.xyz
```

Then we have to invoke the command **LEID** (LEID stands for Load Electrons Initial Data). Pay attention to the fact that the input files have to be of the same dimension of the grid for the Monte Carlo simulation [24].

4.7 MATERIAL

When we simulate a new device, we have the freedom to choose to simulate a heterostructure. we should specify which zone is made of a certain semiconductor material (like Silicon, Gallium Arsenide, Germanium, InSb, AlSb, AlAs and so on..)

This is done by using the command **MATERIAL**. The syntax of this command is as follows

```
MATERIAL X xi xf   Y yi yf semmat
```

Where (*xi,yi*) and (*xf,yf*) are the initial and final corners of a rectangle made of semmat. semmat can be one of the following choice [24]:

```
SILICON
GAAS
GERMANIUM
INSB
ALSB
ALAS

ALxINxSB
ALxIN1-xSB
```

4.8 FARADAY

It is possible to simulate the self-consistent magnetic field produced by the charged particles in the device. This is obtained by simulating the well-known Faraday equation which deals with dynamic magnetic fields due to the moving charged particles.

By default, the simulation of the self-consistent magnetic field is NOT taken into account (since it is usually negligible in the majority of simulated devices). If we want to simulate it, he has to specify it by means of the command FARADAY.

The syntax of this command is straight simple, being

```
FARADAY ON/OFF
```

In other words, the user decides, by means of this command, to switch on or off the self-consistent magnetic field [24].

4.9 BCONSTANT

It is possible to simulate the presence of an externally applied magnetic field. This is done by means of the command **CONSTANTMAGNETICFIELD** (in this case, it is strongly suggested to switch off the **FARADAY** command).

The syntax of this command is straight forward:

```
CONSTANTMAGNETICFIELD xi yi xf yf B
```

Where (*xi,yi*) and (*xf,yf*) are the corners of the rectangle where the magnetic field *B* is applied (*B* is in *Weber/m*^{2} units, i.e. a Newton / (Ampere * meter) [24]

4.10 TRANSPORT

When the device is defined, the user has to choose what kind of transport **Archimedes** have to simulate, i.e. if the transport is unipolar, bipolar and what kind of particles have to be simulated. This is done by the command **TRANSPORT**. The following list shows the choice we can make. The only choices which is still not implemented is the Monte Carlo simulation of heavy holes. They are simulated by means of a simplified MEP model since they can be considered as almost fixed and do not contribute to the total device current. First of all, after typing the command **TRANSPORT** we have to specify the model (i.e. Monte Carlo or MEP). This is done typing one of the following two choices.

1. **MC**. This is invoked when we want to simulate a device by means of Monte Carlo method.
2. **MEP**. This is invoked when we want to simulate a device by means of the simplified MEP model.

Once the method has been specified, we have to choose what particles have to be simulated. This is done choosing between the followings:

1. **ELECTRONS**. This is invoked when the transport is unipolar and made of only electrons.
2. **HOLES**. This is invoked when the transport is unipolar and made of only holes.
3. **BIPOLAR**. This is invoked when the transport is bipolar and made of both electrons and holes.

So, some examples of this command, in the **Archimedes**, are [24]

```
# Monte Carlo simulation of only electrons
 TRANSPORT MC ELECTRONS
# Simplified MEP simulation of only electrons
 TRANSPORT MEP ELECTRONS
```

```
# Monte Carlo for electrons and MEP for heavy holes
 TRANSPORT MC BIPOLAR
# Simplified MEP model for both electrons and holes
 TRANSPORT MEP BIPOLAR
```

4.11 FINALTIME

It is important to choose the final time of a simulation. In fact, if we want to simulate the stationary solution of a semiconductor device, he have to choose an appropriate final time, in order to get the stationary state but without waiting for not-necessary long simulation run-time. At the moment, there is no algorithm which can predict the final time in order to get the stationary state of a device, so the freedom of choosing this final time is given to us. This is also useful in the case in which we want to study and simulate the transient behavior of a semiconductor device. So, for example, if we want to set the final time to 5 picoseconds, this will be done by the following line of the input file [24]

```
FINALTIME 5e-12
```

4.12 TAUW

This command is only needed if we are using the simplified MEP model for electrons. Indeed, in this case the relaxation time approximation for the electrons energy can be equal to zero (as we can see from the definition of the function τ_W) comporting the presence of NaN. This is avoided by specifying a value for tauw that will be used in the case it is equal to zero [24]. The command is invoked as it follows

```
TAUW value
```

Where value is the value specified. Usually a good value is 0.4 picoseconds. We report an example

```
TAUW 0.4e-12
```

As we can see from the example, the unit of measure is the second.

4.13 TIMESTEP

It is very important to choose correctly the time step of a simulation. This is a very important topic of a simulation. This have to be done in a very accurate manner, in order to avoid unphysical phenomena like strange oscillations in electric field or too much scattering effects and so on... Actually, algorithm based on the plasma oscillations of a gas of charged particles exists which are useful in the choice of a correct time step [24]. This is not implemented, at the moment, so we have to specify it. This is done in the following fashion. For example, if we want to set the time step to 0.01 picoseconds, the following command line in the input file is needed

```
TIMESTEP 0.01e-12
```

4.14 XLENGTH

In **Archimedes**, the device is defined as a rectangular domain. So we have to specify the x-direction and the y-direction length of this rectangular domain. Concerning the x-direction length, this is settled by the following command line [24]. For example, if the x-length is 5 micron, we have to write

```
XLENGTH 5.e-6
```

4.15 YLENGTH

In **Archimedes**, the device is defined as a rectangular domain. So we have to specify the x-direction and the y-direction length of this rectangular domain. Concerning the y-direction length, this is settled by the following command line [24]. For example, if the y-length is 1 micron, we have to write

```
YLENGTH 1.e-6
```

4.16 XSPATIALSTEP

Once the x-direction length of the new device is defined, we have, obviously, to define the number of cells in the x-direction. This is done in order to solve every equations of the simulation in the finite difference approximation context. The bigger is the number of cells in the x-direction the best will be the accuracy in that direction, but we will pay the better accuracy in a more long

run-time. So, pay attention in the choice of this number of cells. Usually, a grid of 100 × 50 is enough for the majority of devices, but it strongly depends on the device structure and the requirements of the user [24]. To specify, for example, 100 cells in the x-direction, the following line have to be typed in the input

```
XSPATIALSTEP 100
```

4.17 YSPATIALSTEP

Once the y-direction length of the new device is defined, we have, obviously, to define the number of cells in the y-direction. This is done in order to solve every equations of the simulation in the finite difference approximation context. The bigger is the number of cells in the y-direction the best will be the accuracy in that direction, but we will pay the better accuracy in a more long run-time. So, pay attention in the choice of this number of cells. Usually, a grid of 100 × 50 is enough for the majority of devices, but it strongly depends on the device structure and the requirements of the user [24]. To specify, for example, 50 cells in the x-direction, the following line have to be typed in the input

```
YSPATIALSTEP 50
```

4.18 QUANTUMEFFECTS

As we have said in a precedent chapter, Archimedes is able to simulate, at least at first order, the quantum effects present in a semiconductor device, by means of the recently introduced effective potential method. So, if we want to take into account the quantum effects in the simulation, we have to tell to **Archimedes** in the input file. This is done in the following fashion

```
QUANTUMEFFECTS
```

Pay attention to the fact that taking into account the quantum effects can be numerically heavy, so we will need to wait for more long run-time to get the solution. So attention have to be putted in this choice. If we know appropriately, that the quantum effects are not relevant in that type of device (because, for example, the characteristic length of the device is bigger than 1 micron), it is probably a good choice to avoid these extra calculations [24].

4.19 NOQUANTUMEFFECTS

In the case we want to avoid the calculations of the quantum effects, it is necessary to set it into the input file processed by **Archimedes**. This is done in the following way

```
NOQUANTUMEFFECTS
```

4.20 MAXIMINI

During the simulation computations, it is, sometimes, important to see in a very rapid way, how the macroscopic variables evolves. This can be very useful in the debugging process of a new defined (and not yet well-defined) semiconductor device. **Archimedes** gives a simple way of doing it. When we inserts the following line in an input file, it will get some interesting information about the macroscopic evolution of the devices.

