Copyright © 1981, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

A UNIFIED CAD MODEL FOR MOSFETS

by

S. Liu

Memorandum No. UCB/ERL M81/31

20 May 1981

A UNIFIED CAD MODEL FOR MOSFETS

by

Sally Liu

Memorandum No. UCB/ERL M81/31

20 May 1981

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

ABSTRACT

The operation of Metal-Oxide-Semiconductor Field-Effect Transistors is analyzed, emphasizing the effects of quantum-mechanical statistics, the joining of strong- and weak-inversion regions and the modeling of small-geometry devices.

When the gate voltage exceeds the threshold voltage, the surface-carrier concentration is higher than the doping concentration and the surface may degenerate. In order to understand the changes at the surface, we must explore the quantum-mechanical effects. The quantum-mechanical effects must be ascribed to both the quantum-mechanical statistics and the quantization of the energy band. Numerical evaluation of the drain current, channel conductance etc., based upon quantum-mechanical statistics, demonstrates that quantum-mechanical statistics alone do not result in a significant deviation from the classical prediction of these device characteristics within practical operational voltage ranges.

When the gate voltage changes from above to below the threshold voltage, the operational mode shifts from strong to weak inversion. The difficulties in modeling the transition between the strong- and weak-inversion regions, where no simple approximations can be applied, are overcome by joining the strong- and weak-inversion characteristics by properly defining the transition region. This approach provides an efficient and self-consistent way to simulate the operations of both strong- and weak-inversion regions.

Program TWIST has been developed to simulate the characteristics of weak inversion and weak-injection punchthrough by solving the two-

dimensional Poisson equation. The generation of non-uniform doping profiles including a two-dimensional impurity redistribution and graded meshes, and the application of modified Gummel's algorithm and Successive-Over-Relax iteration, together with a by-pass scheme, are implemented.

The punchthrough phenomena are explored by theoretical analysis and two-dimensional device simulations. The results of the two-dimensional simulations establish the relationship between the drain-induced lowering of the surface barrier and the punchthrough. The onset voltage of the punchthrough is derived from a quasi one-dimensional Poisson equation.

To provide an efficient model of small-geometry devices, a semi-empirical model, MOS3, has been developed and installed into the circuit simulation program SPICE2.G. The equations in a simple format allow easy parameter extraction, a property which is as critical as the accuracy of the model itself. A close correlation is obtained between the calculated and measured characteristics of small-geometry devices.

ACKNOWLEDGEMENTS

The author wishes to express her sincere appreciation to Professor D.O.Pederson for his continuing encouragement and guidance throughout the course of this work. She also gratefully acknowledges the numerous helpful discussions with Professors B.Hoefflinger, A.R.Newton and A.M.Portis. It is her pleasure to acknowledge the steady and helpful discussions with E.Cohen, A.Vladimirescu and the other members of the CAD group at the University of California, Berkeley.

Both the financial and technical support received from the Signetics Corporation and the encouragement provided by T.Young are gratefully acknowledged. Special thanks are extended to P.T.Chuang for many interesting discussions in the early stages of this work.

The predoctoral fellowship from IBM for the spring and fall of 1976 helped the preliminary development of this research. The generous equipment grant from the Hewlett-Packard Company and the extensive computer resources supplied by the Computer Center of the University of California, Berkeley are also acknowledged.

The author is especially indebted to Mrs. E.A.Baker for her careful proofreading of the entire manuscript.

The author wishes to express her deep gratitude to her parents, Kuei-Ju and Hui-Hsiang, for their support and encouragement throughout the course of this work.

