GNU Archimedes, the Free Semicondutor Device Simulator

Documentation 0.0.1 for Archimedes 0.0.6

Jean Michel Sellier
Universit'a di Catania
Department of Mathematics and Computer Sciences
Via A.Doria 6, 95125 Catania, Italy
sellier@dmi.unict.it
www.gnu.org/software/archimedes

December 29, 2006

Contents

Chapter 1

Copying

This file documents for the **GNU Archimedes** program for simulation of submicron semiconductor devices.

Chapter 2

GNU Free Documentation License

Version 1.2, November 2002

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document ”free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of ”copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used

for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **”Transparent”** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not ”Transparent” is called **”Opaque”**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **”Title Page”** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ”Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section **”Entitled XYZ”** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **”Acknowledgements”**, **”Dedications”**, **”Endorsements”**, or **”History”**.) To **”Preserve the Title”** of such a section when you modify the Document means that it remains a section ”Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in

this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or

with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled ”Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled ”Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled ”Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination

all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled ”History” in the various original documents, forming one section Entitled ”History”; likewise combine any sections Entitled ”Acknowledgements”, and any sections Entitled ”Dedications”. You must delete all sections Entitled ”Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ”aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s

users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled ”Acknowledgements”, ”Dedications”, or ”History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Chapter 3

Why GNU Archimedes? A brief history...

Coding a working program is an astonishingly beautiful challenge. Every hacker who starts to write a code will have different reasons to do that. The followings are my reasons.

3.1 The Scientifical and Industrial Motivations

In today semiconductor technology, the miniaturization of devices is more and more progressing. In this context, it is easy to see that numerical simulations play an important role at every level of device manufacture. In fact, the cost of designing and physically constructing prototypes for VLSI semiconductor devices is very high and without the availability of advanced simulators the efforts for devices miniaturization would, likely, be brought to a halt. From assessing the performance of individual transistors, to circuits and systems, and, consequently, with the promise of improved device performance, industries are encouraged to keep on miniaturizing with lower manufacture costs.

But, unfortunately, such simulations are not whithout their challenges... A first consequence of device miniaturization is that simulations of submicron semicondutor devices requires advanced transport models. Because of the presence of very high and rapidly varying electric field, phenomena occur which cannot be described by means of the well-known drift-diffusion models, which do not incorporate energy as a dynamical variable. That is why some generalization has been sought in order to obtain more physically accurate models, like energy-transport and hydrodynamical models. The energy-transport models which are implemented in commercial simulators are based on phenomeno-

logical constitutive equations for the particle flux and energy flux depending on a set of parameters which are fitted to homogeneous bulk material Monte Carlo simulations. So, this is not, certainly, a satisfactory physical description of the internal electronic dynamics in a semiconductor device.

As current device technologies quickly approach the scales whereby quantum effects due to strong confinement of carriers and direct source-drain tunneling will begin to dominate, new simulation techniques are required in order to fully understand and acurately simulate the physics behind the technology operation.

Of all the simulation methods currently employed, ensemble Monte Carlo has always been, both in the accademic and industrial community, the most vigorous and trusted method for device simulation, as it is proven to be reliable and predictive, as one can easily see from the vast bibliography on this subject. However, as Monte Carlo relies on the particle nature of the electron (in fact we consider an electron like a 'biliard ball'), quantum effects associated with the wave-like nature of electrons cannot fully incorporated into the actual simulators, i.e. the ensemble Monte Carlo have to be lightly (or strongly, it depends on the point of view and on the methods implemented...) modified to take into account the quantum effects, at least at a first order of approximation, which is certainly enough to take into account correctly all the relevant quantum effects present in the present-day semiconductor devices (till 2015 probably...). In order to take into account the wave-like nature of electrons we use a recently introduced quantum theory, the so-called Bohm effective potential theory.

So it is challenging and very interesting to develop such a code for 2D quantum submicron semiconductor devices. This is why I have decided to implement this code, but these are not the only motivations...

3.2 The Ethical Motivations

The very sad situation you quickly observe working in a semiconductor industry, but also in all places in which researches about semiconductor devices are made, the only codes for simulation you can find are not free and are proprietary codes. That is a very bad situation because, at the present time, if you need to develop your own code for the purpose of simulating a device it is IMPOSSIBLE to obtain

an advanced one in a short time, and, trust me, this is EXTREMELY BAD for scientific research... (Immagine if you had to re-discover the Newtonian laws every time you need them...) So, you can find a huge amount of papers describing a lot of numerical methods for simulating, in a very advanced way, semiconductor devices (even in the quantum case), but nobody will give you a code on which you can construct your own method (with the unlikely exception that at least one of the programmers is a friend of yours :)).

Even worst, if you are a semiconductor device designer and you want to simulate ”realistically” a new device, you have to pay (trust me, at very high costs!) a BINARY (just a binary and not the code!) from some well-known software industry. This binary will certainly have some bugs (because it is coded by humans which are not perfect...) and you will never have the possibility of fix them on your own. Of course, you can write to the software house and tell them that there is a bug, but, how many time do you will wait for a new release without those bugs? I don’t think it will be a short time...

My impression is that, after a long research on the Web for a Free Software dealing with advanced 2D semiconductor device simulation, there was not a free code for the purpose of semiconductor devices simulation (i mean under GPL license). To be sure about it, I asked to the great Richard Stallman (by mail) if it will be worth to do a code like this and he encouraged me to code it, because there wasn’t a code like this as free. So I decided to write this code..

3.3 A Short Remark on Acknowledgments

If you use **GNU Archimedes** as a benchmark for your codes, or if you put some of the results obtained by it in your papers, it should be very nice if you write in your papers some aknowledgments or references to it.

For example, you can write a sentence like the followings

*”The author wants to thank the author of **GNU Archimedes**, Jean Michel Sellier, for giving it under GPL license...”*

or in your figure captions

*"This results have been obtained by **GNU Archimedes**..."*

or a reference to the web site of **GNU Archimedes**

*"You can download **GNU Archimedes** under GPL license at the following web site www.gnu.org/software/archim*

This will be very encouraging for me in developping new and more powerfull versions of **GNU Archimedes**. So, many thanks, for everybody will reference to me, or to **GNU Archimedes**, in his/her papers!

It will, also, be very nice if you send me a copy of your paper, in order to understand exactly in what contexts this code is being used. This will help me to understand in what direction I can go in the next versions. You can send any copy of your papers which use **GNU Archimedes** results at

`sellier@dmi.unict.it`

Every eventual comment, suggestion and advice on this code will be welcome.

3.4 Do you want to support GNU Archimedes?

GNU Archimedes has been developed in my free time and it took a huge amount of time for the development of a trustable and predictive code, a code which can certainly be an alternative to the proprietary simulation programs used in industry and/or research (certainly a better alternative if you to use a code on which construct your own code/method). So if you use it and you like it and you are an industry researcher, a university researcher, or a researcher in an other organisation, (or if you just want to encourage my effort in doing something important for the scientifical/industrial community) please think about the possibility of buying my DISTRIBUTION, in order to support and encourage my effort of making new versions of **GNU Archimedes**. It is very important for the community and for the future of the technology and science to support Free Software, so, in the case you want the distribution or you just want to make a donation write to me at :

`sellier@dmi.unict.it`

You will receive the distribution which means that you will receive the last version of the printed manual along with the last version of **GNU Archimedes** code (SOURCES and binaries). Furthermore, if

you give your email address, you will be informed by the author (me) in case of new downloadable releases :)

Chapter 4

Introduction

4.1 Overview

The GNU **GNU Archimedes** program is a 2D Quantum Monte Carlo simulator for semiconductor devices. At the present time it can simulate the transport of electrons and Holes only in Silicon and GaAs devices but, in the next versions, i will develop the possibility of choosing different materials like Germanium, InP and so on (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 zincblende structure along with all the heterostructures possible). In this version **GNU Archimedes** can simulate electrons for both Silicon and GaAs material and heavy holes for Silicon. In the future i will also develop the heavy and light holes transport for both Silicon and GaAs.

GNU 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 **GNU Archimedes** you 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 accademic community, as you can see from scientifical papers. Furthermore, up to the release 0.0.4, you can simulate a simplified MEP (Maximum Entropy Principle) model which is very usefull for making Archimedes faster than the precedent release, as you will see in the next chapters. This is a completely new results which is under publication.

All the particles in this code have a ‘statistical weight’ which is made a piecwise-function of the position. You 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 milions. If you want a bigger number you have to change it in the code (modifying the definition of NPMAXIMUM in the file ”archimedes.c” and recompiling it). I have choosed 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 enormousely tax the velocity of the simulation, which is very undesirable in a Monte Carlo simulation!

GNU **GNU Archimedes** was written by Jean Michel Sellier (sellier@dmi.unict.it). Because it is protected by the GNU General Public License, users are free to share and change it. You can download it at the following web page: www.gnu.org/software/archimedes

Let us see, now, a first example of definition for a device...

4.2 A First Example: The n^+-n-n^+ Diode

In this section I introduce a first example of semiconductor device simulated by **GNU Archimedes** in order to show how it is easy to define a new general device. Let us report, in the following, the definition of a device which is the benchmark in semiconductor simulation: the n^+-n-n^+ Silicon diode.

```
# Silicon DIODE test-1
# created on 30 sep.2004, J.M.Sellier
# modified on 06 oct.2004, J.M.Sellier
```

```
# This file simulate a Silicon Diode.
# To run it type:
# archimedes diode.input

MATERIAL SILICON
TRANSPORT MC ELECTRONS

FINALTIME 5.5e-12
TIMESTEP 0.0015e-12

XLENGTH 1.0e-6
YLENGTH 0.1e-6

XSPATIALSTEP 100
YSPATIALSTEP 25

# Definition of the doping concentration
# ======================================
DONORDENSITY    0.       0.    1.0e-6    0.1e-6    2.e21
DONORDENSITY    0.       0.    0.3e-6    0.1e-6    5.e23
DONORDENSITY    0.7e-6   0.    1.0e-6    0.1e-6    5.e23
ACCEPTORDENSITY 0.       0.    1.0e-6    0.1e-6    1.e20

# Definition of the various contacts
# ==================================
CONTACT LEFT  0.0    0.1e-6 OHMIC     0.0    5.e23
CONTACT RIGHT 0.0    0.1e-6 OHMIC     1.0    5.e23
```

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

NOQUANTUMEFFECTS
MAXIMINI
# SAVEEACHSTEP

LATTICETEMPERATURE 300.