```
MAXIMINI
```

Pay attention to the fact that the calculation of this information can be, in some cases (for example, for highly refined grid), computationally heavy. So use it only in the case it is necessary or not heavy for the simulation, in order to avoid too big run-time [24].

4.21 NOMAXIMINI

In the case we want to avoid to have extra information about the macroscopic evolution of some variables, this have to be specified in the input file processed by **Archimedes**. This is done by the following row

```
NOMAXIMINI
```

This will be useful in the case we already know the transient macroscopic behavior of the device and does not need such information, in order to not tax the computer run-time [24].

4.22 SAVEEACHSTEP

When it is required to simulate the transient behavior of a semiconductor device, it is very useful to save all the solutions at each time step of the simulation. In this way, it is possible to create

movies which show the transient behavior of the density, or of the electrostatic potential, for examples. This movies are very good in the comprehension of the transient states of a new semiconductor device. The only thing to do is to put the following line in the input file processed by **Archimedes**

```
SAVEEACHSTEP
```

This will save all the solution, in the chosen format, with the following convention: the file are named in increasing order, i.e., for density, density001.xyz, density002.xyz, density003.xyz, ..., density010.xyz and so on. Instead, the last final step will be saved with the suffix '000'. Pay attention to the fact that the savings of all this information can be, in some cases (for example, for highly refined grid), computationally heavy. So use it only in the case it is necessary or not heavy for the simulation, in order to avoid too big run-time [24].

4.23 LATTICETEMPERATURE

Archimedes is a semiconductor device simulator in a quite general context. So when we simulate a new device, we have to specify the lattice temperature. If we do not specify this value, the room temperature will be taken as default (i.e. 300 Kelvin degrees). All the temperatures are given in Kelvin [24]. So if we deal with a cryogenic device, i.e. working at 77 Kelvin degrees, we have to specify the following row in the input file

```
LATTICETEMPERATURE 77
```

4.24 STATISTICALWEIGHT

The method implemented in **Archimedes** is the Monte Carlo one. So every particle carries a statistical weight which is a piecewise-function of the position. The greater is the statistical weight, the greater is the number of super-particles in a cell. In this **Archimedes**, the statistical weight we can specify is that of the cell in which the density is at maximum. In the other cells, the statistical weight is opportunely calculated [24]. If we, for example, want to set the statistical weight equal to 1500, we have to write in the input file.

```
STATISTICALWEIGHT 1500
```

Pay attention to the fact that the bigger is the statistical weight the longer will be simulation run-time.

4.25 MEDIA

Monte Carlo method is a statistical one. So, in order to get the macroscopic variables at a certain time, we need to compute the average mean value of this variable on an enough long period of time. This is done, in **Archimedes**, by specifying on how many final time step the mean average value have to be computed [24]. So, for example, if we want to compute the average mean value of the macroscopic variables on the last 500 time steps of the simulation, we have to type in the input file processed by Archimedes

```
MEDIA 500
```

4.26 OUTPUTFORMAT

When **Archimedes** saves the various solution outputs, it has to know in which format to do it. This is specified in the following way, in the input file

```
OUTPUTFORMAT formattype
```

Where format type can be one of the following two choices

1. **GNUPLOT**. This sets that the output files will finish by the extension xyz which means that the file will be in the following format

$$
\begin{array}{lll}
x_i & y_j & v(x_i, y_j) \\
x_{i+1} & y_j & v(x_{i+1}, y_j) \\
x_{i+2} & y_j & v(x_{i+2}, y_j) \\
x_{i+3} & y_j & v(x_{i+3}, y_j) \\
 & \ldots & \\
x_{N_x} & y_j & v(x_{N_x}, y_j) \\
 & \ldots &
\end{array}
$$

$$
\begin{array}{lll}
x_i & y_{j+1} & v(x_i, y_{j+1}) \\
x_{i+1} & y_{j+1} & v(x_{i+1}, y_{j+1}) \\
x_{i+2} & y_{j+1} & v(x_{i+2}, y_{j+1}) \\
x_{i+3} & y_{j+1} & v(x_{i+3}, y_{j+1}) \\
 & \ldots & \\
x_{N_x} & y_{N_y} & v(x_{N_x}, y_{N_y})
\end{array}
$$

2. **MESH**. The MESH format is a little bit more complex than the GNUPLOT one. In this case, we have a file which describes the mesh and another which describes the solution on that mesh. Concerning the mesh file, it have a file structure like the following

```
MeshVersionFormatted 1
Dimension 2
Vertices
number of vertices

x1 y1 0
x2 y1 0
x3 y1 0
...
xn y1 0
...
xn yn 0
Quadrilaterals
number of quadrilaterals
v1_1 v2_1 v3_1 v4_1 0
v1_2 v2_2 v3_2 v4_2 0
...
v1_N v2_N v3_N v4_N 0
```

Concerning the file for the solution on the mesh, the file have the following structure

