Modelling of Recessed Gate MESFET Structures.

T.M. Barton, C.M. Snowden, and J.R. Richardson

Microwave Solid State Group,
Department of Electrical and Electronic Engineering,
The University of Leeds, Leeds, LS2 9JT, England.

Summary

The development of a new semiconductor device is a process which often involves an iterative cycle of design, fabrication, and re-design, until the desired specification is reached. Physical modelling of semiconductor devices is being used to a greater and greater extent to reduce the time spent in the design process, by allowing the engineer to rapidly determine the effect of the changes he may make to a device upon the device's final performance. The diverse nature of the geometries used in modern devices, even those in the same class, makes it almost essential that a modelling scheme be used which is capable of adapting, with the minimum of change to the software, to the geometry of the device to be modelled.

The current work is concerned with the development of such a model. The software uses the techniques of automatic finite-difference grid generation, and adaptive refinement, to enable it to be used, without modification, on almost any device geometry. The software has been used to simulate a variety of GaAs MESFET's, with and without gate recesses and surface charge effects. The results of the simulation of both a planar and a recessed gate 1.0μm gate length device is presented. It will be seen that near pinch-off a large proportion of the current between the source and the drain of the device is flowing deep within the substrate, a phenomenon which greatly increases the calculated pinch-off voltage above that predicted by simple one-dimensional calculations, and results in an increase in calculated drain conductance, compared with experimental devices.

1. Introduction

In order to provide the maximum flexibility, a program which is to be used in the engineering environment to simulate a variety of semiconductor devices must have several features. First, it must allow the user to quickly and easily change the parameters of the device to be modelled. These include the shape of the device, the doping profiles, and the number and position of the different types of contacts. Second, it must be possible to rapidly post-process the results of a simulation. This will often involve interactive, graphical presentation of data, which may be produced at any time after the simulation is complete. Third, the turnaround time for a simulation must be as sort as possible. Ideally, the simulation should be done interactively, however, this is rarely possible given the computer technology currently available and the complexity of the physical model. Simulation programs should be made as efficient as possible in order to minimise CPU time requirements, and, perhaps more importantly, to make the best possible use of scarce or expensive computer resources.

Many of the simulation programs in use today do not have these features. Device simulations are often written for the purely research environment, as opposed to the engineering environment in which device design is carried out. Many simulation programs define the device geometry and parameters as part of the computer code, and to change them requires a knowledge of the language and techniques used by the writer. Many programs allow some degree of modification to be performed to the device as data to the running program, for example the length of a FET gate may be altered, or the depth of the active channel changed. There are still few programs which allow the entire device to be arbitrarily defined at run-time, and thus are able to be used for many different devices.

The current work describes the implementation of a program which allows the user to define, in the data supplied at run-time, the device geometry and parameters, the type of simulation to be performed, the contact bias, and so fourth. Data is read, either from a computer terminal or a data file, as a set of instructions to the program. The program produces output in the form of binary files which may be read by a second program, which is used to interactively analyse the results of the simulation. Graphical data may be drawn on graphics terminals, or plotted, in response to instructions given at the user's terminal (for example figure 8). The program is written in the PASCAL language, and uses the advanced data types and structures provided by this language in order to efficiently store the finite difference grid used in the simulation. While this package is at present used as a

research tool, it's advantages for engineering design are clear.

2. Semiconductor Model

The model for the semiconductor used in the current simulation is based upon the well known classical equations, which may be derived from an approximate solution of the first two moments of the Boltzmann transport equation [1]. This model requires the self-consistent solution of three coupled partial differential equations.

The first of these is Poisson's equation,

$$\nabla^2 \phi = \frac{-q}{\epsilon}\left(N_D - n\right) \tag{1}$$

where ϕ is the electrostatic potential, q the electronic charge, ϵ the permittivity of the semiconductor material, N_D the donor density, and n the density of electrons in the conduction band.

Second is the current density equation, given by

$$J_n = -qn\mu_n \nabla\phi + qD_n \nabla n \tag{2}$$

where μ_n and D_n are respectively the electron mobility and diffusion coefficient, and J_n the electron current density.

The third equation is the so-called current continuity equation, which is

$$\frac{\partial n}{\partial t} = \nabla . J_n + G \tag{3}$$

where G is the time rate of generation of electrons in the conduction band. At present, the model is used for unipolar devices only, and thus holes are neglected.