STATISTICALWEIGHT 1500

MEDIA 500

OUTPUTFORMAT GNUPLOT

# end of MESFET test-1
```

The name of this file is ”diode.input” and you can find it in the distribution directory :

```
archimedes-0.0.4/tests/DIODE
```

If you run it by typing in the shell command line

```
# archimedes diode.input
```

You will get, after the computation and plotting the results, the pictures (which are cuts on y=0.05 micron) you can find in figures (4.1)-(4.5).

As we will see soon, it is very easy to define a new semiconductor device. First of all, we see that the rows starting by the symbol ”#” are just comments. Let us analyze, now, some keywords present

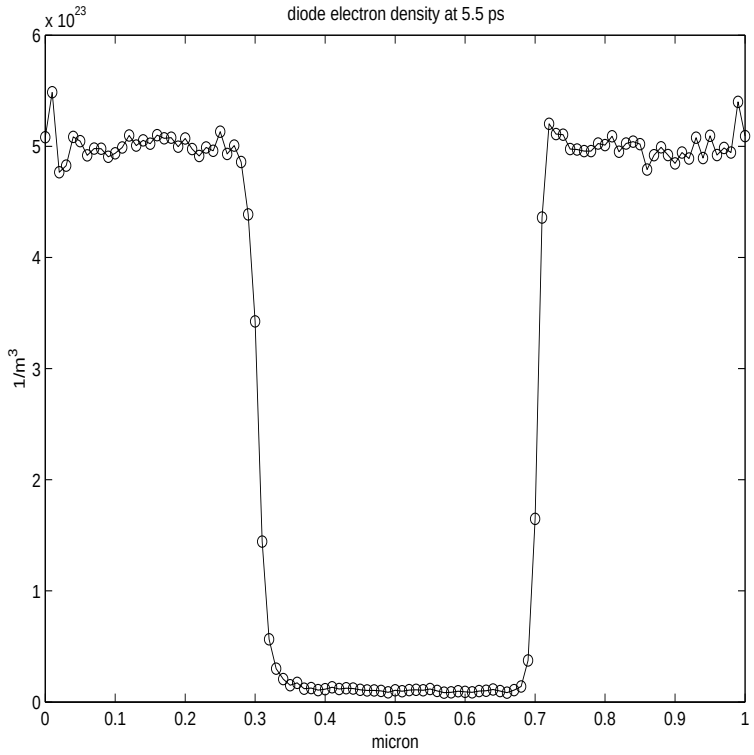


Figure 4.1: Density Profile obtained by **GNU Archimedes**

in this example. For more informations about all the syntax commands of **GNU Archimedes** you must read the chapter related on this topic.

1. The keyword **MATERIAL**. This keyword is easy to understand. Invoking it, you choose the material of which your device is made. At the present time, **GNU Archimedes** accept only the Silicon and GaAs materials, but in the next future you will can choose other materials. So, for the moment, the only correct uses you can do of **MATERIAL** are the following :