```
2 1 number of vertices 2
solution on the 1-st vertex
solution on the 2-nd vertex
solution on the 3-th vertex
solution on the 4-th vertex
...
solution on the last vertex
```

This kind of format is becoming very popular in the scientific/numerical community.

CHAPTER 5

SIMULATION RESULTS AND DISCUSSION

Simulation of electronic transport in GaAs based MESFET is done by ensemble Monte Carlo method using Archimedes software package and GUI plot. The following device modelling is used for simulation.

5.1 DEVICE MODELLING

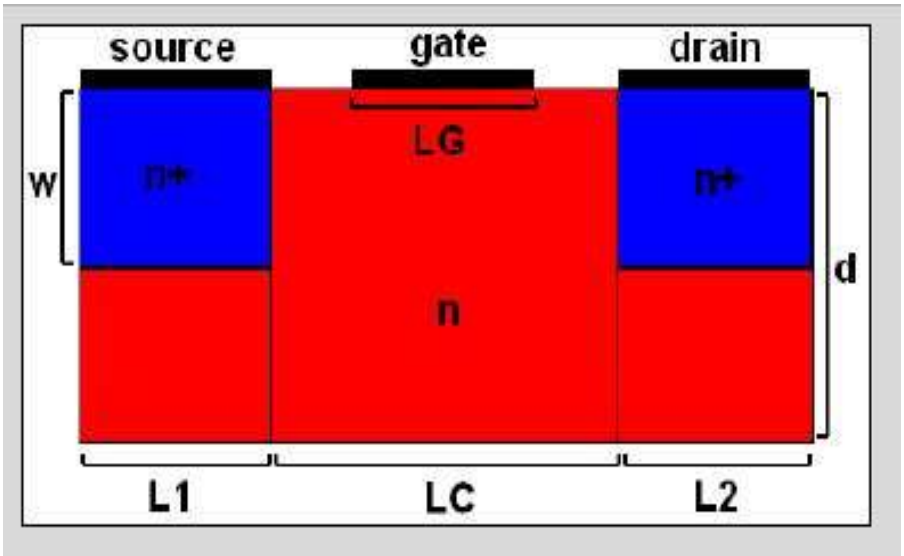

Fig 5.1: GaAs MESFET Device structure

Where

L1: Length of Source =250nm,

L2: Length of Drain =250nm,

LC: Length of Channel=250nm,

d: Width of MESFET =500nm,

n^+: Doping concentration in Source and Drain= $10^{18}/cm^3$

n=doping concentration of substrate = $10^{16}/cm^3$

5.2 OBJECTIVES

(i) Simulation of energy band structure in x-y plane.

(ii) Simulation of electron density in x-y plane.

(iii) Simulation of electron energy in x-y plane.

(iv) Simulation of electrostatic potential in device area.

(v) Simulation of E-field in device area.

(vi) Simulation of carrier velocity in x-y plane.

5.3 SIMULATION RESULTS

All the simulations are performed at the room temperature.

(i) Simulation of energy band structure in x-y plane

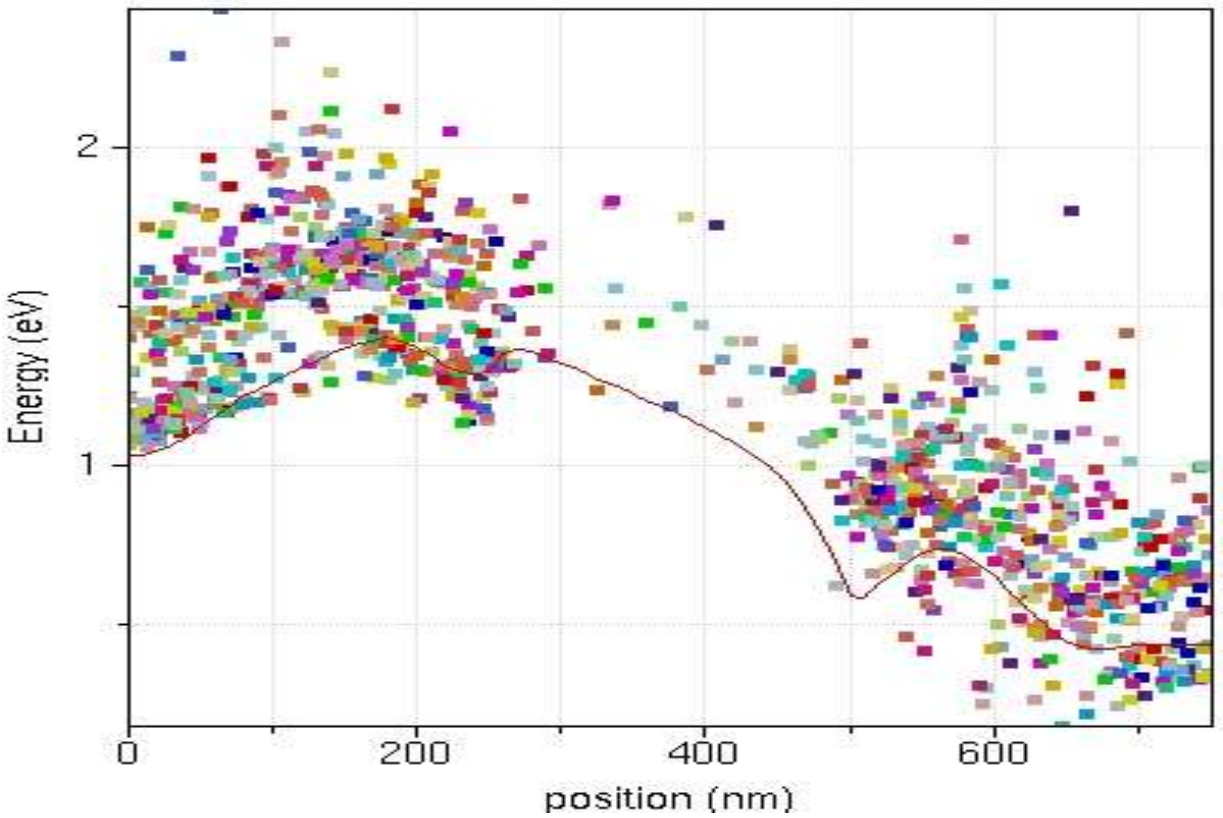

Fig 5.2: Conduction band in x-direction

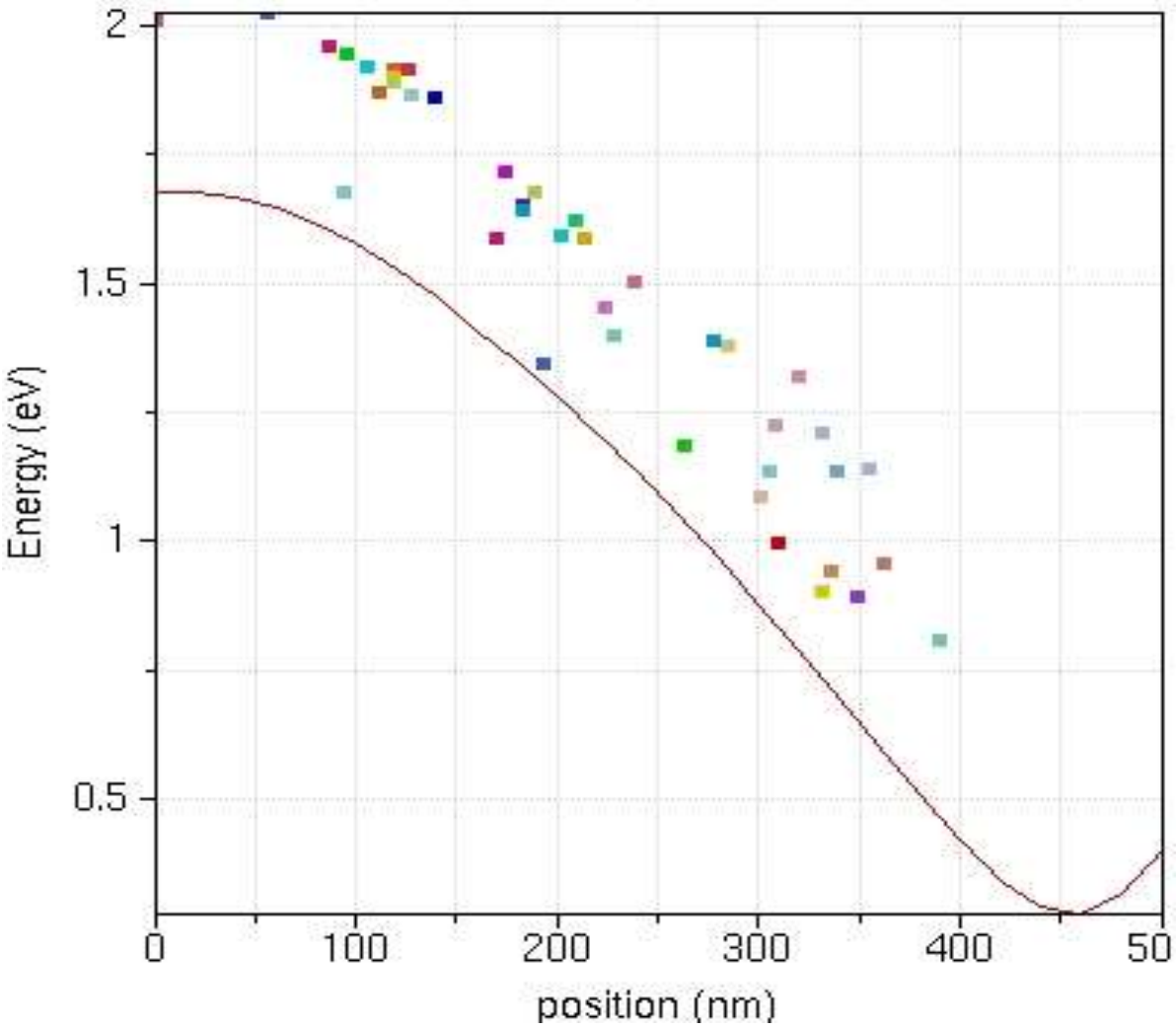

Fig 5.3: Conduction band in y-direction

From the figure it can be seen that electrons in the conduction band of source region has higher energy than the electrons in the conduction band of drain region. Since the biasing voltage in drain is larger than source region, the barrier potential for the electrons is lesser in drain region than source region causing electrons to move from source to drain and producing current I_{DS} in opposite direction.

(II) SIMULATION OF ELECTRON DENSITY IN X-Y PLANE

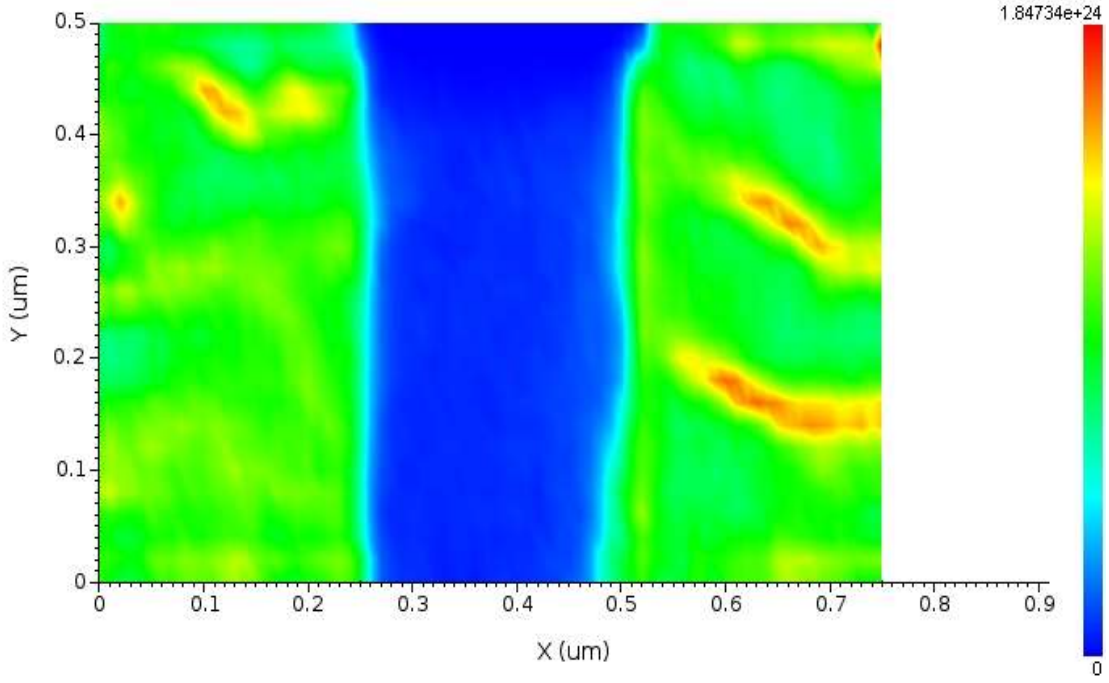

Fig 5.4: Electron density profile of GaAs MESFET

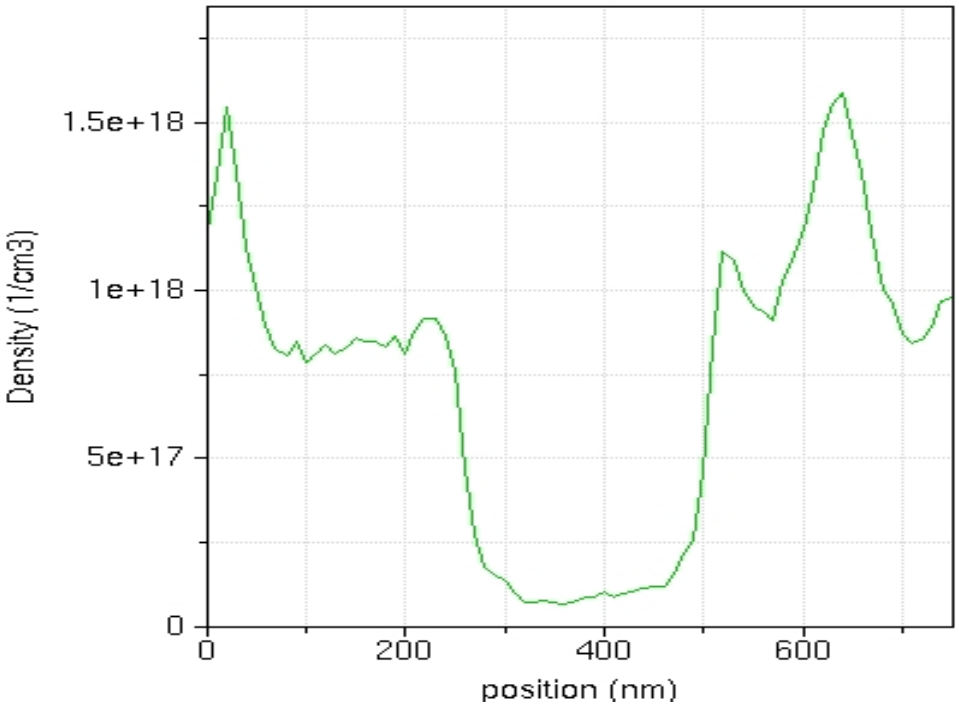

Fig 5.5: Electron density in x-direction