These equations are solved over the simulation domain using finite-difference techniques, which are in general faster and simpler to code than comparable finite-element methods. The Poisson equation is discretised using the normal Taylor series expansion for the potential about each node to find the second derivative [2]. The current density equation needs to be discretised with care. A Taylor series expansion for the electron density, in the fashion of that used in the discretisation of the potential function for the Poisson equation, causes numerical instability if the potential between adjacent nodes in the finite-difference mesh exceeds 2kT / q volts. In order for this scheme to work, extremely small mesh spacings must be used. In order to overcome this problem, a discretisation which allows an exponential variation of electron density between nodes is used. The discrete current equation used here is similar to that proposed by Scharfetter and Gummel [3], which allows large space steps to be used, with the corresponding saving in

storage and calculation time.

The current continuity equation is solved using a semi-implicit, "half-point" discretisation scheme, similar to that used by Reiser [4]. A finite-difference discretisation is used to find the time derivative in equation (3). This results in a discrete continuity equation of the form

$$\frac{\partial n}{\partial t} = \frac{n^{k+1} - n^{k}}{\Delta t} = -\frac{1}{q}\left[(\omega)\nabla.J\left(n^{k+1},(\mu E)^{k}\right) + (1-\omega)\nabla.J\left(n^{k},(\mu E)^{k}\right)\right] \quad (4)$$

where the superscripts k and k+1 refer to the values of the superscripted variables at the current and the next timestep respectively. For ω=0 this discretisation is fully explicit, and the solution is trivial. However, the explicit solution shown a strong instability, and great care must be exercised in the choice of space and time steps. For ω>0, the solution becomes partially implicit, and subsequently much more stable. The assumption that the electron velocity (μE) remains constant between timesteps may lead to some instability for long timesteps, it is found, however, that for doping densities below 10^{18} cm^{-3} in GaAs, a timestep of less than 10fS almost invariably guarantees stability. The current work takes ω to be equal to ½.

The discretised forms of the semiconductor equations (1), (2), and (3) are solved using a successive over-relaxation (SOR) iterative scheme. Poisson's equation is first solved based upon the existing electron density. The current continuity equation is then solved in conjunction with the current density equation, to obtain the electron density at the next timestep, which again used in the solution of Poisson's equation. Repetition of this loop thus gives the time-dependent potential and electron density distributions throughout the device. Contact potentials are found by integration of the current equation around the contacts, and thus the variation of the contact currents with time may be found.

3. The Finite-Difference Grid

In order to ensure the maximum flexibility for the simulation software, details of the shape and characteristics of the semiconductor must be able to be specified at run-time, as part of the data supplied to the program by the user. As a result, the software must have no built-in specification for the shape of the device, the doping distribution, the contact types and positions, and so on. To be able to employ the optimum finite-difference grid for each device, the software must be able to generate the grid in response to the geometry and characteristics of the input device. This is done in two stages. When the simulation is commenced, a sparse,

regularised grid is generated, upon which a solution is obtained. The grid is then refined in response to this initial solution by the addition of rows and columns of mesh nodes into the existing grid.

The geometry of the device is given to the program as a series of edges describing the outline of the device. Each of these has a type associated with it, for instance it may be a GaAs surface, an ohmic or Schottky contact, or a general Neumann boundary. These device edges are then checked to ensure that they form a closed polygon, and then joined by placing nodes at the ends, thus they form the first grid lines (Figure 1). These nodes are then joined to edges on the other side of the device, and a node is inserted at each intersection, thus further grid lines are generated. This process results in the first fully connected grid (Figure 2).

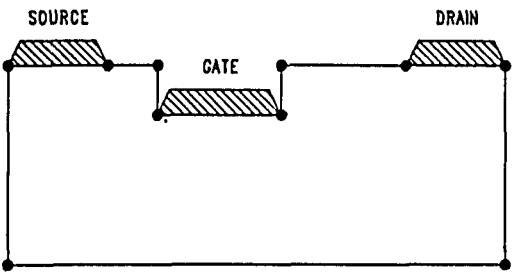


Figure 1: Grid lines and Nodes Generated by initial geometry data.