```
MATERIAL SILICON
```

or

```
MATERIAL GAAS
```

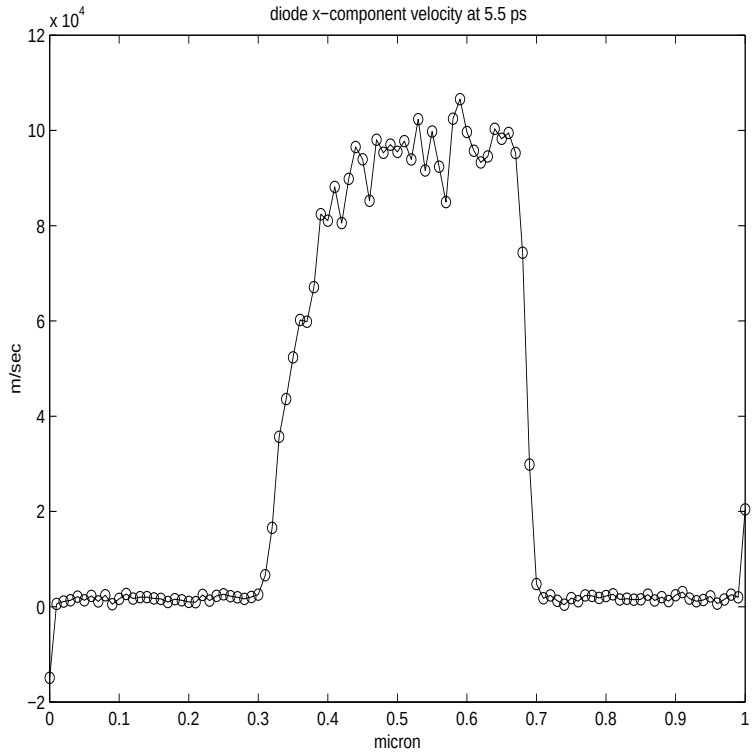


Figure 4.2: Electron Velocity obtained by **GNU Archimedes**

2. The keyword **TRANSPORT**. Also this one is easy to understand. By this command, you choose what kind of charge transport you want, including the mathematical model for the transport (Monte Carlo or simplified model). In this case you can choose between only electrons, only holes, or bipolar transport (up to the release 0.0.4 it is possible to simulate all this particles). Pay attention to the fact that the syntax for this command has been changed in the release 0.0.4.

3. The keyword **FINALTIME**. Nothing is easier to understand :) By this command, you choose the final time at which you want to stop the simulation and save the results.

4. The Keyword **TIMESTEP**. When you start a simulation and you have to reach the choosed final time, you have to proceed by time step. The time step you choose have to respect an appropriate condition, i.e. it must be not too big, in order to avoid unphysical effects during the simulation. We will describe this conditions better in a next paragraph.

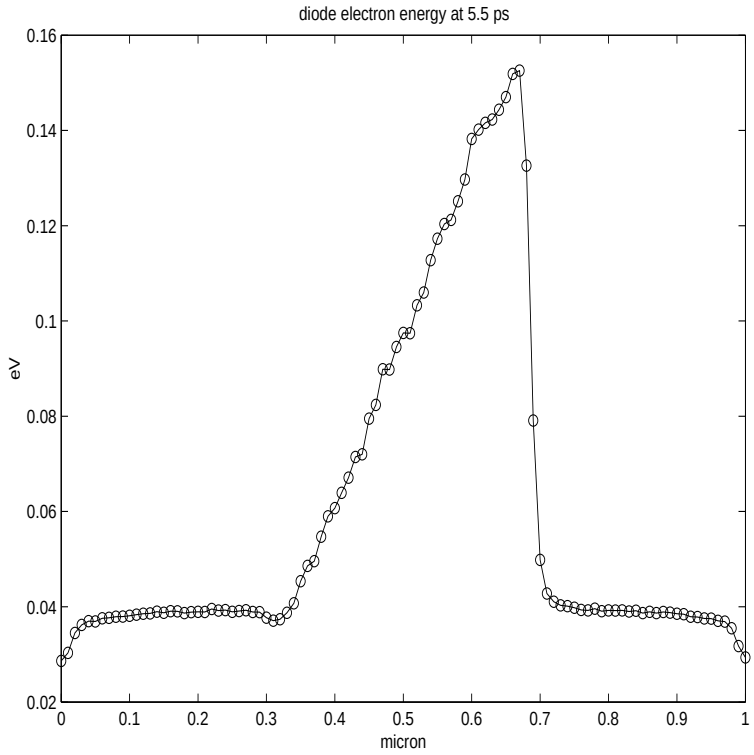


Figure 4.3: Electronic Energy obtained by **GNU Archimedes**

5. The Keywords **XLENGTH** and **YLENGTH**. Also these keywords are easy to understand. If one think of our simulated device as a simple rectangle, then we have to specify the length of the edges in the x-direction and the y-direction. This is done by these two keywords. Even if not all semiconductor devices are rectangular, in this first release of **GNU Archimedes** we can simulate only rectangular domains. This will be improved in some next version.

6. The keywords **XSPATIALSTEP** and **YSPATIALSTEP**. These commands are invoked when one want to define the number of cells in the x-direction and the y-direction of a rectangular domain.

7. The keyword **DONORDENSITY**. This keyword is needed when one wants to define a rectangular sub-domain in which one defines a certain donor density. This command is more complex than the precedent one (but, don't worry, quite easy to understand). See the paragraph related to this command to known more about it.

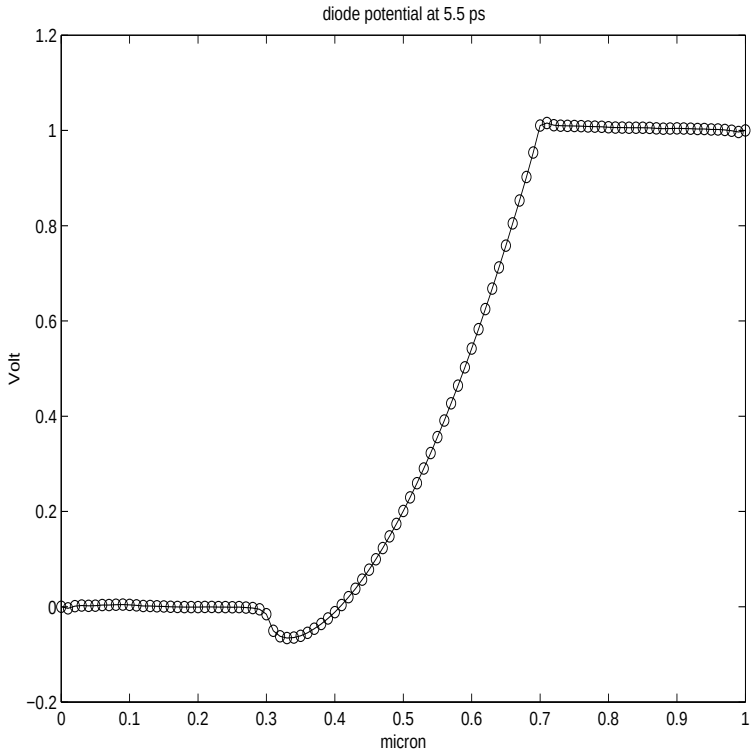


Figure 4.4: Potential Profile obtained by **GNU Archimedes**

8. The keyword **ACCEPTORDENSITY**. This is the same as the precedent keyword (i.e. **DONOR-DENSITY**), but for the holes. Pay attention to the fact that even if you are not simulating the transport of holes, you need to fix a value of the holes concentration on the devices. This is needed by the Poisson equation which take into account the accpetor density, in order to solve accurately the electric field. See the paragraph related to this topic for more informations.

9. The keyword **CONTACT**. By this command, you choose where are positioned the various contacts. Furthermore, you can specify what kind of contact it is, chossing among ohmic, insulator and Schottky contact. When desired you can specify the applied voltage on the contact.

10. The keyword **NOQUANTUMEFFECTS**. This command is invoked when you know, a priori, that the quantum effects are negligible, and so you don't want **GNU Archimedes** compute them (for example when the electronic wave lenght is negligible with respect to the caracteristic length of the device). This is important when you want to get a smaller run-time with respect

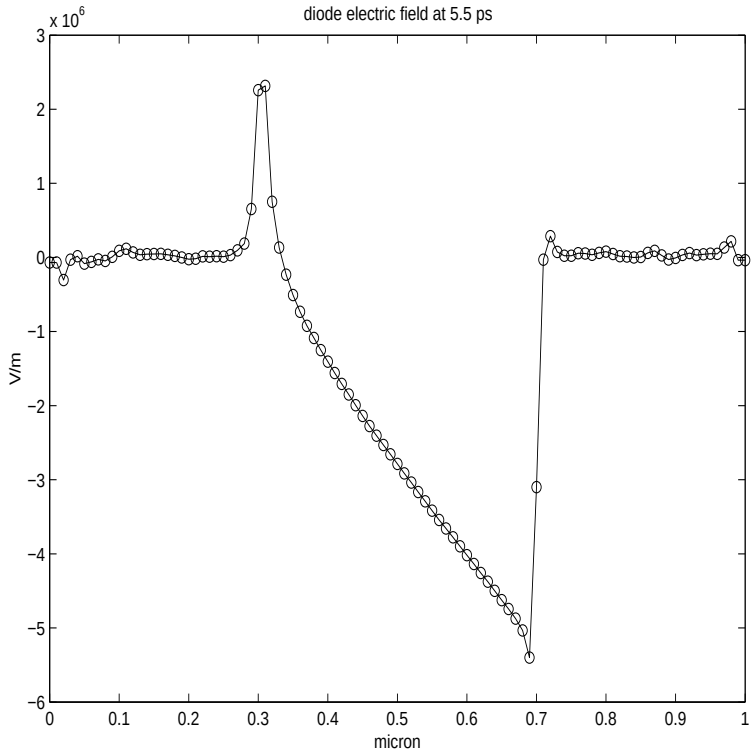


Figure 4.5: Electric Field obtained by **GNU Archimedes**

to the run-time of a full-effects simulation.

11. The keyword **MAXIMINI**. This is a simple command for the visualisation of the maximum and minimum of some macroscopic values in real-time, i.e. during the simulation. Avoid to invoke this command if you want to obtain a shorter run-time.

12. The keyword **SAVEEACHSTEP**. This is needed when you want to save all the solutions at each time step. This is a very usefull command in the case you want to follow the transient behaviour of a device in real-time or when you want to create a movie showing the transient dynamics of a simulated device.

13. The keyword **LATTICETEMPERATURE**. It is easy to understand that you have to fix a temperature of the lattice of the device simulated. This is done by this simple command.

14. The keyword **STATISTICALWEIGHT**. By this command you choose the statistical weight of

the particles. Pay attention to the fact that the statistical weight is a piecewise-function of the position, so the statistical weight coincides to the number of super-particles per cell only in the most doped sub-domain of the device.

15. The keyword **MEDIA**. As **GNU Archimedes** is a Monte Carlo simulator, it is impossible to avoid the noise in the solutions. The noise is intrinsic to the method. So, in order to get less noisy solutions, it is necessary to take an average mean in time of the values simulated. Fixing a value to MEDIA means that you will take the mean average of the solution over the last MEDIA time steps.

16. The keyword **OUTPUTFORMAT**. By this command, you choose the format of the output, i.e. the output files generated by **GNU Archimedes** during or the end of a simulation.

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

Chapter 5

Physical Models employed in GNU Archimedes

This chapter describes the physical models employed in **GNU Archimedes**. It is important to fully understand this chapter in order to fully exploit the possibility offered by this code. Obviously, everything here is a brief review of what you can find in papers and books on the Monte Carlo subject. So, if you don't understand some aspects of this chapter, don't esitate to read the papers reported in this chapter. We will, also, describe the simplified MEP (Maximum Entropy Principle) model, which is a simplified version of the MEP model. This model has been developped by A.M.Anile and V.Romano, two professors of the Department of Mathematics and Computer Sciences of the University of Catania. It is a very good model, which is able to give very accurate results, compared to the other hydrodynamical models.

In the present release of **GNU Archimedes** we use a simplified version because it is enough for our purpose, i.e. the coupling of MEP model and Monte Carlo method in order to obtain very accurate simulation results in very short running times.

In the following we report a (i hope) complete list of usefull papers for people interested in the complete MEP model:

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

"Non parabolic band transport in semiconductors: closure of the production terms in the moment

equations", V. Romano, Cont.Mech.Thermodyn., 1999, 12:31-51.

"Non-parabolic band hydrodynamical model of silicon semiconductors and simulation of electron devices", V.Romano, Mathematical methods in the applied sciences, 2001, 24:439-471.

"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)

"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.

"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.

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

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

"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.

5.1 The Semiclassical Approach

When we talk about semiclassical approach then we talk about a well-defined group of approximations. Let us briefly review these last ones in the following list.

1. **Quantum size effects**. First of all, 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-localised in the phase-space. In this case, we can consider the particles as "biliard balls".

2. **Slow Physical Phenomena**. This means that the phenomena simulated in **GNU Archimedes**, i.e. the dynamics of electrons or holes, are significantly slow with respect to the dynamics of the electric and/or magnetic field. This means that we work in a physical context in which it is justified to use electrostatic instead of full set of Maxwell's equations. Then we simulate 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 we will adopt in **GNU Archimedes**, in order to take into account the effects of the lattice on the particles.

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

5. **The Many Body effects**. In our simulations, we neglect the Pauli principle, i.e. all the particles in the simulation have interation with each other. Probably, the Pauli principle will be taken into account in a next version of **GNU Archimedes**, even if it seems, from experiments, that in the real world and for enough diluted doping concentrations, the electrons don't interact with other electrons, i.e. there are no collisions between electrons.

5.2 The Quantum Effects

Concerning the quantum effects, while we consider the particles as semiclassical objects, we want to have the possibility of simulating relevant quantum effects in today manufactured semiconductor devices (like the diode tunnel, or the nanoscopic MOSFETs). This is a quite difficult goal, 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 an other approach than the Wigner equation one.

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 Schroedinger, 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

$$\nabla \cdot [\epsilon(\mathbf{x})\nabla\phi_{cl}(\mathbf{x},t)] = -q[N_D(\mathbf{x}) - N_A(\mathbf{x}) - n(\mathbf{x},t) + p(\mathbf{x},t)] \tag{5.1}$$

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

$$\phi_{quant} = \frac{1}{\sqrt{2\pi}a}\int_{\mathbb{R}^n} \phi_{cl}(\mathbf{x}+\xi,t)exp(-\frac{\xi^2}{2a^2})d\xi \tag{5.2}$$

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

For more information about this recent method, see the following papers:

1. D.K.Ferry, R.Akis, D.Vasileska, "Quantum effects in MOSFETs: Use of an effective poten-

tial in 3D Monte Carlo simulation of ultra-short channel devices", IEDM Tech.Dig.,pp.287-290,2000

2. 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.

3. 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.

5.3 The Particle Dynamics

The purpose of **GNU Archimedes** 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)

$$\frac{\partial f}{\partial t} + \frac{1}{\hbar}\nabla_{\mathbf{k}}\mathcal{E}\cdot\nabla_{\mathbf{x}}f - \frac{q}{\hbar}\mathbf{E}\cdot\nabla_{\mathbf{k}}f = \mathcal{C}[f](\mathbf{k},\mathbf{x},t) \tag{5.3}$$

$$\frac{\partial w}{\partial t} + \frac{\hbar\mathbf{k}}{m^*}\nabla_{\mathbf{x}}w\cdot\nabla_{\mathbf{x}}w - \frac{1}{2\pi\hbar}\int_{\mathbb{R}} d\mathbf{k}^{'}V^W(\mathbf{k}-\mathbf{k}^{'},\mathbf{x})w(\mathbf{k}^{'},\mathbf{x},t) = \mathcal{C}^W[w](\mathbf{k},\mathbf{x},t) \tag{5.4}$$

with

$$V^W(\mathbf{k},\mathbf{x}) = i\int_{\mathbb{R}} d\eta exp(i\eta\cdot\mathbf{k})(V(\mathbf{k},\mathbf{x}+\frac{\eta}{2}) - V(\mathbf{k},\mathbf{x}-\frac{\eta}{2})) \tag{5.5}$$

where $f = f(\mathbf{k},\mathbf{x},\mathbf{t})$ is the Boltzmann probability density function, $w = w(\mathbf{k},\mathbf{x},\mathbf{t})$ the Wigner probability density function, $\mathbf{k}$ the particle pseudo-wave vector, $\mathbf{x}$ the position vector, $\imath$ the immaginary unity, $V(\mathbf{x},\mathbf{t})$ the classical electrostatic potential, $\mathbf{E} = -\nabla\phi$ the classical electric field, $\mathcal{E} = \mathcal{E}(\mathbf{k})$ the energy band relation. The operators $\mathcal{C}[f]$ and $\mathcal{C}^W[w]$ 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.

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

For more informations about the precedent two equations, please refer to the following papers

1. E.Wigner, ”On the Quantum Correction For Thermodynamics Equilibrium”, Physical Review, Vol.40, pp.749-759, 1932

2. 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

3. 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

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

5.3.1 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 V_L generated by the ions located at the points of the crystal lattice L. The Schroedinger equation is

$$H\psi = \mathcal{E}\psi \tag{5.6}$$

with the Hamiltonian H given by

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

Then, this theorem states that the bounded eigenstates have the following form:

$$\psi(\mathbf{x}) = \exp(\imath\mathbf{k}\cdot\mathbf{x})\mathbf{u}_\mathbf{k}(\mathbf{x}) \tag{5.7}$$

and

$$u_k(\mathbf{x} + \mathbf{X}) = \mathbf{u_k}(\mathbf{x}) \tag{5.8}$$

with $\mathbf{x}$ beloging to L. Furthermore, it is possible to prove the existence of an infinite sequence of eigenpairs of solutions

$$\mathcal{E}_l(\mathbf{k}), \mathbf{u_{k,l}}$$

with l belonging to the non negative integers set $\mathbb{N}$. The function $\mathcal{E} = \mathcal{E}_l(\mathbf{k})$ 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 **GNU Archimedes** 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

$$\mathcal{E}(k)[1 + \alpha\mathcal{E}(k)] = \frac{\hbar^2 k^2}{2m^*} \tag{5.9}$$

where α is the non-parabolicity parameter.

It is possible to choose other energy band relations, but they are actually not implemented in **GNU Archimedes**. Anyway, it seems, from precedent experiment simulations, that the Kane dispersion relation is the best choice if we consider the accuracy of the electron energy and velocity along with velocity of computation.

5.3.2 The Drift Process

As seen previously, an electron moving in a crystal lattice moves just like a free electron, but with a change of mass. This fact justify us to use the classical equations of motion, in order to describe the motion of electrons and holes in a semiconductor device. We can, thus, use the Hamilton formalism

to get the electron equations of motion. They read as follow

$$\frac{d\mathbf{x}}{dt} = \frac{1}{\hbar}\nabla_{\mathbf{k}} H \tag{5.10}$$

$$\frac{d\mathbf{k}}{dt} = -\frac{1}{\hbar}\nabla_{\mathbf{x}} H \tag{5.11}$$

where H is the Hamiltonian of the system, i.e.

$$H = \mathcal{E}(\mathbf{k}) + \mathbf{V}(\mathbf{x})$$

Then, if we use the Kane dispersion relation, we get, after some simple algebra, the following expression for the electron velocity

$$\mathbf{v} = \frac{\hbar\mathbf{k}}{m^*}\frac{1}{\sqrt{1+4\alpha\frac{\hbar^2k^2}{2m^*}}} \tag{5.12}$$

5.4 Initial Conditions

In this paragraph, we explain how **GNU Archimedes** specifies the initial conditions for the superparticles. 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 consider all the particles, at the initial time of the simulation, nearly the thermal equilibrium, which means that the energy of a particle reads

$$\mathcal{E}(k) = -\frac{3}{2}k_B T_L ln(r) \tag{5.13}$$

where r is a random number between 0 and 1.

Once we have specified the energy of the electrons, then we can choose the pseudo-wave vectors of all particles. This is done, trivially, by the following algorithm.

1. We can, from the Kane dispersion relation, compute the modulus of the pseudo-wave vector. This is done by the following expression

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

2. We, then, generate two random numbers between 0 and 1, say θ and ϕ

3. We compute the three component of the pseudo-wave vector by

$$k_x = k \sin\theta \cos\phi \tag{5.14}$$

$$k_y = k \sin\theta \sin\phi \tag{5.15}$$

$$k_z = k \sin\theta \tag{5.16}$$

5.5 Contacts and Boundaries

In this section, we describe the various contacts we can choose in **GNU Archimedes**. First of all, we can divide the contacts into three categories: Insulator boundary, Ohmic and Schottky contacts. In this code we immagine a contact, or a boundary, as a line on an edge of the device. In **GNU Archimedes** there is no limit in the number of contacts, so you can specify whatever contacts you want.

5.5.1 Insulator Boundaries

This kind of boundary is considered as a ”mirror boundary”. This means that when a particle interact with such a contact it will simply be reflected by this one. This is necessitated when you want to simulate, for example, the insulator boundaries of a device. Furthermore, you can apply a non-zero potential on a boundary even if this is an insulator edge. This will be explained further in the paragraph concerning the keyword **CONTACT**.

5.5.2 Ohmic Contacts

Ohmic contacts are contacts which are open, i.e. particles can both go out from or enter in the device trough it, but also contacts which hold the neutrality charge condition, i.e. the charge have to be assigned constant on it. Thus we can be think of ohmic contacts as electron reservoirs from which the particles can go out from the device.

5.5.3 Schottky Contacts

Schottky contacts are, as the ohmic ones, open contacts but they don't have an electron reservoir, which means that electrons (or more generally, particles) can go out through this it, but they are just absorbing contacts, so the neutrality charge condition have to hold on it.

5.6 The Scattering Process

At the term of the free flight of a particle, this one scatters with the phonons of the lattice (phonons are the quantization of the motion of ions of the lattice). So, at the end of the free flight, a scattering process have to be choosed. Let us see, how this happens in **GNU Archimedes**. First of all, let us report the list of all scatterings taken into account in **GNU Archimedes**. We note that, while the self-scattering is computed simply, the probability of scattering for acoustic and optical phonons are computed by means of the quantum mechanics and we will show all the details about them.

1. **Self-Scattering**. We introduce this scattering in order to determine the flight time. It is important to accurately compute this scattering, because it influences all process during the simulation. For more informations about this topic, read the book of K.Tomizawa, "Numerical Simulation of Submicron Semiconductor Devices", Artech House, Boston, London. Let us report, briefly, how the self-scattering is introduced in the simulation. If the various scatterings read

   $$\mathcal{W}_i(\mathcal{E}(k))$$

   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

   $$\Gamma = \sum_{i=0}^{N} \mathcal{W}_i(\mathcal{E}(k)) \tag{5.17}$$

   Then the free flight τ of a particle will read

   $$\tau = -\frac{ln(r)}{\Gamma} \tag{5.18}$$

where r is a random number between 0 and 1. The factor Γ will be used, as we will see at the end of this paragraph, to determine when the self-scattering occurs.

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 $\mathbf{k}$ scatters with an elastic acoustic phonon and having a final pseudo-wave vector $\mathbf{k}^{'}$, is

$$S(\mathbf{k},\mathbf{k}^{'}) = \frac{\pi \Xi^2 k_B T_L}{\hbar c_L \Omega} \frac{k}{q_w \mathcal{E}(k)} \delta(\frac{q_w}{2k} \pm \cos\theta^{'}) \tag{5.19}$$

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

$$\mathcal{W}(\mathbf{k}) = \frac{2\pi \Xi^2 k_B T_L}{\hbar c_L} N(\mathcal{E}_k) \tag{5.20}$$

where $N(\mathcal{E})$ is the density of states and reads

$$N(\mathcal{E}(k)) = \frac{(2m^*)^{\frac{3}{2}} \sqrt{\mathcal{E}(k)}}{4\pi^2 \hbar^3} \tag{5.21}$$

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

$$S(\mathbf{k},\mathbf{k}^{'}) = \frac{\pi D_{opt}^2}{\rho \omega_0 \Omega}(n_0 + \frac{1}{2} \mp \frac{1}{2}) \delta(\frac{\hbar^2 q_w^2}{2m^*} \pm \frac{\hbar^2 k q_w \cos\theta^{'}}{m^*} \mp \hbar\omega_0) \tag{5.22}$$

$$\mathcal{W}(\mathbf{k}) = \frac{\pi D_{opt}^2}{\rho \omega_0}(n_0 + \frac{1}{2} \mp \frac{1}{2}) N(\mathcal{E}(k) \pm \hbar\omega_0) \tag{5.23}$$

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. Pay attention that in **GNU Archimedes** we take into all the six optical phonons for the Silicon material. For more informations about this topic read the following paper C.Jacoboni, L.Reggiani, "The Monte Carlo method for the solution of charge transport insemiconductors with applications to covalent materials", Reviews of Modern Physics, Vol.55, No.3, July 1983

5.6.1 The Choice of the Scattering

In **GNU Archimedes** 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

$$\Lambda_i(\mathcal{E}) = \frac{\sum_{j=1}^{n} \mathcal{W}_j(\mathcal{E})}{\Gamma} \tag{5.24}$$

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, choosed generating a number r lying between 0 and 1 and doing the following comparison

$$\Lambda_{i-1}(\mathcal{E}) < r <= \Lambda_i(\mathcal{E}) \tag{5.25}$$

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

5.7 The Simplified MEP Model

From the vast literature, it is well known that Monte Carlo method is the best way for obtaining very accurate simulations for the transport of electrons in semiconductor devices. Even if this method is very accurate, it has the price of very high simulation time. This is why we introduce the MEP model in this release of **GNU Archimedes**. In this way, as we will see, we will have the possibility of coupling MEP and Monte Carlo in order to make Monte Carlo method faster.

As you can see from the precedent papers reported in the introduction of this chapter, the MEP model is a very advanced hydrodynamical model for both electrons and holes in Silicon devices. For our purpose we will need only a simplified version of it. This because, we only need simple initial conditions for the Monte Carlo method. In the following we report a sketch of this model. A paper is under construction and will be refered in the next versions of this manual. The MEP model is based on the closure of the semiclassical Boltzmann equation by means of the maximum entropy principle. 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

$$f = (\frac{3\hbar^2}{4\pi m^* W})(\frac{3}{2})ne^{-\frac{3\epsilon}{2W}} + \frac{27\hbar^3}{32\pi\sqrt{2m^*}}\frac{n}{W}\epsilon^{-\frac{1}{2}}e^{-frac3\epsilon 2W}u^i v_i \tag{5.26}$$

we get the following hydrodynamical model for electrons, which we will call the **Simplified MEP model**.

$$\frac{\partial n}{\partial t} + \frac{\partial n u^i}{\partial x^i} = C_n \tag{5.27}$$

$$\frac{n u^j}{\partial t} + \frac{n K_B T^{ij}}{\partial x^i} + qE^j\frac{n}{m^*} = -\frac{nu^i}{\tau_p} \tag{5.28}$$

$$\frac{\partial nW}{\partial t} + \frac{\partial nS^i}{\partial x^i} + qE_i n u^i = -n\frac{W - W_0}{\tau_W} \tag{5.29}$$

where τ_W is a function of the electrons energy, as you can see from the precedent papers. This function is computed numerically and reads:

$$\tau_W(W) = -\frac{W - \frac{3}{2}K_B T_L}{-7.24\times 10^{10}W^5 + 5.13\times 10^{11}W^4 - 1.36\times 10^{12}W^3 + 3.69\times 10^{11}W^2 - 3.07\times 10^{12}W + 9.97\times} \tag{5.30}$$

Furthermore, we have the following relations:

$$K_B T^{ij} = \frac{2}{3}\frac{W}{m^*}\delta^{ij} \tag{5.31}$$

$$S^i = \frac{4}{3}Wu^i \tag{5.32}$$

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

$$\tau_p = \frac{k_{\tau_p}}{T} \tag{5.33}$$

where

$$k_{\tau_p} = \frac{m^*\mu_0 T_L}{q} \tag{5.34}$$

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.

5.7.1 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 as it will be possible to see in the example reported in a next chapter. 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. What it is done in **GNU Archimedes** is very simple. 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, surprinsingly, 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. The results are very interesting, as you will see in a next chapter in which we report it, and are the subject of a paper under preparation.

Chapter 6

Coupling between Monte Carlo and Poisson

6.1 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 Schroedinger-Poisson system. Even if some works have been done on this topic, it remains a very difficult and, somehow, open problem. For more informations, read the following very interesting book

L.Ramdas Ram-Mohan, ”Finite Element and Boundary Element Applications in Quantum Mechanics”, Oxford University Press, 2002

Important problems still remains in the application of the boundary conditions for the Schroedinger equation, and it is very difficult, and for some process still impossible, to take into account all the relevant scattering events. Furthermore, solving Schroedinger-Poisson system is a very difficult task also from the numerical point of view, since the Schroedinger 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 use the Monte Carlo method for simulations in **GNU Archimedes**.

In Monte Carlo electron gas simulations, it is very important 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. This is why, in this chapter, we will explain how the Monte Carlo simulations and the Poisson equation are coupled in **GNU Archimedes**. This is an important topic for Monte Carlo simulations.

6.2 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 generated by **GNU Archimedes**. That is why we have 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. For this purpose, we use in **GNU Archimedes** the well-known cloud-in-cell algorithm. For more informations about cloud-in-cell method and more advanced algorithms see the following papers

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

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

We report here a brief description of the cloud-in-cell method.
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

$$n_{i,j} = \frac{S_{i,j}}{A^2_{i,j}}(x_{i+1} - x)(y_{j+1} - y) \tag{6.1}$$

$$n_{i+1,j} = \frac{S_{i+1,j}}{A^2_{i+1,j}}(x - x_i)(y_{j+1} - y) \tag{6.2}$$

$$n_{i,j+1} = \frac{S_{i,j+1}}{A^2_{i,j+1}}(x_{i+1} - x)(y_{j+1} - y) \tag{6.3}$$

$$n_{i+1,j+1} = \frac{S_{i+1,j+1}}{A^2_{i+1,j+1}}(x - x_i)(y - y_j) \tag{6.4}$$

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 heterostructures. They are not implemented, at the present time, in **GNU Archimedes** because do not yet deal with such structures. They will be implemented in the next future.

6.3 The Stationary Poisson Equation

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

1. We can neglect the computation of the magnetic field (at least for the majority of semiconductor devices, but certainly for the Silicon devices)

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.

We, thus, report the Poisson equation

$$\nabla \cdot [\epsilon(\mathbf{x}) \nabla \phi(\mathbf{x}, t)] = -q[N_D(\mathbf{x}) - N_A(\mathbf{x}) - n(\mathbf{x}, t) + p(\mathbf{x}, t)] \tag{6.5}$$

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, because the boundary conditions are difficult to implement in a generic simulator such as textbfGNU Archimedes and, furthermore, this algebraic system is consuming from the point of view of computer memory (even if we can use well-known methods applied to sparse matrices).
These reasons have influenced the author of **GNU Archimedes** to adopt a lightly different approach in the simulation of the electrostatic potential. (And for this, I thanks Vittorio Romano for his excellent advices).

6.4 The Non-Stationary Poisson Equation

We introduce in this section what I call the "non-stationary" Poisson equation, hereafter NSP equation. This equation is very easy to implment, in a very general numerical context, and very easy to and solve with simple, but very robust, numerical schemes. Let us report in the following the NSP equation

$$\frac{1}{k_S}\frac{\partial \phi}{\partial t} + \nabla \cdot [\epsilon(\mathbf{x})\nabla\phi(\mathbf{x},t)] = -q[N_D(\mathbf{x}) - N_A(\mathbf{x}) - n(\mathbf{x},t) + p(\mathbf{x},t)] \tag{6.6}$$

where k_S is a constant for giving the right dimensions of the term $\frac{\partial \phi}{\partial t}$, and the other variables have the usual meaning. It is very easy to show that 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 will 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 will see in the next paragraph.

6.4.1 Numerical Resolution of the NSP Equation

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

$$\frac{\partial \phi}{\partial t}(x_i, y_j) = \frac{\phi_{i,j}^{n+1} - \phi_{i,j}^{n}}{\Delta t} \tag{6.7}$$

$$\nabla^2\phi(x_i, y_j, t^n) = \frac{\phi_{i+1,j}^{n} - 2\phi_{i,j}^{n} + \phi_{i-1,j}^{n}}{\Delta x^2} + \frac{\phi_{i,j+1}^{n} - 2\phi_{i,j}^{n} + \phi_{i,j-1}^{n}}{\Delta y^2} \tag{6.8}$$

where $\phi_{i,j}^{n}$ 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, one get the following numerical scheme

$$\phi_{i,j}^{n+1} = \phi_{i,j}^{n} + \Delta t(-\epsilon_{i,j}(\frac{\phi_{i+1,j}^{n} - 2\phi_{i,j}^{n} + \phi_{i-1,j}^{n}}{\Delta x^2} + \frac{\phi_{i,j+1}^{n} - 2\phi_{i,j}^{n} + \phi_{i,j-1}^{n}}{\Delta y^2}) - q[N_{Di,j} - N_{Ai,j} - n_{i,j}^{n} + p_{i,j}^{n}]) \tag{6.9}$$

Note that the presented scheme is valid only in the case of homogeneous case, but it is easy to generalize it to the heterogeneous structures.

As we can see, once we have the initial conditions and the boundary conditions, it is very easy to implement this equation in a generic semiconductor devices simulator, as it is done in **GNU Archimedes**.

6.5 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

$$\mathbf{E}(x,y) = -\nabla\phi(x,y) \tag{6.10}$$

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

$$E_{xi,j} = -\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x} \tag{6.11}$$

$$E_{y_{i,j}} = -\frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y} \tag{6.12}$$

These simple expressions are used in **GNU Archimedes** and, even if very simple, they are accurated and robust.

Chapter 7

GNU Archimedes Commands Syntax

In order to define a new general device, in **GNU Archimedes**, we need to describe it by means of keyword of a meta-language implemented in our code. This meta-language is very generic, so it can give the possibility of defining semiconductor devices of quite general structure. Furthermore, since the keyword are very simple to understand, it is possible to define devices in short time period and change them with small modifications, in case of optimisation process, for example, or similar.
Thus, in this chapter, we describe the syntax of all the commands actually implemented in **GNU Archimedes**. The number of elements in this list will certainly increase during the development of new versions.
The definition of a new device have to be written and saved in an ASCII file, which will be processed by **GNU Archimedes**. This is done typing the following row 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 **GNU Archimedes**. For this type in a shell window