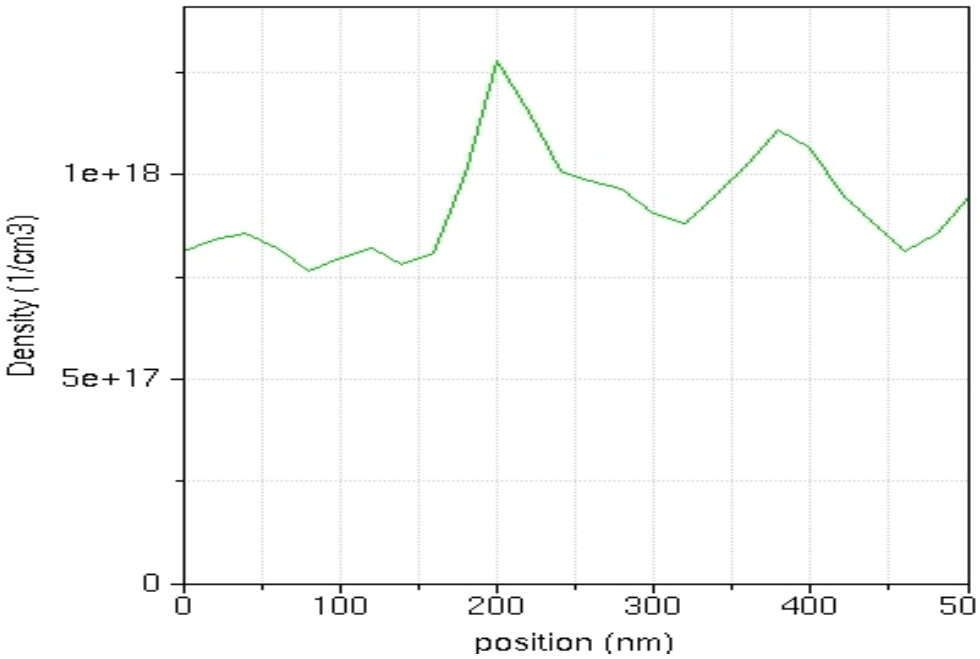

Fig 5.6: Electron density in y-direction

As Source and drain are doped with higher concentration than substrate we can see from the graph there are very less charge carriers exist in the channel region. Density fluctuation can be seen in source and drain region due to thermal vibration of carriers.

(III) SIMULATION OF ELECTRON ENERGY IN X-Y PLANE

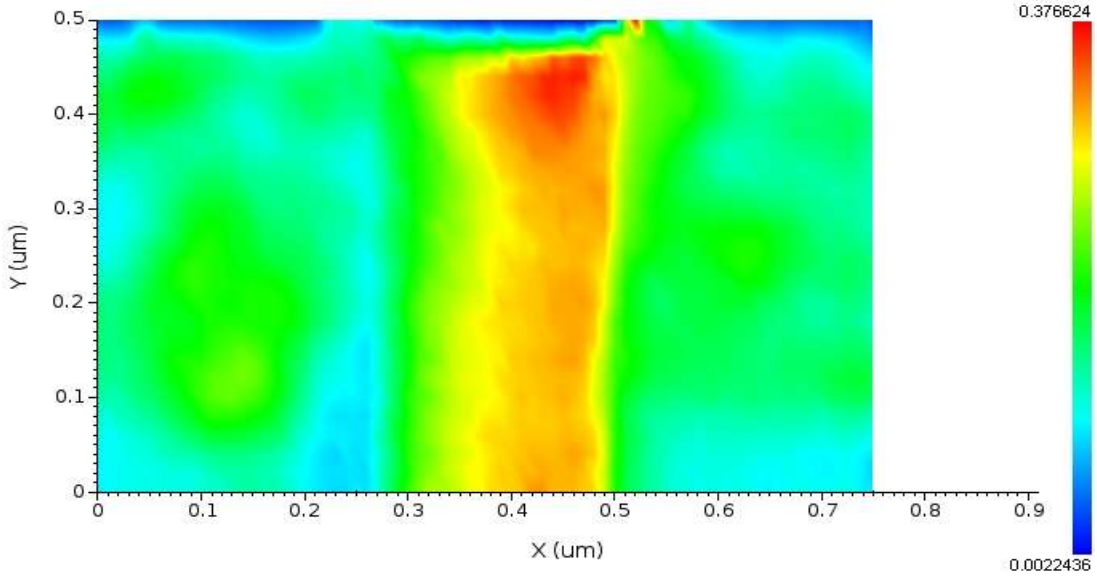

Fig 5.7: Electron energy profile in GaAs MESFET

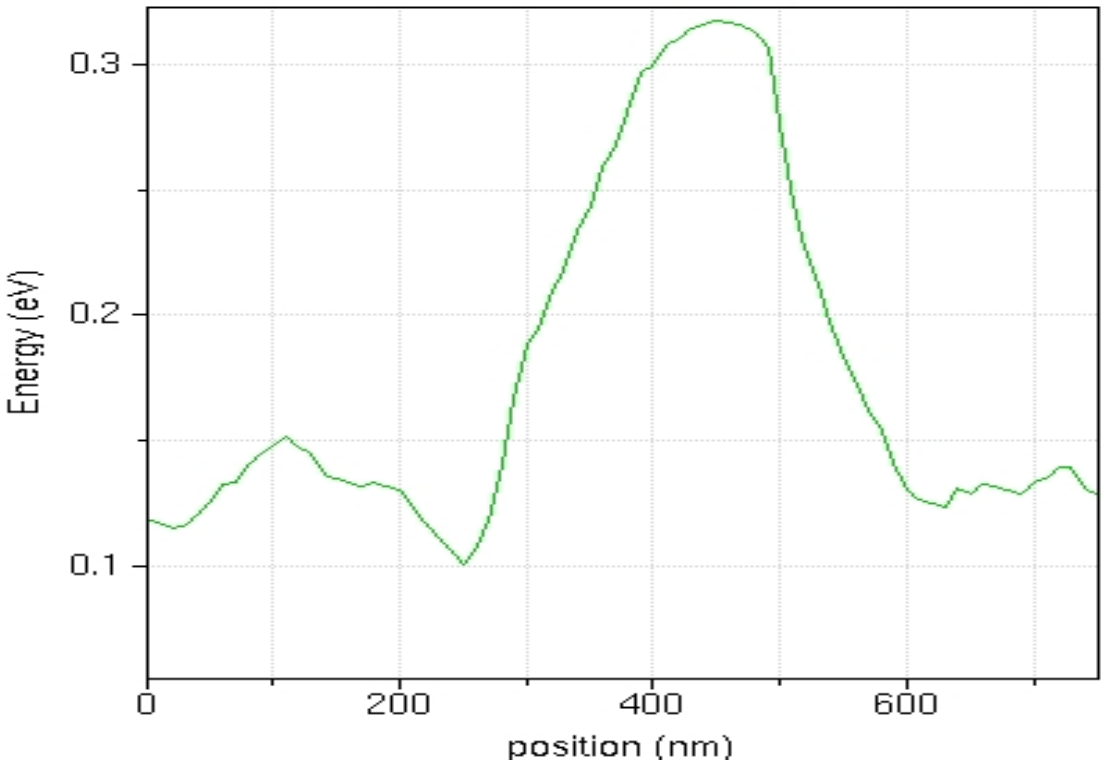

Fig 5.8: Electron energy in x-direction

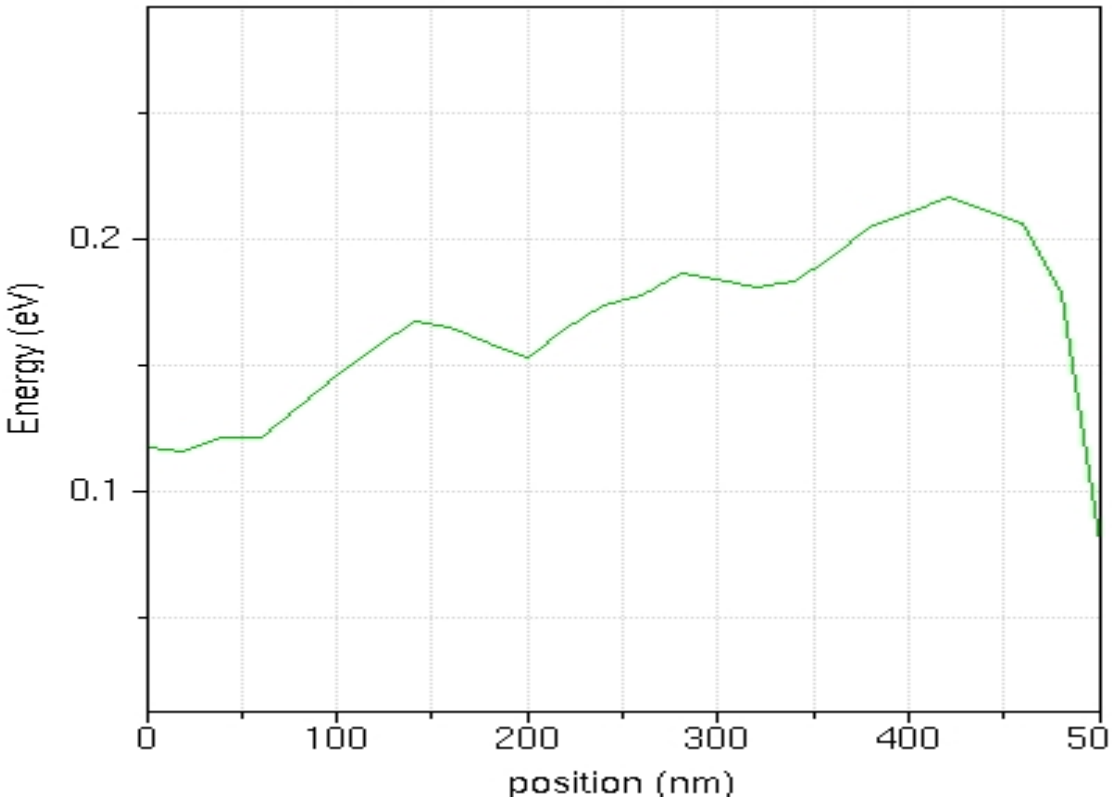

Fig 5.9: Electron energy in y-direction

As seen from the above figures maximum energy of electrons is found in the channel region. The reason for this is the channel has more free space than source and drain as it is less crowed where carriers can move freely and gain kinetic energy. The maximum energy difference between valleys of conduction band is found to be 0.2 eV.

(IV) SIMULATION OF ELECTROSTATIC POTENTIAL IN DEVICE AREA

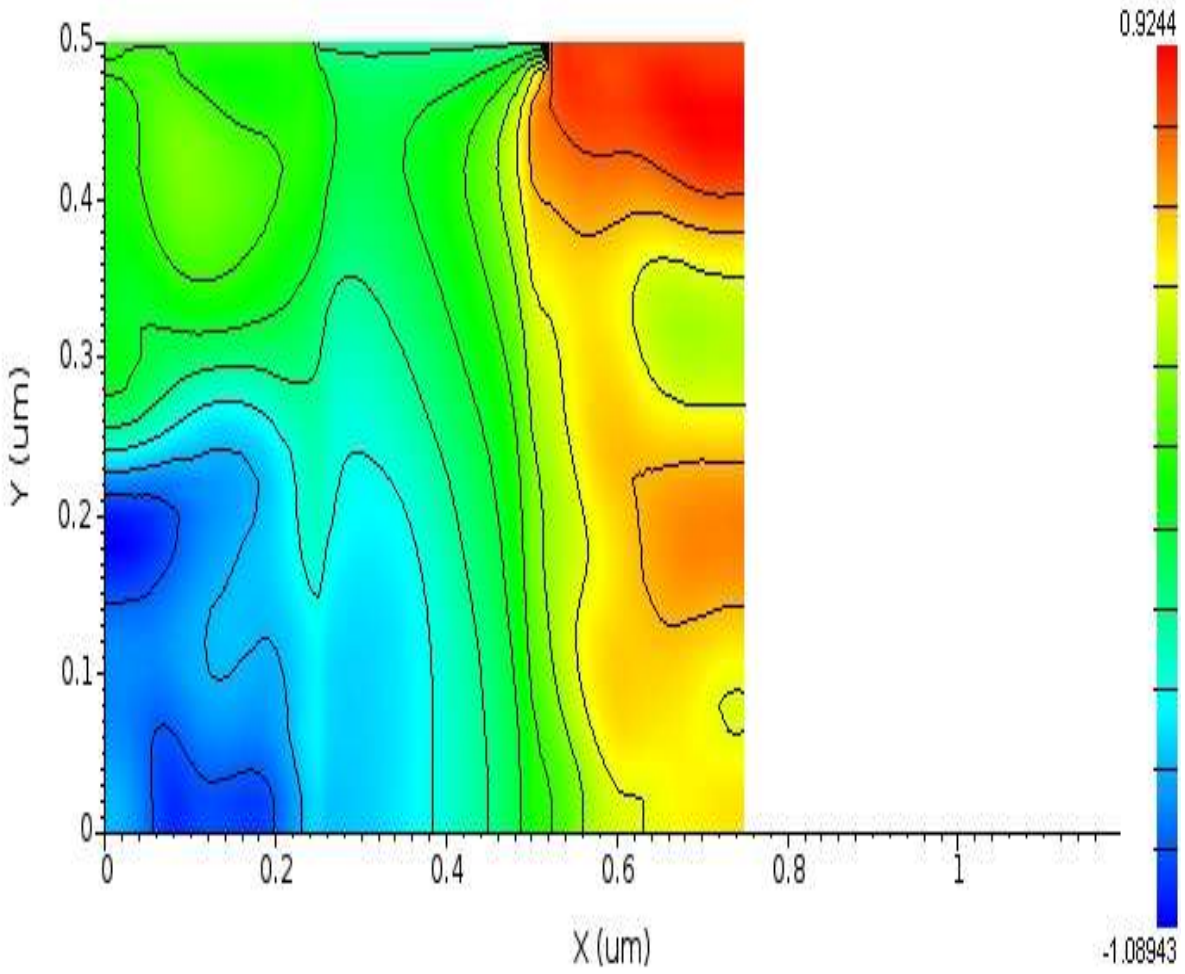

Fig 5.10: Electrostatic potential profile in GaAs MESFET

As seen from the above figure electrostatic potential decreases form drain to source region as drain is applied with more bias voltage than source.

(V) SIMULATION OF ELECTRIC FIELD IN DEVICE AREA

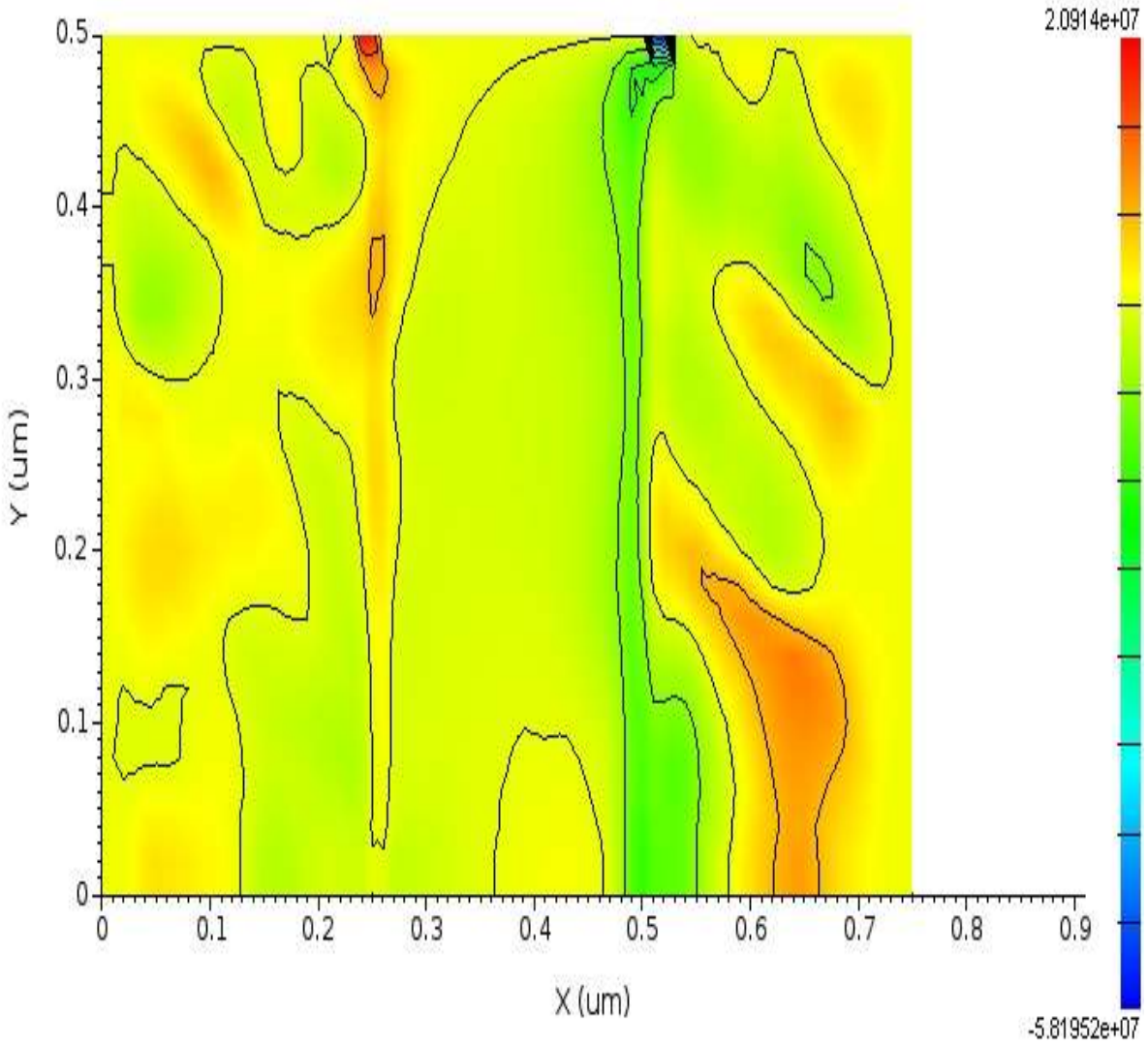

Fig 5.11: Electric field (V/m) profile in GaAs MESFET

As seen from above figure maximum electric field intensity is found in source and drain regions as carriers in the source and drain regions are at higher concentration, their effect along with biasing field is reason for high field intensity in source and drain regions. The negative electric field experienced by electrons is due to the inter valley transfer of electrons and maximum field is 5.81 x 10^7 V/m.

(VI) SIMULATION OF CARRIER VELOCITY IN X-Y PLANE

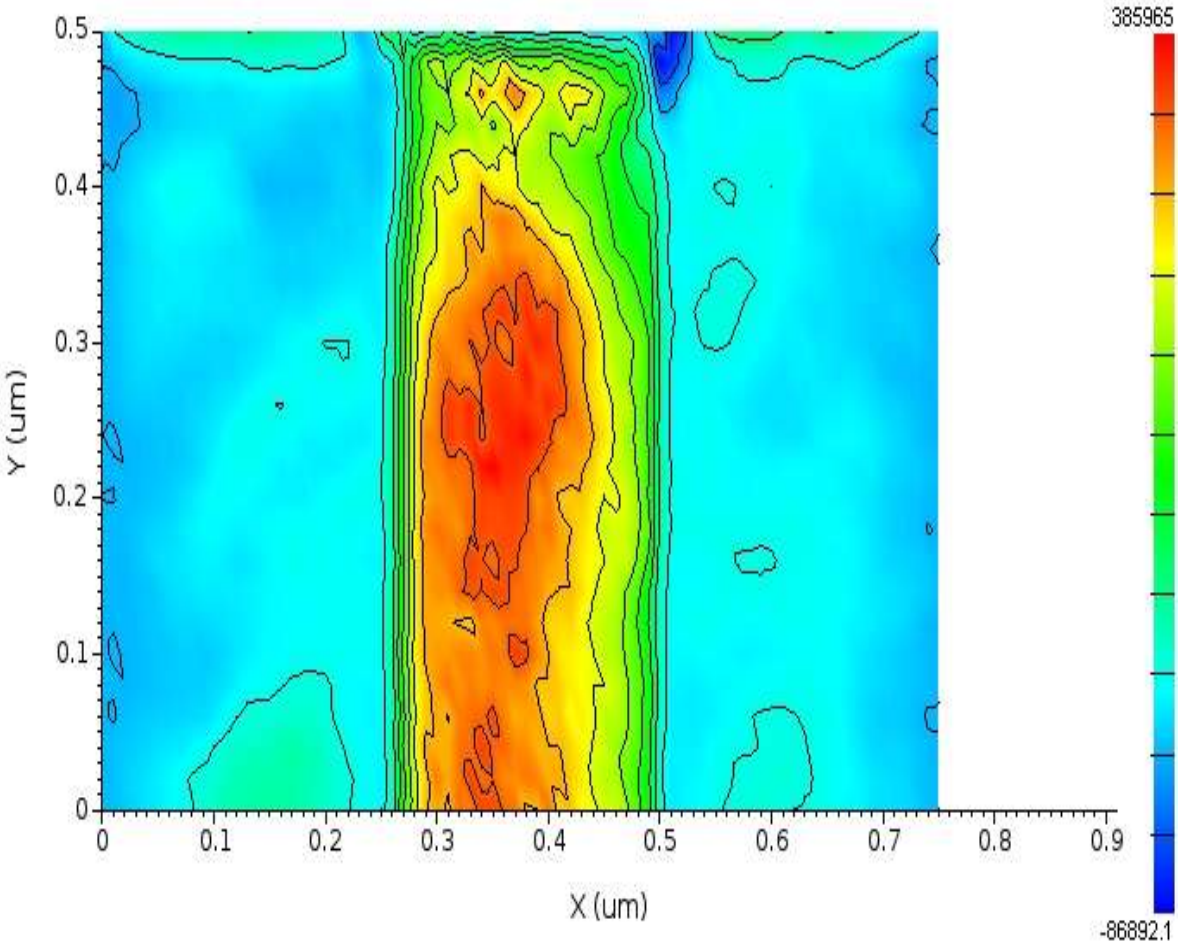

Fig 5.12: velocity (cm/s) profile in GaAs MESFET