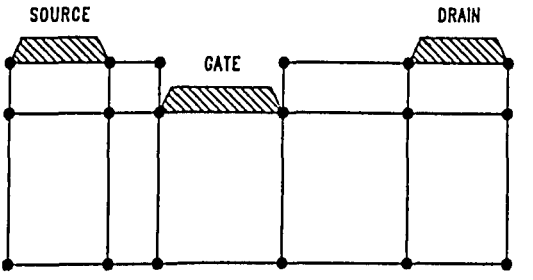


Figure 2: Pre-Initial Grid formed by connecting initial nodes.

The next step is to insert grid lines until some maximum spacing criterion has been satisfied. Sufficient new grid lines are inserted at regular intervals between existing lines to ensure that the spacing between lines is less than a

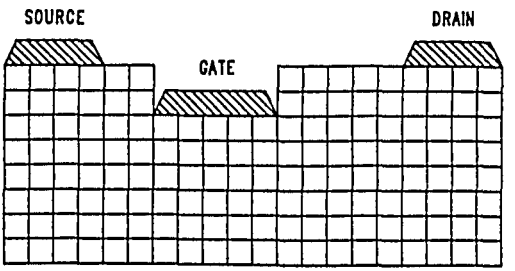


Figure 3: Initial Finite-Difference Grid.

certain distance, generally 0.1 to 0.05 μm for a typical GaAs MESFET. The resulting grid (Figure 3) is called the initial grid. The semiconductor equations are then solved on this grid to give an initial steady-state solution. This is done by solving the time-dependent semiconductor equations until the contact currents remain constant between timesteps.

This initial solution is used to further refine the grid. New nodes are inserted in regions where the solution accuracy is poor. These are usually those regions at which the potential or the electron density varies by more than a certain amount between nodes. An example of a refined grid may be seen in figure 4. The values for the solution variables at the new nodes are found initially from a quadratic interpolation of the values at the pre-existing nodes on either side of the new node. This updated grid is then used to re-compute the steady-state solution over the entire mesh, retaining the original contact potentials, after which several more grid updates may be performed until some final solution accuracy is reached. After this process is complete, an A.C. or transient simulation may be performed by

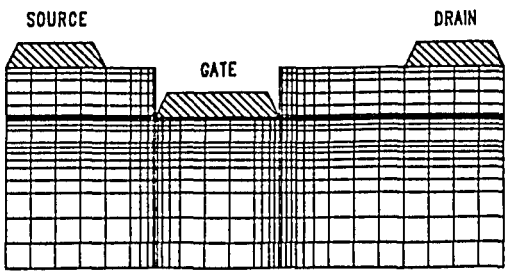


Figure 4: Example Refined Grid.

varying the contact potentials incrementally over a number of timesteps, as would happen in a real device stimulated by an external circuit.

4. Programming Considerations

Computer simulations of semiconductor devices, especially those which may be used to generate A.C. or transient results, typically use large amounts of computer resources. The most important of these are time and memory. For example, on a VAX 11/780 minicomputer it is not unusual for a simulation to require an hour of CPU time (which can translate to several hours elapsed time), and two or three megabytes of memory, to calculate the steady-state solution for one bias point. This can incur a large cost to the user.

In order to minimise this cost, careful consideration must be made to the language, algorithms, and data structures used in the program. Often it is found that there is a trade-off to be made between time and memory. Calculating and storing internal variables once for use many times throughout the execution of the program may take less time than re-calculating them every time they are used, but requires more memory.

The efficiency of a program is also dependent upon the initial definition of the problem. In general, if all of the geometry and characteristics of a device are defined when the program is written, a very efficient program can result. If the geometry is undefined when the program is written, the program will, in general, be less efficient.

Programs which have the geometry of the simulated device built in to their structure often use fixed, rectangular two-dimensional arrays to store the function values at each of the finite-difference nodes. In this structure, the position of a point in the device is related in some invariant way to it's position within the array, that is the point [i,j] in the array may be at coordinate $(i\times\Delta x, j\times\Delta y)$ in real space. In addition, the spatial relationship between nodes is reflected in the structure of the array, the points [i,j] and [i,j+1] in the array are also adjacent in real space. The ability of the current program to define the device at run-time makes it impossible to determine the shape of the grid, the number of nodes, and the relationship between them, when the program is compiled. As a result it is not possible to use this structure to store the nodal data.