```
# archimedes --help
```

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

7.1 ACCEPTORDENSITY

In the definition of a new device, even if the holes are considered as fixed in this version of **GNU 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 the user does not specify any constant value or distribution of the acceptors, **GNU 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 the user specify the value of the acceptor distribution only on a part of the device, the restant part will be considered equal to the intrinsic density.
Let us see, now, how to specify the acceptor distribution on a device. In **GNU 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
```

7.2 CIMP

Since the impurity scattering can be very relevant in the GaAs material, we have implemented it up to 0.0.4 release of **GNU Archimedes**. In order to specify the density of impurity, you type, for example, as follows

```
CIMP 1.e23
```

7.3 COMMENTS

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

```
# this is comment
# MATERIAL 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 could be a sintaxicaly correct line of an input file.

7.4 CONTACT

When a new device is defined, the user needs to specify where the edges are insulator, where there are ohmic contacts and where the Schottky contacts are positioned. Even, you can have the need of applying a potential on a isulator 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 a ”reflective mirror” for the particles, i.e. the particles can not 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 ecception that this is not a particle reservoir, so this contact is only a absorbing one.

The choice *pote* 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 the user, it will be considered as of insulator type with zero potential applied, as default.

Here we report some examples.

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 the user 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 the user have to write

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

Finally, if a user wants 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 user have to write

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

7.5 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 the user does not specify any constant value or distribution of the donors, **GNU 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 the user specify the value of the donor distribution only on a part of the

device, the restant part will be considered equal to the intrinsic density.

Let us see, now, how to specify the donor distribution on a device. In **GNU Archimedes** a sub-domain (but also the entire device) on which we want to specify an 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 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 donor density on the rectangular sub-domain. Let us denote it by N_A

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

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
```

7.6 LEID

This is a very powerfull command. This command is invoked when the user wants to obtain Monte Carlo simulations in a very fast way. Let us suppose that the reader 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 the user 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.