TABLE OF CONTENTS

CHAPTER 1: Introduction 1
CHAPTER 2: The Quantum-Mechanical Effects on the Operation of MOSFETs 5
2.1. A Survey of the El-Mansy and Boothroyd Model 6
2.2. The Impact of Quantum-Mechanics 13
2.2.1. The Wave Property and the Band Quantization 14
2.2.2. Fermi-Dirac Statistics 15
2.2.3. The Impact on Device Characteristics 17
CHAPTER 3: A Unified Approach to MOSFET Modeling 19
3.1. General Overview 20
3.2. Strong-Inversion Region 25
3.3. Weak-Inversion Region 27
3.4. Join Together 28
3.5. The Influence of Fast-Surface States 31
CHAPTER 4: Two-Dimensional Simulations of Small-Geometry MOSFETs 33
4.1. Overview of Program TWIST 35
4.2. Graded Mesh and Impurity Distribution 36
4.3. Basic Equations and Boundary Conditions 37
4.4. Quasi-Fermi Level 39
4.5. Potential Initialization 42
4.6. Iteration Algorithm and Program Performance 44
4.7. Device Characteristics 45
CHAPTER 5: The Punchthrough 47
5.1. Static Feedback and Punchthrough 48

5.2. The Saddle Point 50
5.3. Quasi-One-Dimensional Analysis 51
CHAPTER 6: Small-Geometry Effects and the Semi-empirical Model MOS3 56
6.1. Threshold Voltage 58
6.1.1. Short-Channel Effects 59
6.1.2. Narrow-Channel Effect 60
6.1.3. The Model Equations 61
6.2. Basic Drain-Current Equation 62
6.3. Surface-Mobility Modulation by Gate Voltage 64
6.4. Velocity Saturation of Hot Electrons 64
6.5. Saturation Voltage 65
6.6. Channel-Length Modulation 67
6.7. Capacitance Model with Charge Conservation 68
6.8. The Comparison Between MOS2 and MOS3 69
CHAPTER 7: Summary 73
APPENDIX A: TWIST User's Guide 77
APPENDIX B: Example Input to Program TWIST 90
APPENDIX C: Example Console Session of Program TWIST 92
APPENDIX D: Test-Circuit Inputs to SPICE2.G 97
APPENDIX E: Listing of Program TWIST 105
REFERENCES 179

CHAPTER 1

Introduction

A Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET), also commonly known as an Insulated-Gate Field-Effect Transistor (IGFET), is a semiconductor device with four terminals: source, drain, gate and substrate. It is the major ingredient of today's Large- and Very-Large-Scale Integrated Circuits (LSI/VLSI). The extensive development of MOS technology is a result of the simplicity of its structure. This simple structure permits low production costs and high packing density which make MOS integrated circuits very economical.

The invention of MOSFETs can be traced to the 1930's [1-2]. The first reported laboratory study of MOSFETs was carried out in the 1940's [3]. Commercial MOSFETs became available in the 1960's [4], after the development of planar integrated-circuit technology [5]. Since then, MOS technology has been developed rapidly [6-7]. First, the polysilicon gate replaced the metal gate. Then ion implantation [8-10] replaced thermal diffusion and the devices could be tailored with much more freedom and precision. The devices are getting smaller and smaller. Today's typical channel length and width are as small as $2\mu m$. Now, people are looking forward to the era of devices as small as or even smaller than $1\mu m$.

Theoretical research on MOSFETs and technological improvements have always gone hand in hand. Detailed physical analysis has led to better understanding of device operations and to the development of transistor models that are widely used in circuit simulation programs. Device models which are compatible with these simulators have gained very much attention. Characteristics equations with sets of device parameters are often used [11].

Table-Look-Up is an alternative when dealing with large-scale circuits [12-13]. Both approaches to transistor modeling require either available device data from which parameters may be extracted [14] or parameter values obtained from one-dimensional or pseudo-two-dimensional physical models of the transistors [15]. Numerous models have been developed [6-7]. The complexity ranges from the most compact Shichman-Hodge's [16] model to those models requiring iterative solutions [17]. A good CAD (Computer-Aided-Design) model should be able not only to reflect the state of the technology but also to provide accuracy and computational efficiency.

Most MOSFET circuits are designed so that the devices operate in the strong-inversion region. The strong-inversion region is the region in which the concentration of the minority carriers exceeds that of the majority carriers, which is in the range of 5.0×10^{14} to 5.0×10^{16} cm^{-3}. The presence of excessive minority carrier concentration inverts the type of net surface concentration. The threshold voltage of strong inversion is usually on the order of 1 volt. Degeneracy, the condition in which the surface concentration is $10^{19}cm^{-3}$ or more, does not occur until a much higher gate voltage is reached. The impact of this heavy concentration and the effect due to quantum mechanics are explored in Chapter 2.