In the current simulation three additional complications present themselves. First, the mapping between grid row and column and real space coordinate may be non-linear (as for a non-uniform mesh). Second, it is necessary to be able to insert rows and columns into the mesh. Third, there may be

more than one row (or column) at the same x (or y) position, as are the first few horizontal lines in the FET of figure 2. Special data structures have been developed to store the nodal data. The current software has been implemented in both FORTRAN and PASCAL, with different data structures used in each.

The FORTRAN implementation uses single dimensional arrays to store the function values and the position of each node. These are of sufficient length to be able to contain the maximum possible number of points. Different arrays are used for each variable (potential, electron density etc), the same index in each array referring to the same mesh node. In

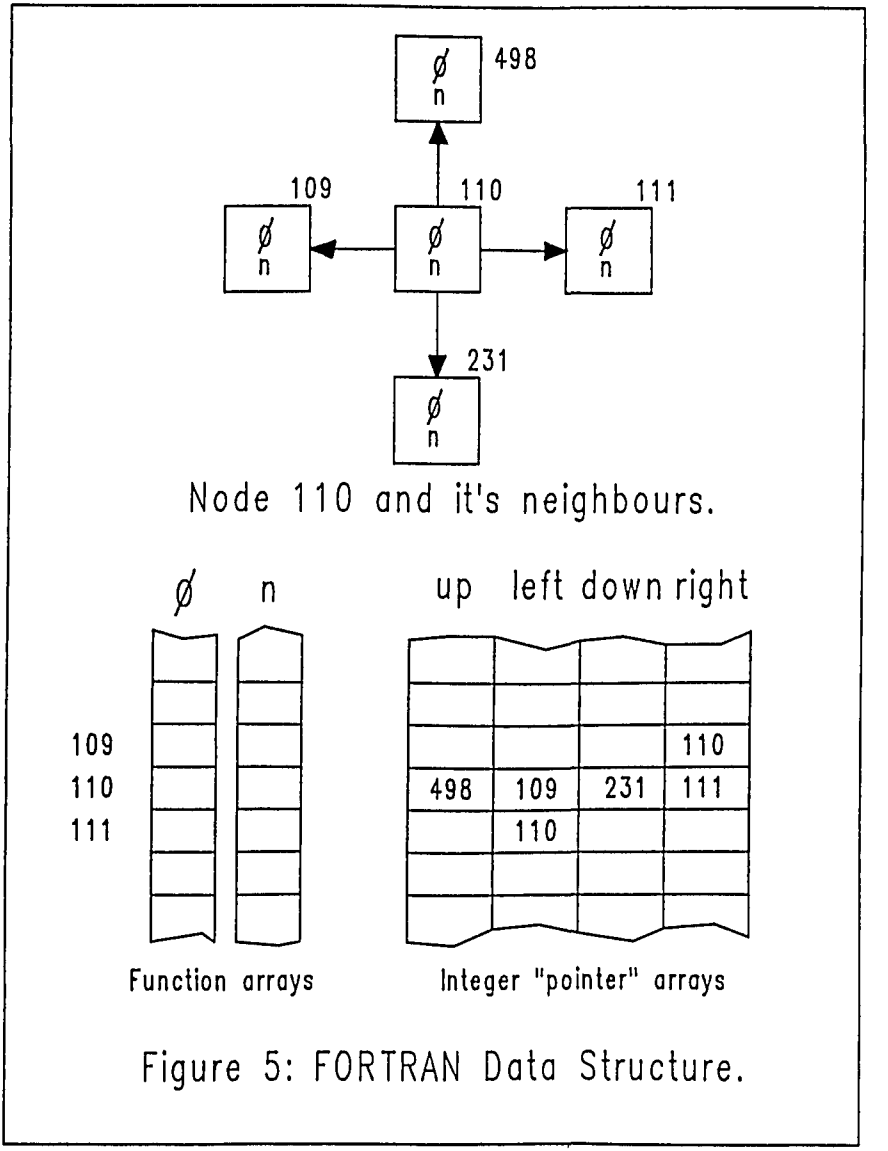


Figure 5: FORTRAN Data Structure.

addition, four integer arrays are used to store the indices of the node's neighbours, i.e. the nodes immediately above, below, left and right of the current node. These arrays effectively "point" to the nodes on either side of the current node (Figure 5). This implementation was found to be relatively inefficient. The data pertaining to a node and it's neighbours is spread in a random fashion throughout the computer memory, on a virtual memory system this can result in a great deal of time being spent performing relatively slow paging operations during the iteration process. In addition, many costly address calculations must be performed at run-time while accessing the data associate with a node and it's neighbours. In contrast, much of this calculation can be performed at compile-time by an efficient compiler optimiser when a fixed, two-dimensional, array structure is used, as the position in memory of each array, and point within the array, is known.