7.7 MATERIAL

When you simulate a new device, you can have the freedom to choose to simulate an heterostructure. Even if in this actual version of **GNU Archimedes** the heterostructures are not taken into account, the user have to specify that the semiconductor device simulated is made of Silicon or of GaAs. This is done in order to prepare the next versions of **GNU Archimedes**. So, at the moment, the only two correct uses of the command **MATERIAL** are the following

```
MATERIAL SILICON
MATERIAL GAAS
```

7.8 TRANSPORT

Pay attention to the fact that this command has a different syntax respect to the previous releases!! When the device is defined, the user has to choose what kind of transport **GNU 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 the user 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** the user has 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, the user has 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 present version of **GNU Archimedes**, are

```
# 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
```

7.9 MOSFET

See the file MOSFET.input in the tests directory of Archimedes for more informations.

7.10 FINALTIME

It is important to choose the final time of a simulation. In fact, if the user wants 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 the user. This is also useful in the case in which the user want to study and simulate the trasient behaviour of a semiconductor device. So, for example, if the user wants to set the final time to 5 picoseconds, this will be done by the following line of the input file

```
FINALTIME 5e-12
```

7.11 TAUW

This command is only needed if the user is 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 the reader can see from the definition of the function τ_W) comporting the presence of NaN. This is avoided by specifying a value for tau_W that will be used in the case it is equal to zero. 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 the reader can see from the example, the unit of measure is the second.

7.12 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 usefull in the choice of a correct time step. This is not implemented, at the moment, so the user have to specify it. This is

done in the following fashion. For example, if the user wants to set the time step to 0.01 picoseconds, the following command line in the input file is needed

```
TIMESTEP 0.01e-12
```

7.13 XLENGTH

In this actual version of **GNU Archimedes**, the device is defined as a rectangular domain. So the user have to specify the x-direction and the y-direction length of this rectangular domain. Concerning the x-direction length, this is setted by the following command line. For example, if the x-length is 5 micron, one have to write

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

7.14 YLENGTH

In this actual version of **GNU Archimedes**, the device is defined as a rectangular domain. So the user have to specify the x-direction and the y-direction length of this rectangular domain. Concerning the y-direction length, this is setted by the following command line. For example, if the y-length is 1 micron, one have to write

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

7.15 XSPATIALSTEP

Once the x-direction length of the new device is defined, the user 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 (at least for the specific equations the author have implemented in this actual version of **GNU Archimedes**), but the user 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. To specify, for example, 100 cells in the x-direction, the following line have to be typed in the input file

```
XSPATIALSTEP 100
```

7.16 YSPATIALSTEP

Once the y-direction length of the new device is defined, the user 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 (at least for the specific equations the author have implemented in this actual version of **GNU Archimedes**), but the user 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. To specify, for example, 50 cells in the y-direction, the following line have to be typed in the input file

```
YSPATIALSTEP 50
```

7.17 QUANTUMEFFECTS

As we have sayed in a precedent chapter, **GNU 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 the user wants to take into account the quantum effects in the simulation, he/she have to tell to **GNU 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 you will need to wait for more long run-time to get the solution. So attention have to be putted in this

choice. If the user knowns, *a priori*, 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.

7.18 NOQUANTUMEFFECTS

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

```
NOQUANTUMEFFECTS
```

7.19 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 usefull in the debugging process of a new defined (and not yet well-defined) semiconductor device. **GNU Archimedes** gives a simple way of doing it. When the user inserts the following line in an input file, it will get some interesting information about the macroscopic evolution of the devices.

```
MAXIMINI
```

This will give, for example, in the case of the precedently presented diode the following output, which is very usefull

```
Max. Potential = 1
Min. Potential = 0
Max. x-elec.field = 0
Min. x-elec.field = -3.57578e+06
Max. y-elec.field = -0
Min. y-elec.field = -0
Max. Density = 5.14954e+23
```

```
Min. Density = 1.33122e+21
```

As we can see from the precedent example, the informations of MAXIMINI are about the maximum and minimum of the following macroscopic variables

1. Electrostatic Potential
2. x-Component of the Electric Field
3. y-Component of the Electric Field
4. Electronic Density

Pay attention to the fact that the calculation of this informations can be, in some cases (for example, for highly refined grid), computationaly 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.

7.20 NOMAXIMINI

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

```
NOMAXIMINI
```

This will be usefull in the case the user already know the transient macroscopic behaviour of the device and does not need such informations, in order to not tax the computer run-time.

7.21 SAVEEACHSTEP

When it is required to simulate the transient behaviour of a semicondutor device, it is very usefull 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 behaviour 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 user have not to run and stop at different time the simulation (which is a very annoying task :)). The only thing to do is to put the following line in the input file processed by **GNU Archimedes**

```
SAVEEACHSTEP
```

This will save all the solution, in the choosed 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 informations can be, in some cases (for example, for highly refined grid), computationaly 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.

7.22 LATTICETEMPERATURE

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

```
LATTICETEMPERATURE 77
```

7.23 STATISTICALWEIGHT

The method implemented in **GNU 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 version of **GNU Archimedes**, the statistical weigth the user can specify is that of the cell in which the density is at maximum. In the other cells, the statistical weigth is opportunely calculated. If the user, for example, wants to set the statistical weight equal to 1500, he/she 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.

7.24 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 a enough long period of time. This is done, in **GNU Archimedes**, by specifying on how many final time step the mean average value have to be computed. So, for example, if the user wants to compute the average mean value of the macroscopic variables on the last 500 time steps of the simulation, one have to type in the input file processed by **GNU Archimedes**

```
MEDIA 500
```

7.25 OUTPUTFORMAT

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

```
OUTPUTFORMAT formattype
```

where *formattype* 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

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

$$\ldots$$

$$x_{N_x} \quad y_j \quad v(x_{N_x}, y_j)$$

$$\ldots$$

$$x_i \quad y_{j+1} \quad v(x_i, y_{j+1})$$

$$x_{i+1} \quad y_{j+1} \quad v(x_{i+1}, y_{j+1})$$

$$x_{i+2} \quad y_{j+1} \quad v(x_{i+2}, y_{j+1})$$

$$x_{i+3} \quad y_{j+1} \quad v(x_{i+3}, y_{j+1})$$

$$\ldots$$

$$x_{N_x} \quad y_{N_y} \quad v(x_{N_x}, y_{N_y})$$

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 scientifical computation community, that is why the author have reputed important to implement it in **GNU Archimedes**.

Chapter 8

Example: The MESFET device.

We report, in this chapter, some examples of 2D Silicon MESFET device simulations. This is a benchmark case which is very usefull in assessing the functionality of a semiconductor device simulator, and so also for textbfGNU Archimedes.

8.1 The Monte Carlo MESFET simulation

You can find the following example in the official ditribution of **GNU Archimedes** in the *test/MESFET* directory.

```
# Silicon MESFET test-1
# created on 22 sep.2004, J.M.Sellier
# modified on 09 oct.2004, J.M.Sellier

MATERIAL SILICON
TRANSPORT MC ELECTRONS

FINALTIME 5.5e-12
TIMESTEP 0.0015e-12
```

```
XLENGTH 0.6e-6
YLENGTH 0.2e-6

XSPATIALSTEP 96
YSPATIALSTEP 32

# Definition of the doping concentration
# ======================================
DONORDENSITY    0.       0.         0.6e-6    0.2e-6    1.e23
DONORDENSITY    0.       0.15e-6    0.1e-6    0.2e-6    5.e23
DONORDENSITY    0.5e-6   0.15e-6    0.6e-6    0.2e-6    5.e23
ACCEPTORDENSITY 0.       0.         0.6e-6    0.2e-6    1.e20

# Definition of the various contacts
# ==================================
CONTACT DOWN  0.0    0.6e-6 INSULATOR 0.0
CONTACT LEFT  0.0    0.2e-6 INSULATOR 0.0
CONTACT RIGHT 0.0    0.2e-6 INSULATOR 0.0
CONTACT UP    0.1e-6 0.2e-6 INSULATOR 0.0
CONTACT UP    0.4e-6 0.5e-6 INSULATOR 0.0
CONTACT UP    0.0    0.1e-6 OHMIC     0.0 5.e23
CONTACT UP    0.2e-6 0.4e-6 SCHOTTKY -0.8
CONTACT UP    0.5e-6 0.6e-6 OHMIC     1.0 5.e23

NOQUANTUMEFFECTS
MAXIMINI
```

```
# SAVEEACHSTEP

LATTICETEMPERATURE 300.

STATISTICALWEIGHT 1500
MEDIA 500

OUTPUTFORMAT GNUPLOT

# end of MESFET test-1
```

We report in the following pictures (8.1)-(8.9)the results obtained by **GNU Archimedes** release 0.0.4.