Very high velocity of electrons is found in the channel region where carriers are moving rather freely than other regions. At the starting point of drain region channel is narrowed down hence limiting the carriers velocity .This region is called pinch off region. The peak negative velocity obtained (blue region) is 8.69×10^4 m/s due to differential mobility phenomenon caused by transfer mechanism of electrons to upper valley and peak positive velocity obtained by electrons is 3.859×10^5 m/s.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

Very high velocity of electrons is found in the channel region where carriers are moving rather freely than other regions. The negative velocity is obtained due to differential mobility phenomenon caused by transfer mechanism of electrons to upper valley and peak velocity obtained by electrons is 3.859×10^5 m/s

Maximum electric field intensity is found in source and drain regions of MOSFET. The negative electric field experienced by electrons is due to the inter valley transfer of electrons. The value of electric field obtained at the peak negative drift velocity is -0.466×10^6 V/m.

Differential mobility is found at the starting edge of drain region of MESFET and calculated to be as

$$v_d = \mu E$$

Where v_d : Average drift velocity of electrons

μ : Mobility of electrons in device

E : Electric field intensity inside device

$v_d = 3.859 \times 10^5$ m/s

$E = -0.466 \times 10^6$ V/m

Hence,

$\mu = 8281$ cm^2/V-s

Note: Simulations are performed at the room temperature.

6.2 FUTURE SCOPE

The approaches to device simulation seen above make use of equations obtained from the **BTE** without an explicit evaluation of the carrier distribution function. They provide a good understanding of the device functioning but cannot describe with sufficient accuracy a number of phenomena that depend critically on such a distribution. Typically, effects occurring above a threshold energy, as impact ionization, requires a more detailed analysis of electron transport. This analysis is provided, still within the semi classical approximation, by Monte Carlo (MC) simulations. To account for such phenomena, full-band MC simulations are, in genera necessary.