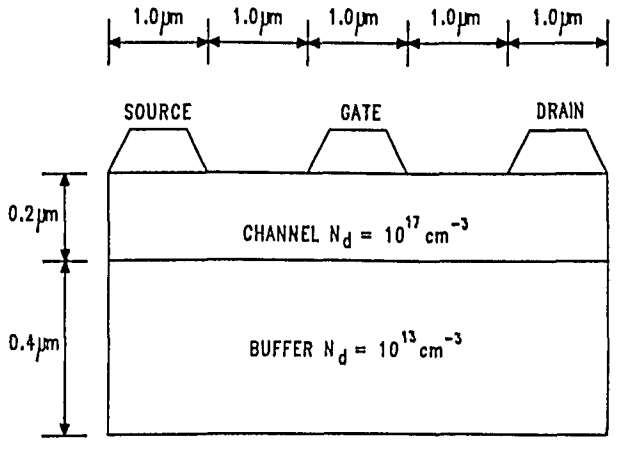
The more recent PASCAL implementation of the software uses a more elegant and efficient method of storing the nodal information. The PASCAL RECORD structure is used to contain all of the information pertaining to one node, thus keeping it in the same vicinity in memory. In addition, the PASCAL type POINTER is used within each of these records to hold the actual address in memory of the records containing the neighbouring nodes, which allows rapid access to adjacent nodes, without the necessity of lengthy address calculations. The dynamic nature of the grid generation process itself gives PASCAL a further advantage. The PASCAL NEW() and DISPOSE() built-in functions may be used to generate new nodal records as they are required during the simulation process. Initially, the program uses very little memory, as the grid grows during the refinement process, so does the memory usage. This can be an important advantage where computer resources are sparse, as the size of the program is related to the size of the problem, unlike the FORTRAN implementations, where the size of the internal arrays, and hence the program as a whole, must be determined at compile-time.

It was found that the PASCAL implementation used less time and memory when compared to the FORTRAN program. The speed of execution was more than doubled, and the program size was less than three-quarters, for the same problem. Comparison of the PASCAL with the best optimised, fixed two-dimensional, uniform rectangular grid FORTRAN program showed that the PASCAL was still only about one third the speed, however, it is felt that the added flexibility of the variable grid program far outweighs this speed difference. The current software is comprised of about 10000 lines of code, divided into two programs. The first of these performs the simulation, producing a binary file of the results. The second reads this file, and may be used to produce graphical output, for example graphs of the terminal currents, plots of

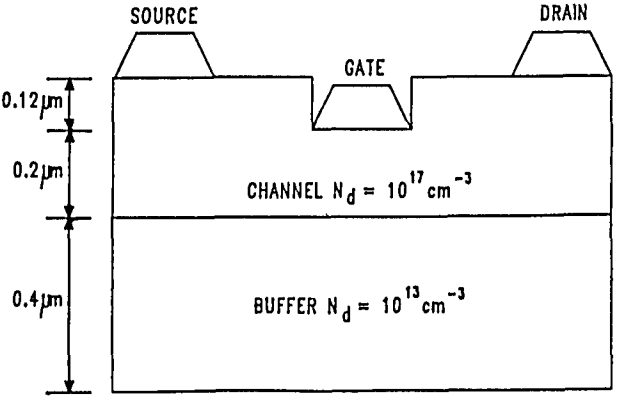
the internal potentials and electron densities etc. (for example, figure 8).