8.2 The Fast Monte Carlo MESFET simulation

In this section we want to show how to obtain faster simulation than the previous section. First of all we simulate the MESFET device by means of the simplified MEP model. This is done by the following script

```
# Silicon MESFET test-1
# created on 27 feb.2005, J.M.Sellier
# modified on 05 mar.2005, J.M.Sellier
# Simplified MEP model

MATERIAL SILICON
TRANSPORT MEP ELECTRONS
```

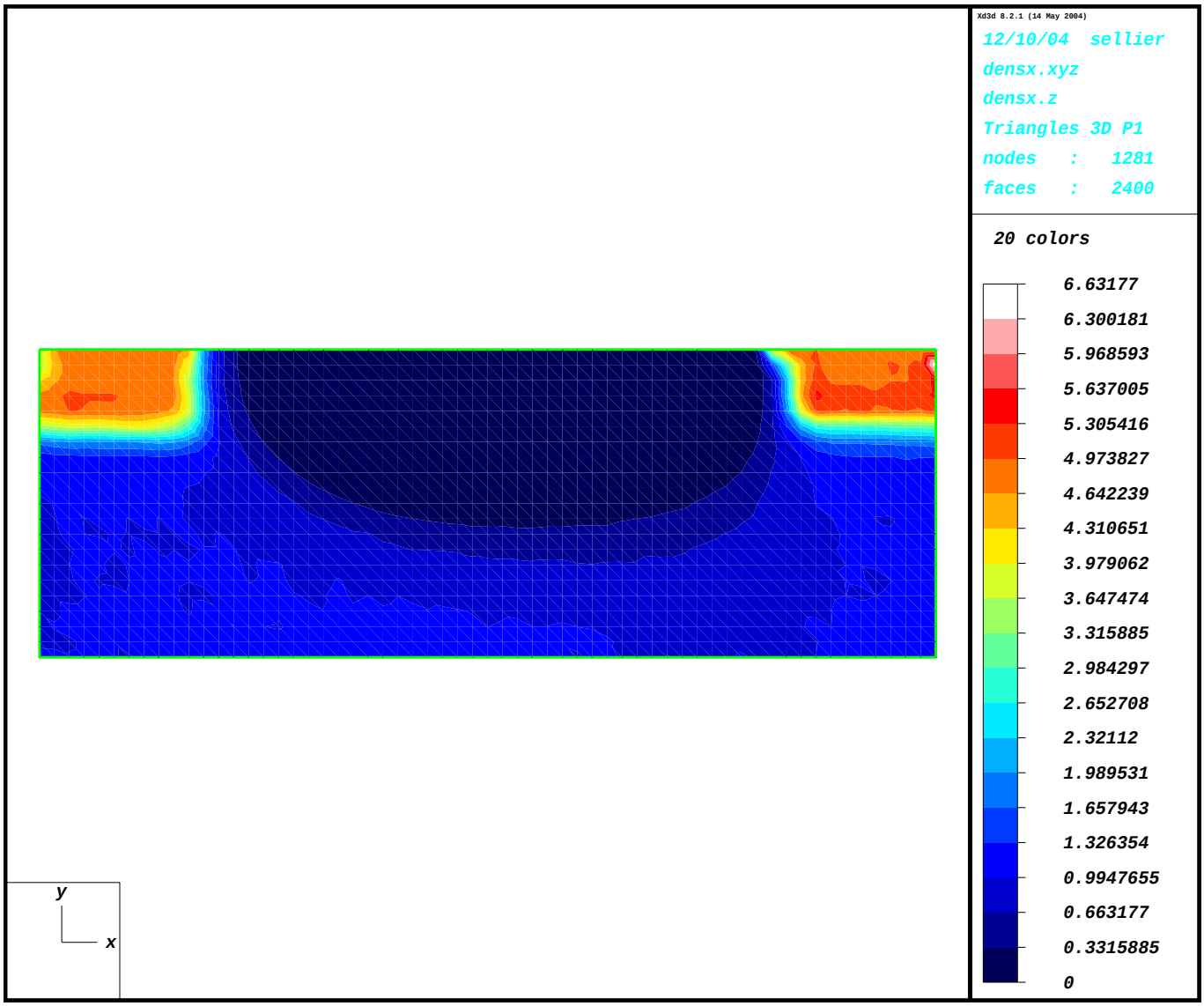


Figure 8.1: MESFET Density Profile computed by **GNU Archimedes**.

```
FINALTIME 5.0e-12
TIMESTEP 0.001e-12

XLENGTH 0.6e-6
YLENGTH 0.2e-6
```

```
XSPATIALSTEP 90
YSPATIALSTEP 20

# Energy relaxation time
# ======================
TAUW 0.4e-12

# Definition of the doping concentration
# ======================================
DONORDENSITY    0.       0.         0.6e-6    0.2e-6    1.e23
DONORDENSITY    0.       0.15e-6    0.1e-6    0.2e-6    3.e23
DONORDENSITY    0.5e-6   0.15e-6    0.6e-6    0.2e-6    3.e23
ACCEPTORDENSITY 0.       0.         0.6e-6    0.2e-6    1.e20
# ACCEPTORDENSITY    0.       0.         0.6e-6    0.2e-6    1.e23
# ACCEPTORDENSITY    0.       0.15e-6    0.1e-6    0.2e-6    3.e23
# ACCEPTORDENSITY    0.5e-6   0.15e-6    0.6e-6    0.2e-6    3.e23

# Definition of the various contacts
# ==================================
CONTACT DOWN  0.0    0.6e-6 INSULATOR 0.0
CONTACT LEFT  0.0    0.2e-6 INSULATOR 0.0
CONTACT RIGHT 0.0    0.2e-6 INSULATOR 0.0
CONTACT UP    0.1e-6 0.2e-6 INSULATOR 0.0
CONTACT UP    0.4e-6 0.5e-6 INSULATOR 0.0
CONTACT UP    0.0    0.1e-6 OHMIC     0.0 3.e23 1.e20
CONTACT UP    0.2e-6 0.4e-6 SCHOTTKY -0.8
CONTACT UP    0.5e-6 0.6e-6 OHMIC     1.0 3.e23 1.e20
```

```
NOQUANTUMEFFECTS
MAXIMINI
# SAVEEACHSTEP

LATTICETEMPERATURE 300.

STATISTICALWEIGHT 100
MEDIA 500

OUTPUTFORMAT GNUPLOT

# end of MESFET test-1
```

Then we copy the density, energy and potential files with the following names

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

and run the following script

```
# Silicon MESFET test-1
# created on 27 feb.2005, J.M.Sellier
# modified on 05 mar.2005, J.M.Sellier
# Fast Monte Carlo method

MATERIAL SILICON
TRANSPORT MC ELECTRONS
```

```
FINALTIME 1.e-12
TIMESTEP 0.001e-12

XLENGTH 0.6e-6
YLENGTH 0.2e-6

XSPATIALSTEP 90
YSPATIALSTEP 20

# Energy relaxation time
# ======================
TAUW 0.4e-12

# Definition of the doping concentration
# ======================================
DONORDENSITY    0.       0.         0.6e-6    0.2e-6    1.e23
DONORDENSITY    0.       0.15e-6    0.1e-6    0.2e-6    3.e23
DONORDENSITY    0.5e-6   0.15e-6    0.6e-6    0.2e-6    3.e23
ACCEPTORDENSITY 0.       0.         0.6e-6    0.2e-6    1.e20
# ACCEPTORDENSITY    0.       0.         0.6e-6    0.2e-6    1.e23
# ACCEPTORDENSITY    0.       0.15e-6    0.1e-6    0.2e-6    3.e23
# ACCEPTORDENSITY    0.5e-6   0.15e-6    0.6e-6    0.2e-6    3.e23

# Definition of the various contacts
# ==================================
CONTACT DOWN  0.0    0.6e-6 INSULATOR 0.0
```

```
CONTACT LEFT  0.0    0.2e-6 INSULATOR 0.0
CONTACT RIGHT 0.0    0.2e-6 INSULATOR 0.0
CONTACT UP    0.1e-6 0.2e-6 INSULATOR 0.0
CONTACT UP    0.4e-6 0.5e-6 INSULATOR 0.0
CONTACT UP    0.0    0.1e-6 OHMIC     0.0 3.e23 1.e20
CONTACT UP    0.2e-6 0.4e-6 SCHOTTKY -0.8
CONTACT UP    0.5e-6 0.6e-6 OHMIC     1.0 3.e23 1.e20

NOQUANTUMEFFECTS
MAXIMINI
# SAVEEACHSTEP

# Load Electron Initial Data
LEID

LATTICETEMPERATURE 300.

STATISTICALWEIGHT 100
MEDIA 500

OUTPUTFORMAT GNUPLOT

# end of MESFET test-1
```

As the reader can note from the following script, we only need 1 picosecond for the Monte Carlo method. This is done because we load the initial conditions from the previous MEP simulation. This is a very fast way to obtain fast Monte Carlo simulations. We report, in the following some interesting

results.

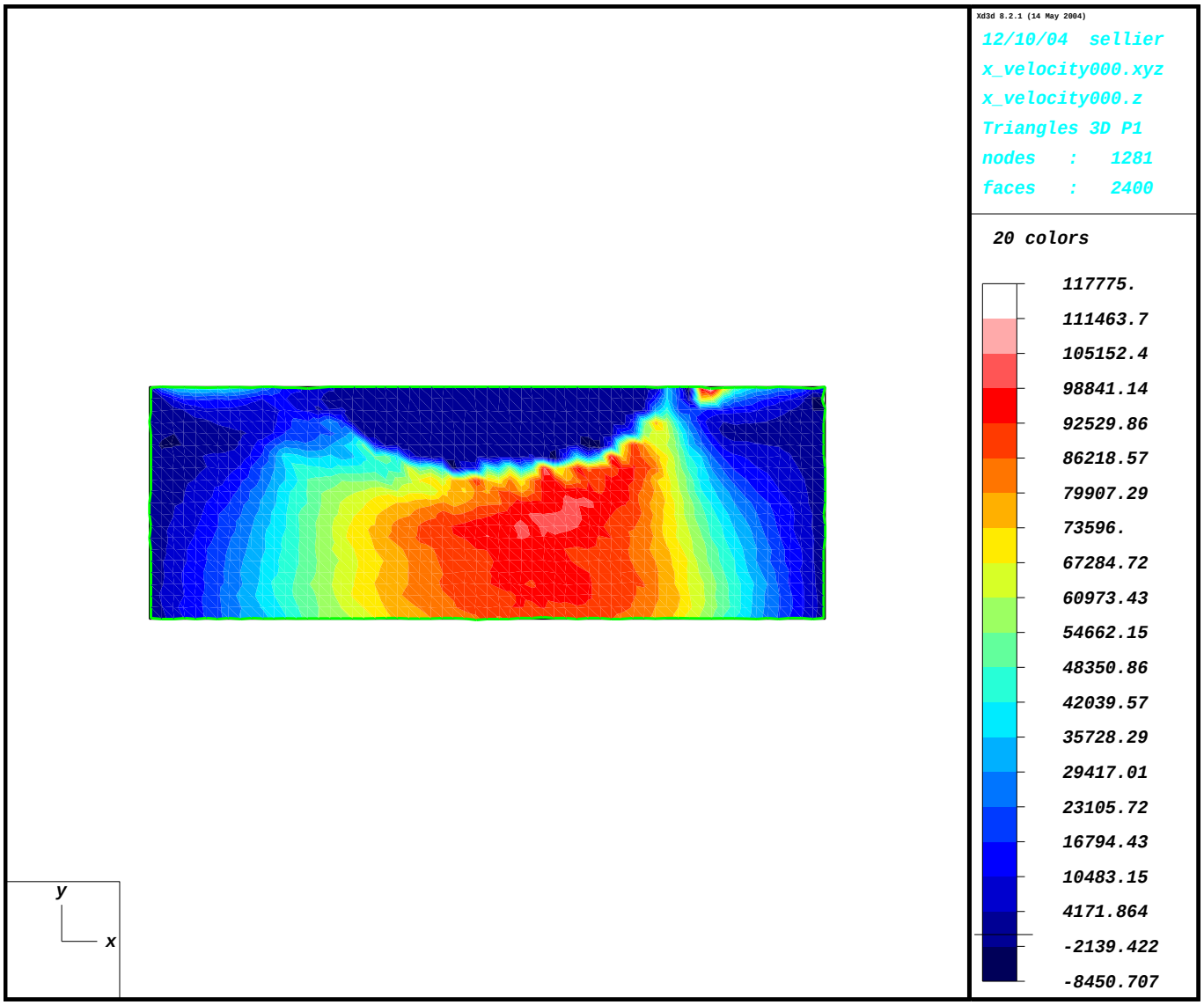
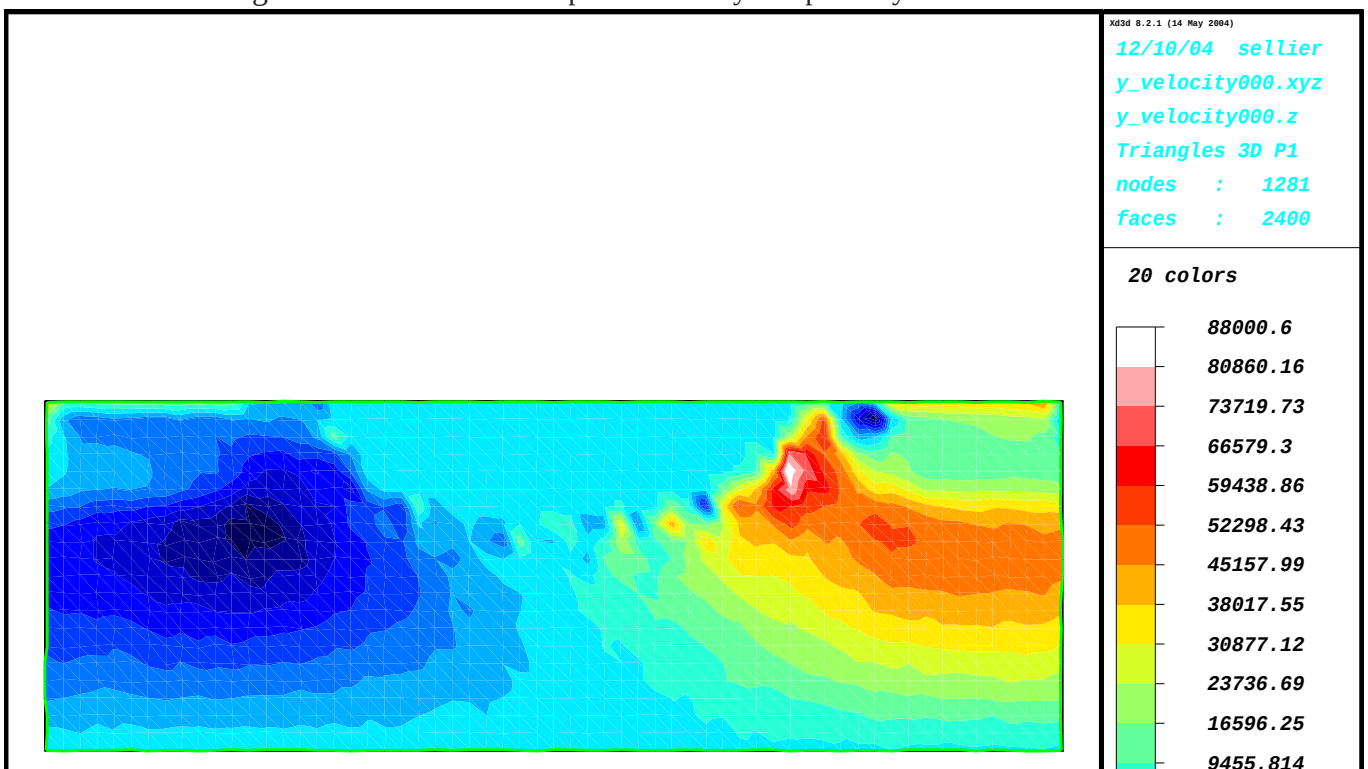