In ensemble MC simulations, when the number of particles in the device is too large, it is not possible to include all of them in the simulation. In future version of Archimedes it is necessary to solve Poisson equation, in the simulation, at very short time. Because the charge density fluctuations are damped by the self-consistent field and may give rise to plasma oscillations.

REFERENCES

[1]. H. Welker, Z. Naturforsch, Vol. A, No. 7, p. 744, 1952.

[2]. S. M. Sze, Semiconductor Devices Physics and Technology, J. Wiley, New York, 1985.

[3]. R. Williams, Modern GaAs Processing Methods, Artech House, Dedham, MA, 1990.

[4]. A. Abrikosov, L.P. Gorkov, I.E. Dzyaloshinski, Methods of Quantum Field Theory in Statistical Physics (Dover Publications, New York, 1975)

[5]. Non parabolic band transport in semiconductors: closure of the moment equations", A.M. Anile, V. Romano, Continuum Mechanics and Thermodynamics, 1999, 11:307-325

[6]. Non parabolic band transport in semiconductors: closure of the production terms in the moment equations", V. Romano, Cont.Mech.Thermodyn., 1999, 12:31-51

[7] 2D Simulation of a Silicon MESFET with a Nonpoarabolic Hydrodynamical Model Based on the Maximum Entropy Principle", V.Romano, Journal of Computational Physics, 176, 70-92 (2002)

[8] http://www.radio-electronics.com/info/data/semicond/fet-field-effect-transistor/gaasfet-mesfet-basics.php

[9] Numerical simulation of 2D Silicon MESFET and MOSFET described by the MEP based energy transport model with a mixed finite elements scheme", A.M. Anile, A. Marrocco, V. Romano, J.M. Sellier, Rapport de recherche, INRIA, N.5095.

[10] Numerical Simulation of the 2D Non-Parabolic MEP Energy-Transport Model with a Mixed Finite Elements Scheme", A. Marrocco, V. Romano, J.M. Sellier, N.5103.

[11] Two dimensional MESFET simulation of transients and steady state with kinetic based hydro dynamical models", A.M. Anile, S.F. Liotta, G. Mascali, S. Rinaudo.

[12] "Parabolic hydro dynamical model for bipolar semiconductors devices and low field hole mobility", G. Mascali, V. Romano, J.M. Sellier, submitted to Continuum Mechanics and Thermodynamics