5. Simulation Results

The computer model described in the previous sections has been used to simulate both a planar and recessed gate MESFET. The geometry of these devices is shown in figure 6. Both



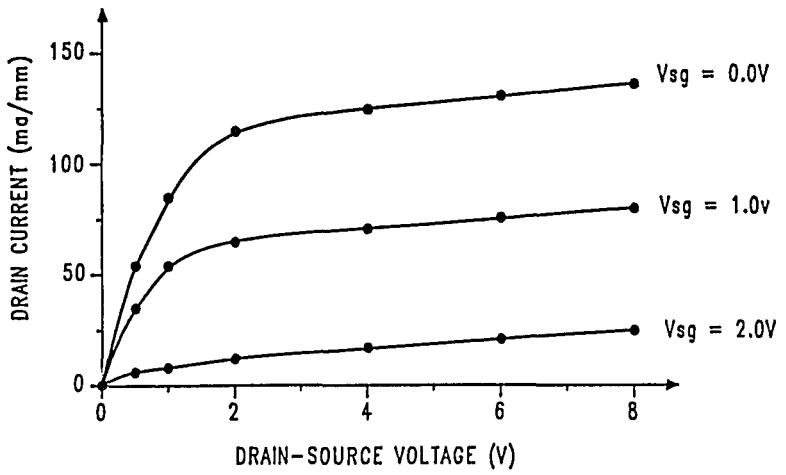
(a) Planar MESFET



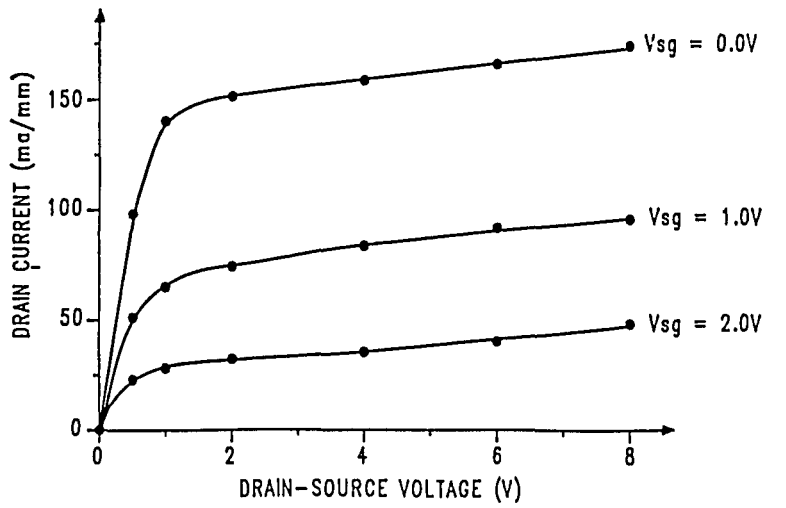
(b) Recessed Gate MESFET

Figure 6: Planar and Recessed Gate Devices.

devices have a gate length of 1.0μm and an active channel depth of 0.2μm below the gate. The recessed gate device has a gate recess depth of 0.12μm. The doping in the active channel is 10^{17} cm^{-3}, and that in the buffer region is 10^{15} cm^{-3}. The spacings between the three contacts is 1.0μm, and the first 1.0μm of the drain and source contacts are modelled.



(a) Planar MESFET



(b) Recessed Gate MESFET

Figure 7: Calculated Transfer Characteristics of the Devices of Figure 6.

The calculated transfer characteristics of these devices are shown in figure 7. These were produced by calculating the steady-state terminal currents for a variety of bias points. The I_{DSS} of the planar device is about 125 ma/mm of device width, and that of the recessed gate device about 150 ma/mm. The pinch-off voltage of the planar device is around 2.5 volts, in comparison with the 2.0 volts calculated from a simple one-dimensional model for the Schottky gate. This difference can in part be ascribed to the perturbation of the electrostatic solution around the gate from that calculated by the one-dimensional model, due to the inherent two-dimensional nature of the contact. The pinch-off voltage of the recessed gate device is somewhat higher, being around 3.0 volts. The electrostatic solution for the potential and charge distribution around the recessed gate is further perturbed from that calculated by the simple one-dimensional model by the edges of the gate recess, as some of the depletion region charge moves into that part of the channel above that gate. This has the effect of both raising the potential at which the channel pinches off, and causing the channel current to vary in a more non-linear fashion with applied gate potential. The output conductance is also marginally higher for the recessed gate device.

Figure 5: Current density in the Recessed Gate FET near Pinch-off.