The current in the weak-inversion region is low. The transition between strong- and weak-inversion regions deserves attention. The major difficulty in modeling the transition region is the fact that no simple physical approximation can be applied. In the transition region, the contributions from the minority and majority carriers are comparable. Both the diffusion and drift currents are equally important. Chapter 3 describes a properly defined transition region in which the strong- and weak-inversion characteristics are joined. This approach does not involve internal iterations and provides both efficiency and accuracy in circuit simulations.

Numerical solutions of the two-dimensional potential and current-continuity equations are necessary to describe the new generation of integrated MOSFETs, which are shorter and/or narrower than the old ones. People have been working on these subjects for the past ten years [18-20]. Initially, idealized impurity distributions and junction boundaries were assumed to facilitate solutions. However, with very small device geometries, modern process simulators show extremely inhomogeneous two-dimensional impurity distributions and junction boundaries, which must then be considered in the potential and current-transport simulations. Program TWIST (TWo-dimensional Interactive Simulation of MOS Transistors) is an interactive device simulation program which handles the solution of the two-dimensional Poisson equation. Because it limits the solution to the Poisson equation only, TWIST is useful both as a pre-selector for structures to be simulated with a more elaborate two-dimensional potential and current-transport program and as an efficient simulation tool for the conditions of weak inversion and/or weak-injection punchthrough. The development of Program TWIST is described in Chapter 4.

One of the most important problems in designing small-geometry MOSFETs is the punchthrough between the source and the drain. This is the result of barrier lowering due to the merging of the source and drain depletion regions. Once the punchthrough condition is reached, the current flowing from the source to the drain increases significantly as V_{DS} increases, and the device characteristics deviate from the norm. This additional current can be viewed as an undesirable component to be avoided or exploited as part of the conduction current in novel applications of MOSFETs. In Chapter 5, the punchthrough phenomenon is explored by both theoretical analysis and two-dimensional device simulations using Program TWIST. The derivation of the onset voltage of punchthrough is based upon the assumption of a uniform substrate doping profile.

The ultimate difficulty in the development of a model suitable for small devices is the correct treatment of the two-dimensional nature of the potential distribution and current flow. A semi-empirical modeling approach is a compromise between simulation accuracy and computational efficiency. The equations in a simple format allow easy parameter extraction, a property which is as critical as the accuracy of the model itself. The MOS3 model has been developed and implemented into the circuit simulation program SPICE2 to provide an accurate model of MOSFETs no larger than $L \leq 2\mu m$ and/or $W \leq 2\mu m$, and to attain computational efficiency.

In developing the MOS3 model, several important issues of MOSFET modeling were considered. Model equations were developed and verified. In the future, modeling work should emphasize the small-geometry effects. Further development of two-dimensional device simulation programs would be very helpful.

CHAPTER 2

The Quantum-Mechanical Effects on the Operation of MOSFETs

In the El-Mansy and Boothroyd model [21-22], hereafter abbreviated the E-B model, one of the major issues is the impact of quantum-mechanical statistics on device characteristics and modeling. The E-B model and the nature of the quantum-mechanical effects are examined in this chapter.

The E-B model is a charge-moment model based upon classical statistics and the assumption of a block-charge distribution. The authors argue that the widely observed channel-conductance modulation by the gate voltage can be attributed to quantum-mechanical statistics, not the surface-mobility modulation. But, in this aspect, their model does not perform better than models using empirical mobility equations.

The quantum-mechanical effects should be attributed to both the degeneracy of the surface carrier population, which can be described properly only by Fermi-Dirac statistics, and the wave property of the carriers, which leads to the quantization of energy band at degeneracy. The statistical impact alone is not large enough to cause a significant difference in the device characteristics in the practical operational range. The onset voltage of degeneracy is calculated. The surface potentials based upon different statistics vary by as much as 30 percent in a degenerate state, but the difference in the device characteristics is less than 1 percent.