[13] "Numerical Simulation of the 2D Non-Parabolic MEP energy-transport model with a mixed finite elements scheme", A.M. Anile, A. Marrocco, V. Romano, J.M. Sellier, submitted to Journal of Computational Electronics

[14] D.K.Ferry, R.Akis, D.Vasileska, "Quantum effects in MOSFETs: Use of an effective potential in 3D Monte Carlo simulation of ultra-short channel devices", IEDM Tech.Dig.,pp.287-290,2000

[15] S.M.Ramey, R.Akis, D.Vasileska, and D.K.Ferry, "Modeling of quantum effects in ultrasmall SOI MOSFETs with effective potentials", in Abst.of Silicon Nanoelectronics Workshop, 2001, pp.50-51.

[16] L.Shifren, R.Akis, and D.K.Ferry, "Correspondence between quantum and classical motion: Comparing Bohmian mechanics with a smoothed effective potential approach", Physics Letters A, vol.274, pp.75-83, Sep.2000.

[17]. E.Wigner, "On the Quantum Correction For Thermodynamics Equilibrium", Physical Review, Vol.40, pp.749-759, 1932

[18]. E.Wigner, "On the Quantum Correction For Thermodynamics Equilibrium", Physical Review, Vol.40, pp.749-759, 1932

[19]. A.Bertoni, P.Bordone, R.Brunetti and C.Jacoboni, "The Wigner function for electron transport in mesoscopic systems", J.Phys.: Condens.Matter 11 (1999), pp.5999-6012

[20]. E.Fatemi, F.Odeh, "Upwind finite difference solution of Boltzmann equation applied to electron transport in semiconductor devices", J.Comput.Phys. 108, (1993), pp.209-217

[21]. A.Majorana, R.Pidatella, "A finite difference scheme solving the Boltzmann-Poisson system for semiconductor devices, J.Comput.Phys., 174 (2001) pp.649-668

[22]. S.E.Laux, "On Particle-Mesh Coupling in Monte Carlo Semiconductor Device Simulation", IBM Research Report, RC 20101

[23]. S.E.Laux, "On Particle-Mesh Coupling in Monte Carlo Semiconductor Device Simulation", IBM Research Report, RC 20081

[24] http://www.gnu.org/software/archimedes/

[25]. K. Barnham, D. Vvedensky (eds.), Low-Dimensional Semiconductor Structures:

Fundamentals and Device Applications (Cambridge University Press, Cambridge, 2001)

APPENDIX I

SCRIPT OF GaAs MESFET