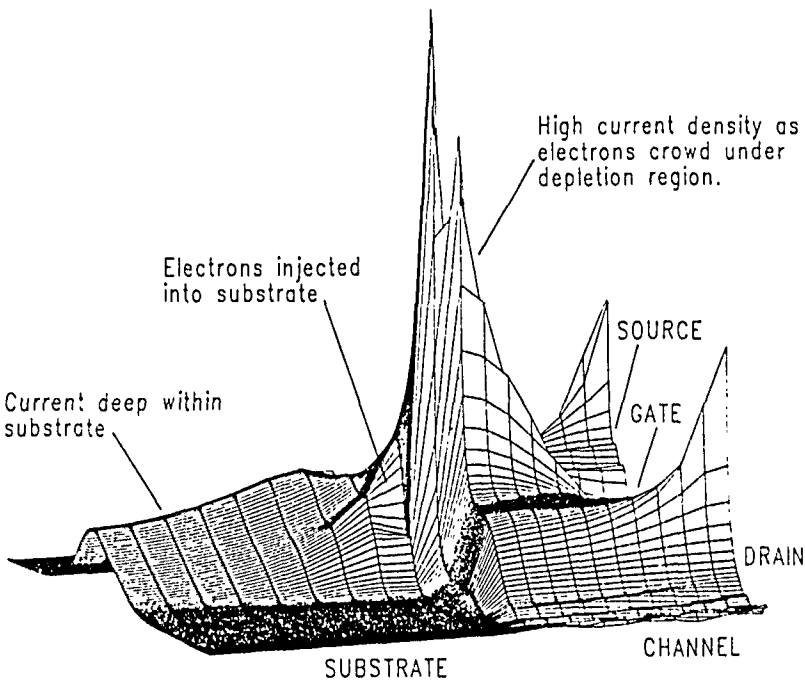


Figure 8 is a diagram of the total current density throughout the recessed gate device, when the device is nearly pinched off. The channel current in this device is 30 ma/mm, and the drain-source potential is 4.0 volts. In this device nearly all of the current between the drain and the source is flowing in the buffer region, as the depletion region extends through the channel into the buffer. This uncharacteristically high substrate current density is a

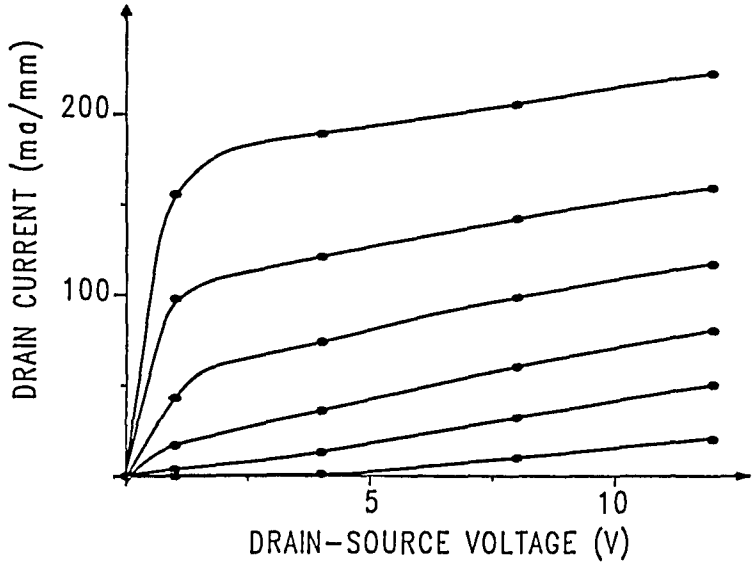


Figure 9: Calculated D.C. Transfer Characteristic for 0.5um Gate Planar MESFET.

feature which becomes more significant at shorter gate lengths. Figure 9 shows the transfer characteristic of a similar device, with a gate length of 0.5μm. Two features are immediately apparent when these results are compared to experimental measurements. First, the calculated output conductance is very high, and second, the pinch-off voltage is large, being over 5 volts. In addition, at a gate bias of 5 volts, the drain current is pinched off for drain potentials less than about 4 volts, after which a source-drain current is seen to flow which increases as the drain potential increases. These features result from the high substrate current calculated by the simulation program. The high output conductance arises due to the fact that the substrate current is predominantly modulated by the drain potential, not the gate potential, as is the channel current. The high pinch-off voltage is caused by the current flow within the substrate which, even at high gate bias, is little affected by the gate.