Figure 8.2: MESFET x-component velocity computed by **GNU Archimedes**.



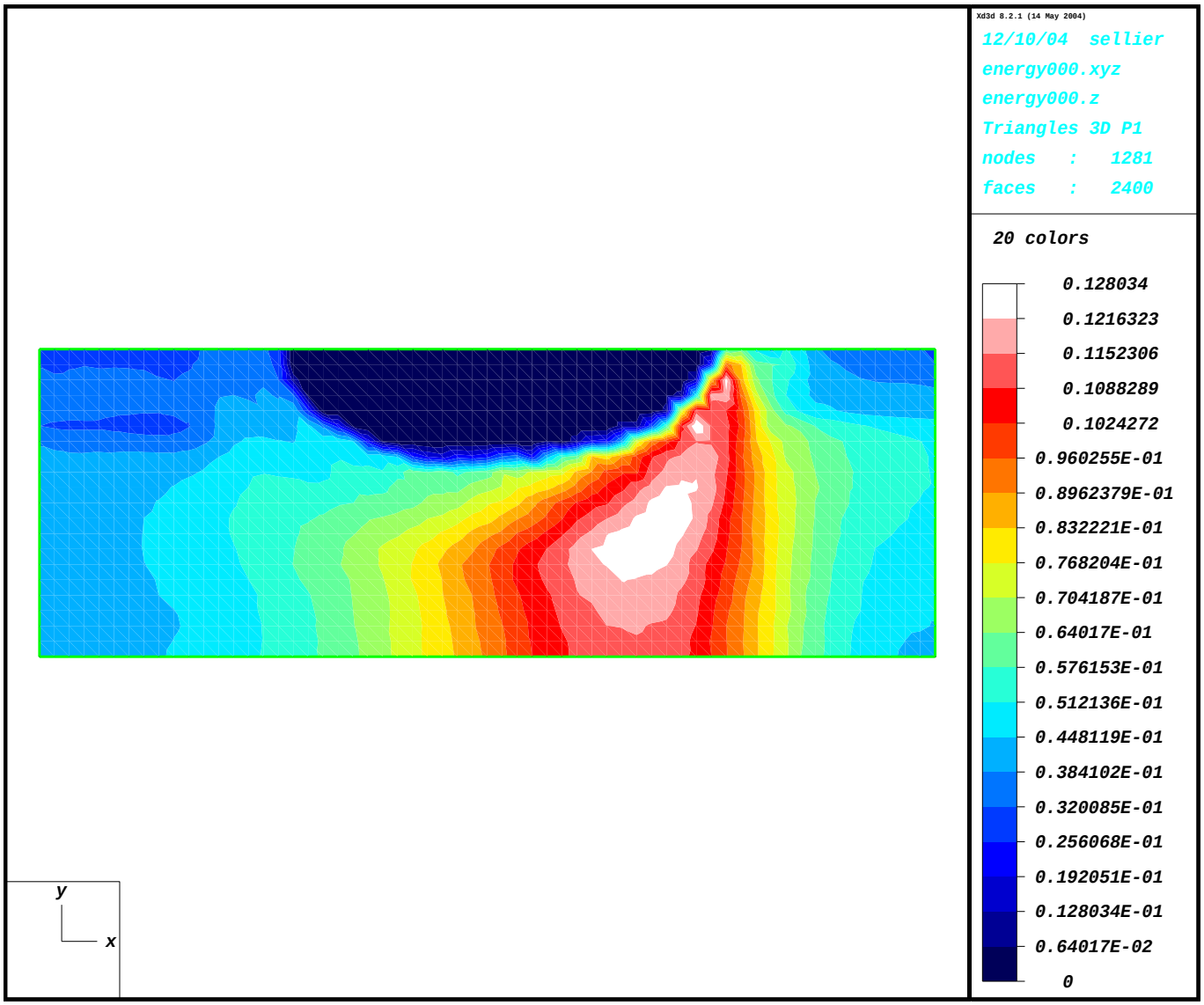
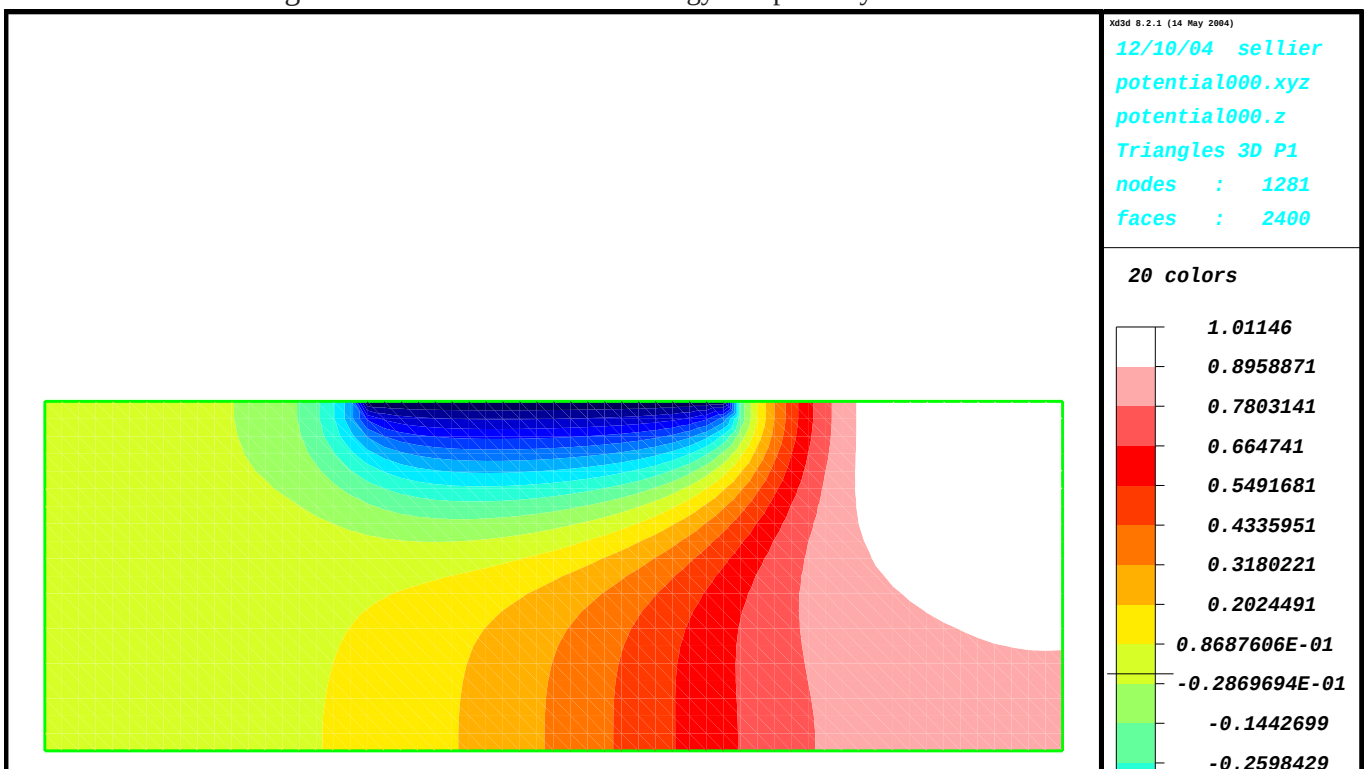


Figure 8.4: MESFET Electron Energy computed by **GNU Archimedes**.



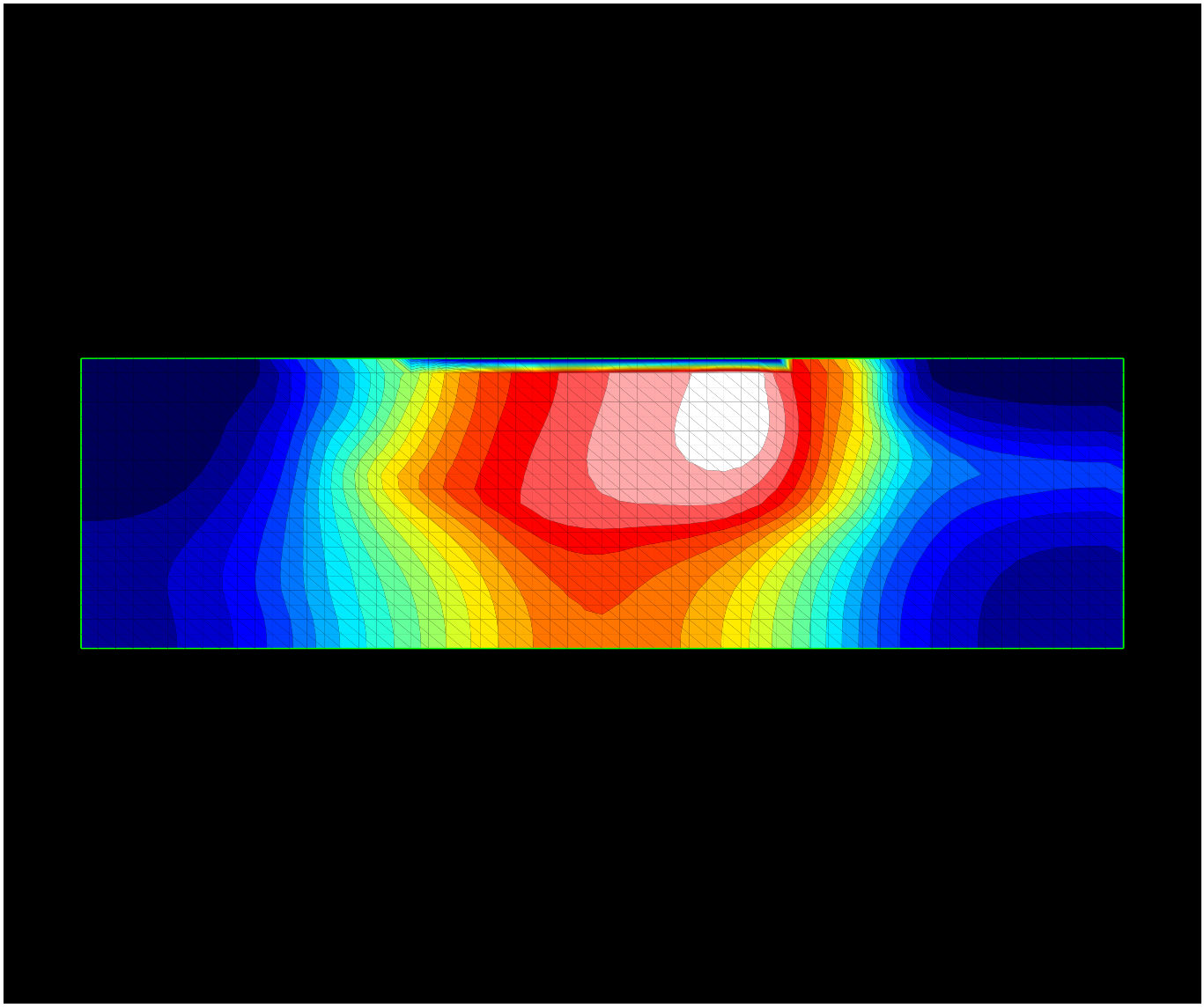


Figure 8.6: MESFET MEP Energy Profile computed by **GNU Archimedes**.