```
# This file simulate a GaAs MESFET.
# To run it type: archimedes mesfet.input

TRANSPORT MC ELECTRONS

FINALTIME 1.0e-12
TIMESTEP 2.e-15

XLENGTH 0.75e-6
YLENGTH 0.5e-6

XSPATIALSTEP 75
YSPATIALSTEP 25

# definition of the material (all the device is made of Gallium Arsenide)
MATERIAL X 0.0 0.75e-6    Y 0.0 0.5e-6  GAAS

# Definition of the Impurity Concentration
CIMP 7.e22

# Definition of the doping concentration
# ======================================
DONORDENSITY    0.       0.    0.75e-6   0.5e-6    7.e22
DONORDENSITY    0.       0.    0.25e-6   0.5e-6    1.e24
DONORDENSITY    0.5e-6   0.    0.75e-6   0.5e-6    1.e24
ACCEPTORDENSITY 0.       0.    0.75e-6   0.5e-6    1.e20

# Definition of the various contacts
# ==================================
CONTACT LEFT  0.0      0.50e-6 INSULATOR 0.0
CONTACT RIGHT 0.0      0.50e-6 INSULATOR 0.0
CONTACT UP    0.0      0.20e-6 OHMIC     0.0   1.e24
CONTACT UP    0.25e-6  0.50e-6 SCHOTTKY  -0.4
CONTACT UP    0.55e-6  0.75e-6 OHMIC     0.8   1.e24
CONTACT DOWN  0.0      0.75e-6 INSULATOR 0.0

NOQUANTUMEFFECTS
MAXIMINI
# SAVEEACHSTEP

LATTICETEMPERATURE 77.

STATISTICALWEIGHT 500

MEDIA 500

OUTPUTFORMAT GNUPLOT
# end of MESFET
```