Simulation of a device consisting of only the active channel portion of the above device (without the buffer/substrate) confirms this conclusion. The pinch-off voltage as calculated by the simulation program is now much closer to that calculated using a one-dimensional model, and the output conductance is greatly reduced. The trends indicate by the computer simulation are essentially correct. It is often found that the pinch-off voltage, and the output conductance, are larger for shorter gate length devices.

This anomalous behaviour is to a large extent to be due to the form of the electron velocity-field characteristic used in the the simulation. This is based on Hilsum's empirical relationship between low field mobility μ_o and carrier concentration [5], giving a low-field mobility

$$\mu_o = \frac{0.8}{\left(1 + \frac{N_D}{10^{23}}\right)^{\frac{1}{2}}} \qquad (5)$$

The equilibrium carrier concentration in the buffer, which is usually small, is taken to be the same as the effective donor concentration, N_D, in the substrate. The low field mobility in the substrate is thus calculated to be very high, approximately $0.8 m^2 V^{-1} s^{-1}$, as opposed to $0.4 m^2 V^{-1} s^{-1}$ in the active channel.

Recent GMR mobility measurements on FET's have, however, indicated that this may not be the case. While the low field mobility in the active channel is close to that calculated from equation (5), that in the buffer region drops rapidly to a value of less than $0.2 m^2 V^{-1} s^{-1}$, in stark contrast to that calculated bu the Hilsum's relationship. This is probably due to the high level of compensation in the buffer regions of most FET's, while the carrier concentration is nearly that of intrinsic GaAs, the donor and acceptor concentrations can be very high.

Preliminary testing of this conclusion using a planar device simulation program by Snowden [6] has indicated that reducing the mobility in the substrate has the effect of reducing both the output conductance and the pinch-off voltage of the simulated FET. At present, results are not available from the current software using such a revised mobility profile, although it is felt that a similar behaviour will be seen.

6. Conclusions

The numerical simulation package described here may be used to simulate a variety of device geometries. Automatic grid generation based upon the geometry of the device to be simulated, and automatic refinement based upon previously

calculated solutions, enables the program to be used, without modification, to simulate different devices. Comparison of FORTRAN and PASCAL implementations of the program has shown that the flexibility of the PASCAL language and data structures enables a better than two-fold increase in efficiency over the FORTRAN implementation, thereby increasing the flexibility of the program.

The simulation has been applied to both a planar and a recessed gate GaAs MESFET, both with gate lengths of 1.0μm. The results indicated that the recessed gate device had a substantially higher I_{DSS} than the planar device, and that the variation of saturated drain current with gate potential was more non-linear in the recessed gate device. An uncharacteristically high substrate current density is noted, which becomes even more apparent at shorter gate lengths. It is found that Hilsum's empirical relationship between carrier concentration and low-field mobility may be inadequate in the highly compensated buffer layer found in many FET's, as it gives a mobility which is much higher than that indicated by GMR mobility measurements on FET's.

Applications of this program are in the area of semiconductor device design. The flexibility of the software makes it ideal for calculating the effects of changes made to the device parameters during the design and manufacturing process, without the use of costly iterative design loops.

Acknowledgement

The authors would like to thank the General Electric Company, Hirst Research Centre, East Lane, Wembley, Middlesex, U.K. for their financial support of this work. The authors would also like to thank Dr. P.H. Ladbrooke of that organisation for his helpful

References

[1] SNOWDEN, C.M.,
"Semiconductor Device Modelling",
Rep. Prog. Phys., Vol. 48, pp 223-275, 1985.

[2] SELBERHERR, S.,
"Analysis and Simulation of Semiconductor Devices",
Springer-Verlag, Wien, 1984.

[3] SCHARFETTER, D.L., and GUMMEL, H.K.,
"Large-Signal Analysis of a Silicon Read Diode Oscillator",
IEEE Trans. Elect. Dev., Vol. ED-16, No. 1, January 1969, pp. 64-77.

[4] REISER, M.,
"Large-Scale Numerical Simulation in Semiconductor Device Modelling",
Comp. Meth. Appl. Mech. Eng., Vol. 1, pp 17-38, 1972.

[5] HILSUM, C.,
"Simple Empirical Relationship Between Mobility and Carrier Concentration"
Electronic Letters, Vol. 10, NO. 12, p259, 1974.

[6] SNOWDEN, C.M.,
"Two-Dimensional Modelling of Non-Stationary Effects in GaAs MESFET's"
This Publication.