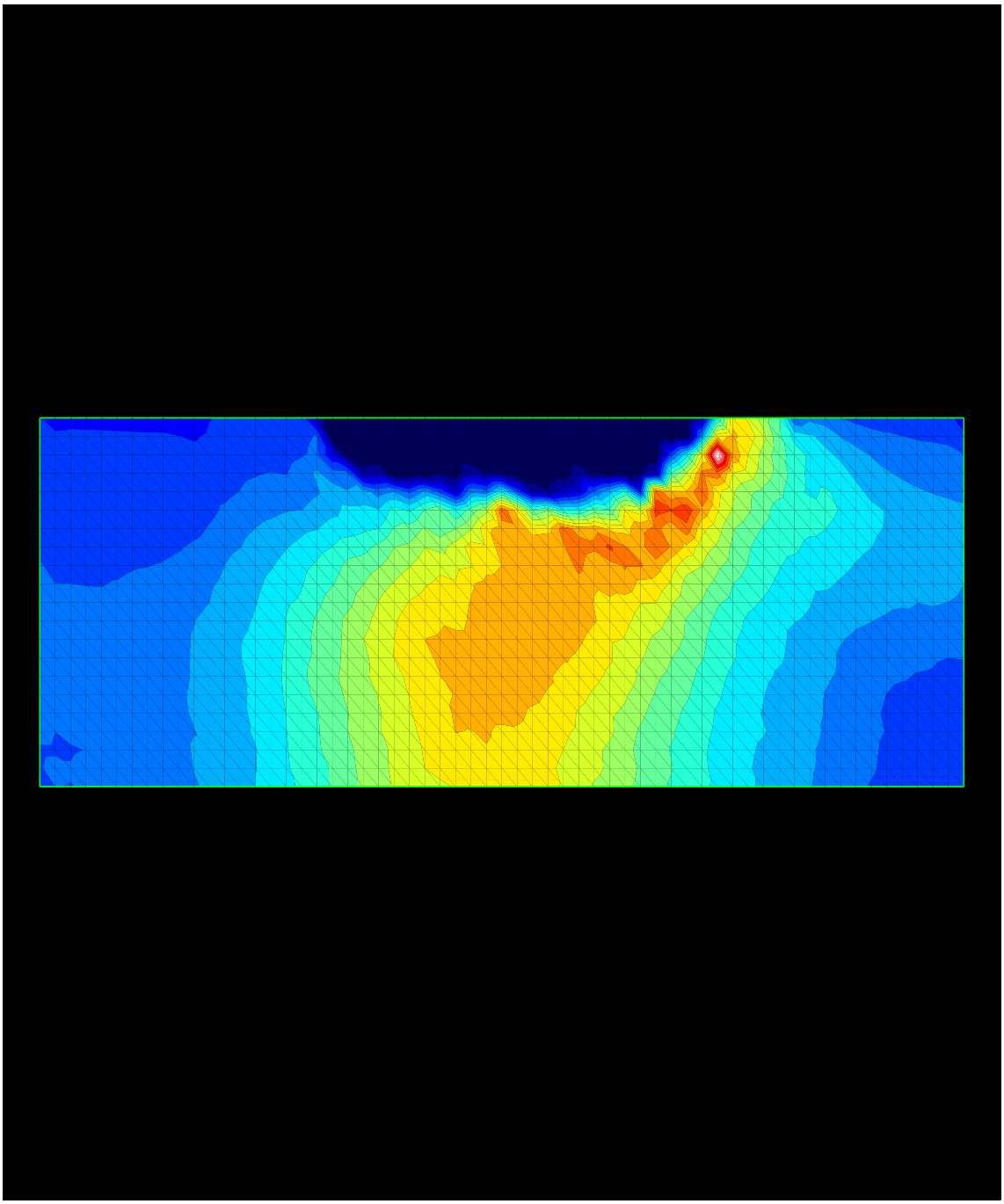


Figure 8.7: MESFET Fast Monte Carlo Energy Profile computed by **GNU Archimedes**.

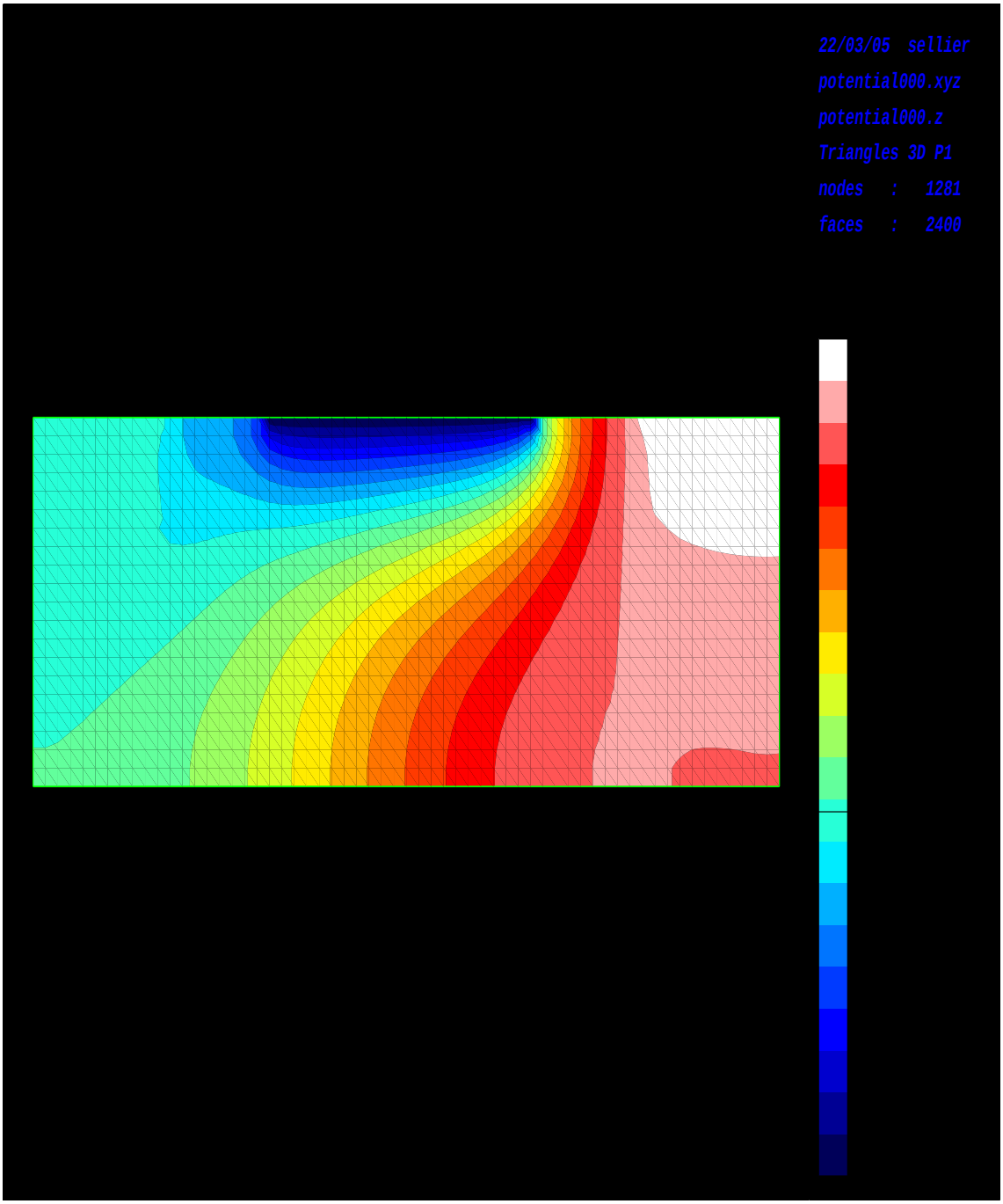
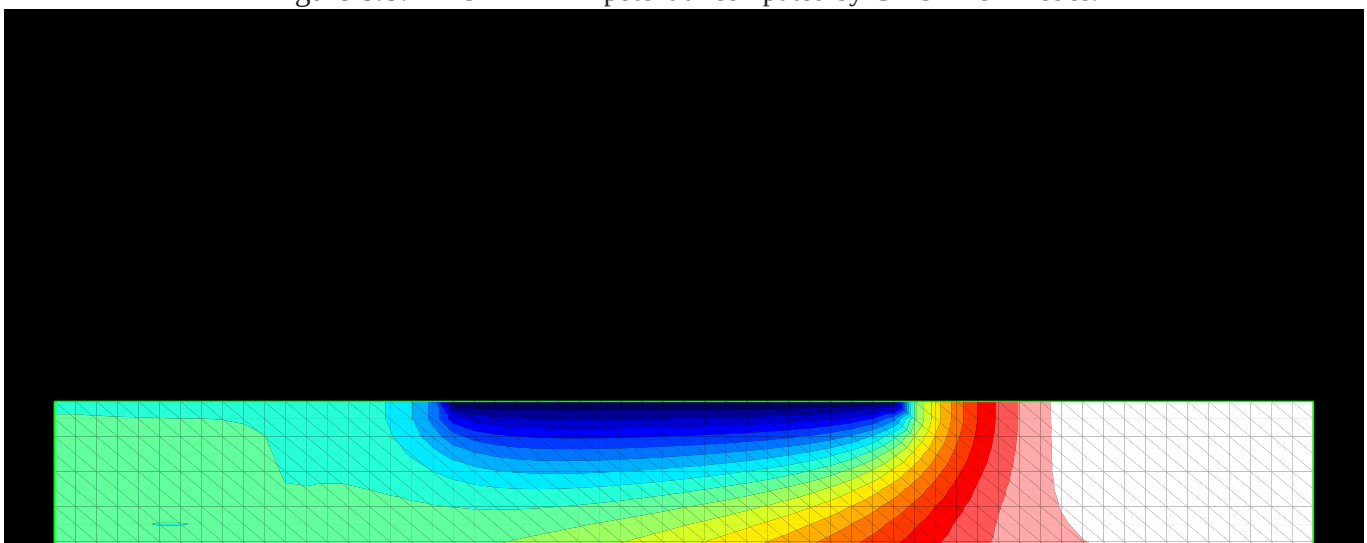


Figure 8.8: MESFET MEP potential computed by **GNU Archimedes**.



Chapter 9

Acknowledgments

In this very short chapter I want to thank the people who have helped and encouraged me in creating **GNU Archimedes**

1. **Richard M. Stallman**. Thanks a lot for all the advices, suggestions and encouraging in creating this code.

2. **Karl Berry**. Thanks a lot for all the advice, suggestions and patience in aswering to all my boring and, sometimes, stupid and trivial questions :)

3. **Vittorio Romano**. Thanks a lot for all your entusiasm and support during the creation stage and development period of **GNU Archimedes**

4. **A. Marcello Anile**. Thanks a lot for everything you did teach me. Without your lessons this code will never exist.