An empirical expression, which relates the surface potential to the charge density per unit area in the degenerate case, is proposed. This expression is correct within three percent for Fermi levels within +4 to -4 $\frac{kT}{q}$ from the energy-band edge. This range, at room temperature, goes up

to approximately $(V_{GB} - V_{FB}) = 30V$ which is well above the maximum bias of practical MOSFET analog circuits. This expression is chosen so that it is differentiable and integratable and permits efficient computations.

2.1. A Survey of the El-Mansy and Boothroyd Model

The E-B model is an analytical model which covers both the strong- and weak-inversion regions of enhancement MOSFETs with a uniformly doped substrate. Basically, it differs from other models in its assumption of charge distribution rather than in its treatment of fundamental statistics. This point has been debated by Brews [23].

An N-channel MOSFET is shown in Figure 2.1. In that figure, L is the distance between the source and drain junctions, W is the width of the channel region, T_{OX} is the oxide thickness, x_j is the depth of the source and drain junctions, and x and y are the spatial coordinates to be used in later analysis. In the E-B model, in order to deal with the spatial charge distribution effect on the potential distribution, the total charges inside the semiconductor, including both the inversion and depletion charges, assume a step distribution [24] of concentration N_S and depth W_S as shown in Figure 2.2. In the case of an N-channel device, the potential, ϕ_s, sustaining this block of charge can be expressed as:

$$\phi_s = \frac{qN_S}{2\epsilon_{si}} W_S^2 \tag{2.1}$$

which is also the surface potential with the substrate potential taken as the reference point. The amount of gate charge per unit area, $Q_g = C_{OX}(V_{GB} - V_{FB} - \phi_s)$, must equal the amount of the total semiconductor charge per unit area, $Q_{SI} = -qN_S W_S$, to satisfy the constraint of charge neutrality. Therefore:

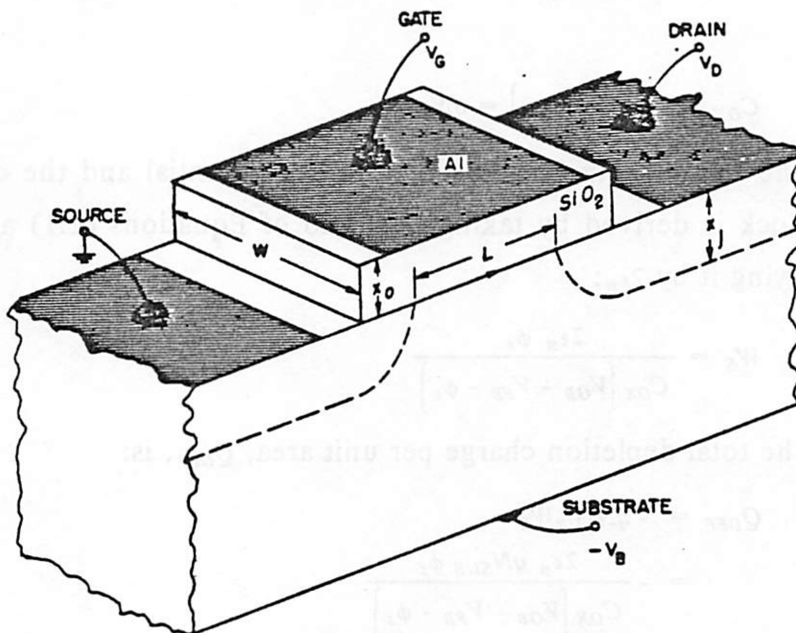


Figure 2.1 The Schematic Diagram of a MOSFET,

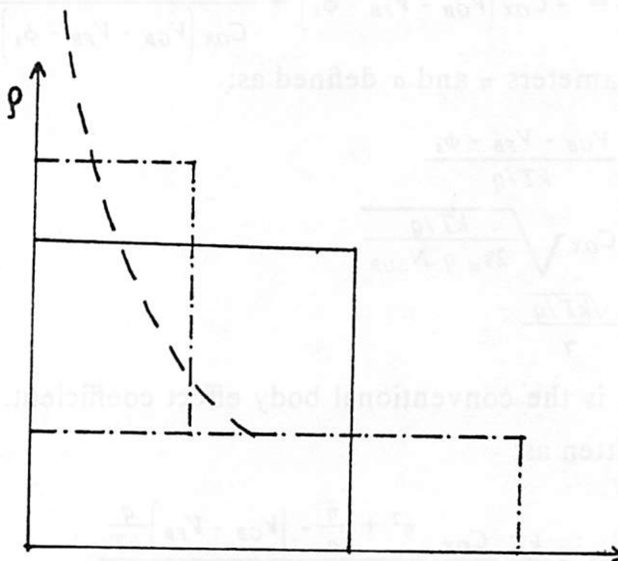


Figure 2.2 The Block Approximation of Charge Distribution in the E-B Model,

$$C_{OX}\left(V_{GB} - V_{FB} - \phi_s\right) = qN_S W_S \tag{2.2}$$

The relationship between the surface potential and the depth of this charge block is derived by taking the ratio of Equations (2.1) and (2.2) and multiplying it by $2\epsilon_{si}$:

$$W_S = \frac{2\epsilon_{si}\,\phi_s}{C_{OX}\left(V_{GB} - V_{FB} - \phi_s\right)} \tag{2.3}$$

The total depletion charge per unit area, Q_{DEP}, is:

$$Q_{DEP} = -qN_{SUB}W_S \tag{2.4}$$

$$= -\frac{2\epsilon_{si}\,qN_{SUB}\,\phi_s}{C_{OX}\left(V_{GB} - V_{FB} - \phi_s\right)} \tag{2.5}$$

where N_{SUB} is the substrate concentration. The inversion charge density, Q_{INV}, equals the negative sum of the gate and depletion charge densities:

$$Q_{INV} = -\left(Q_g + Q_{DEP}\right) \tag{2.6}$$

$$= -C_{OX}\left(V_{GB} - V_{FB} - \phi_s\right) + \frac{2\epsilon_{si}\,qN_{SUB}\,\phi_s}{C_{OX}\left(V_{GB} - V_{FB} - \phi_s\right)} \tag{2.7}$$

with parameters η and a defined as:

$$\eta = \frac{V_{GB} - V_{FB} - \phi_s}{kT/q} \tag{2.8}$$

$$a = C_{OX}\sqrt{\frac{kT/q}{2\epsilon_{si}\,q\,N_{SUB}}} \tag{2.9}$$

$$= \frac{\sqrt{kT/q}}{\gamma} \tag{2.10}$$

where γ is the conventional body effect coefficient. The expression Q_{INV} can be rewritten as:

$$Q_{INV} = -\frac{kT}{q}\frac{C_{OX}}{a}\,\frac{\eta^2 + \frac{\eta}{a} - \left(V_{GB} - V_{FB}\right)\frac{q}{kT}}{\eta} \tag{2.11}$$

This is Equation (25) in Reference [21]. Here, it has been derived solely based upon the assumption of block-charge distribution and is independent of the statistics on which the model is based.

The drain current is the most important device characteristic and can be obtained by integrating the total current density, $J(x,y)$, over a cross section normal to the channel direction:

$$I_{DS} = W \int_0^{W_y} J(x,y)dx \tag{2.12}$$

where W_y is the width of the surface charge layer at location y. The total current density is the sum of the drift and diffusion current densities:

$$J(x,y) = qU_{EFF}\left[N(x,y)E_y + \frac{kT}{q}\nabla_y N(x,y)\right] \tag{2.13}$$

Based upon Boltzmann statistics, the electron density is related to the quasi-Fermi level, ζ_n, which measures the excess charge induced by the external biases, as shown in the following expression:

$$N(x,y) = N_o e^{\frac{q(\phi(x,y)-\zeta_n)}{kT}} \tag{2.14}$$

where N_o is the electron density at zero biases. The total current density can be rewritten as:

$$J(x,y) = -qU_{EFF}N(x,y)\nabla_y\zeta_n \tag{2.15}$$

which is Equation (21) in Reference [21]. The drain current equation becomes:

$$I_{DS} = WqU_{EFF}\int_0^{W_y} N(x,y)\frac{\partial\zeta_n}{\partial y}dx \tag{2.16}$$

$$= -WU_{EFF}Q_{INV}\frac{\partial\zeta_n}{\partial y} \tag{2.17}$$

By integrating both sides of the above equation from source to drain in variable y, one arrives at Equation (24) of Reference [21]:

$$I_{DS} = -\frac{W}{L}U_{EFF}\int_0^{V_{DS}} Q_{INV}d\zeta \tag{2.18}$$

As noted before, Q_{INV} is an explicit function of ϕ_s. In order to carry out the integration in the above equation, ϕ_s must be related to the biases and the

quasi-Fermi level. This is achieved by integrating the differential Poisson equation in the potential domain from 0 to ϕ_s with the boundary condition of zero electric field at the boundary of zero potential:

$$\int_0^{\phi_s} \nabla_x^2 \phi \, d\phi = \frac{q}{\epsilon_{si}} \int_0^{\phi_s} \left[N - N_{SUB} \right] d\phi \tag{2.19}$$

$$\frac{1}{2} \left[\frac{\partial \phi}{\partial x} \right]_s^2 = \frac{q}{\epsilon_{si}} \int_0^{\phi_s} N(\phi) d\phi - \frac{q}{\epsilon_{si}} N_{SUB} \phi_s \tag{2.20}$$

It can be further rewritten as:

$$\int_0^{\phi_s} N(\phi) d\phi = \frac{Q_{SI}^2}{2\epsilon_{si} q} + N_{SUB} \phi_s \tag{2.21}$$

By applying the classical definition of the quasi-Fermi level, carrying out the integration, and replacing the expression Q_{SI}, Equation (8) in Reference [21] is obtained:

$$\frac{N_I^2}{N_{SUB}} \frac{kT}{q} e^{\frac{q(\phi_s - \zeta_n)}{kT}} = \frac{\left[C_{OX} \left(V_{GB} - V_{FB} - \phi_s \right) \right]^2}{2\epsilon_{si} q} N_{SUB} \phi_s \tag{2.22}$$

ϕ_s must be solved from the above equation by either iteration or simplifying the equation. The integration in Equation (2.18) can be carried out:

$$I_{DS} = \frac{W}{L} U_{EFF} C_{OX} I^o \tag{2.23}$$

where

$$I^o = \left(V_{GB} - V_{FB} + 2\frac{kT}{q} + \gamma^2 - \frac{\phi_{s,S} + \phi_{s,D}}{2} \right) \left(\phi_{s,D} - \phi_{s,S} \right) - \gamma^2 \left(\gamma^2 - V_{GB} \right) \left[\frac{V_{GB} - V_{FB} - \phi_{s,D}}{V_{GB} - V_{FB} - \phi_{s,S}} \right] \tag{2.24}$$

which is equivalent to Equation (15) in Reference [21]. Here $\phi_{s,S}$ and $\phi_{s,D}$ are the surface potentials derived from Equation (2.20) at the source and the drain, respectively. Equation (2.24) has been intentionally rewritten in a format similar to that of conventional equations for a convenient comparison.

It might be argued that it is not necessary to introduce the classical definition of the quasi-Fermi level, which contradicts the quantum-mechanical concept put forth by El-Mansy and Boothroyd. To accommodate their argument, Equation (21) of Reference [21] must be derived using a quantum-mechanical approach. Nonetheless, the authors of the E-B model maintain that because the properties of the source and drain junctions are well described by Boltzmann statistics, the first-moment of charge distribution based upon different statistics should be the same at the junction boundaries to provide a smooth transition. They further argue that this is a general property of the Poisson equation and can be applied to the derivation of the drain current equation. This direct application of the boundary condition to the channel region is confusing and contradictory to their main theme. Both Equations (8) and (21) in Reference [21] are the results of classical statistics.

The essence conveyed by the combination of the step charge distribution and the classical voltage-potential relationship can be properly designated as a classical charge-moment model [25]. In the charge-block picture, the total amount of charge inside the semiconductor is the weighted sum of the inversion and depletion charges:

$$Q_{SI} = \sigma_{INV} Q_{INV,o} + \sigma_{DEP} Q_{DEP,o} \tag{2.25}$$

where the weighting factors σ_{INV} and σ_{DEP} are defined as:

$$\sigma_{INV} = \frac{W_S}{W_{INV}} \tag{2.26}$$

$$\sigma_{DEP} = \frac{W_S}{W_D} \tag{2.27}$$

and

$$W_{INV} = 2\frac{\int_0^\infty x\,\rho_{INV}(x)dx}{\int_0^\infty \rho_{INV}(x)dx} \tag{2.28}$$

$$W_D = \frac{\int_0^\infty \rho_{DEP}(x)dx}{N_{SUB}} \tag{2.29}$$

σ_{INV} is a number greater than one, while σ_{DEP} is a number less than one. W_{INV}, W_D and W_S versus V_{GB} - V_{FB}, and σ_{INV} and σ_{DEP} versus V_{GB} - V_{FB} are plotted in Figures 2.3 and 2.4, respectively, for comparison. In the strong-inversion region, the weighting factor of the depletion charge decreases as the gate voltage increases. Thus, the depletion charge in the E-B model diminishes as V_{GB} increases. Nonetheless, it should not be interpreted as a diminution of the real depletion charge.

The charge-block representation enables the E-B model to replace the double integration in the Pao-Sah theory [26] by a closed-form expression and establishes it as a practical CAD model. But quantum mechanics are not involved in the derivation of the model.

The E-B model also differs from the other models in that it uses a constant, instead of modulated, surface mobility. The drain conductance can be derived by differentiating the drain current equation with respect to the drain to source voltage, V_{DS}:

$$G_{DS} = -\frac{W}{L}U_{EFF}Q_{INV}(drain) \tag{2.30}$$

$$= -\frac{W}{L}U_{EFF}\left[Q_{SI} - Q_{DEP}\right] \tag{2.31}$$

Clearly, the drain-conductance modulation may result from variations in the surface mobility, the surface potential or the depletion charge or in all three of them. In the E-B model, the drain-conductance modulation is the consequence of repartitioning charges. The weighting factor of the depletion

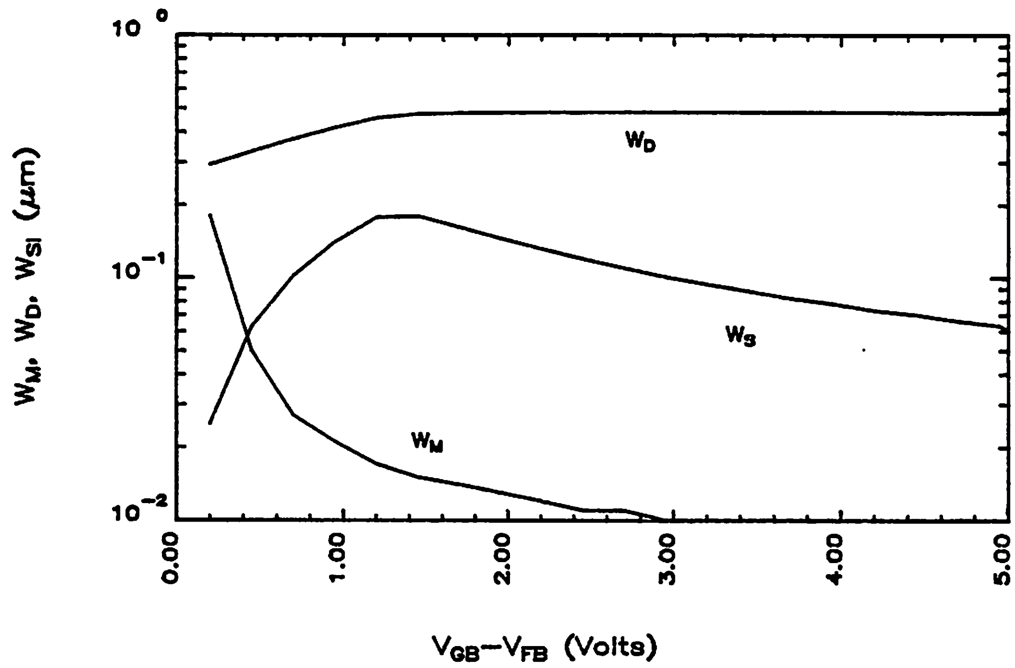


Figure 2.3 The Step Widths of Depletion and Inversion Charges and the Block Charge in the E-B Model,

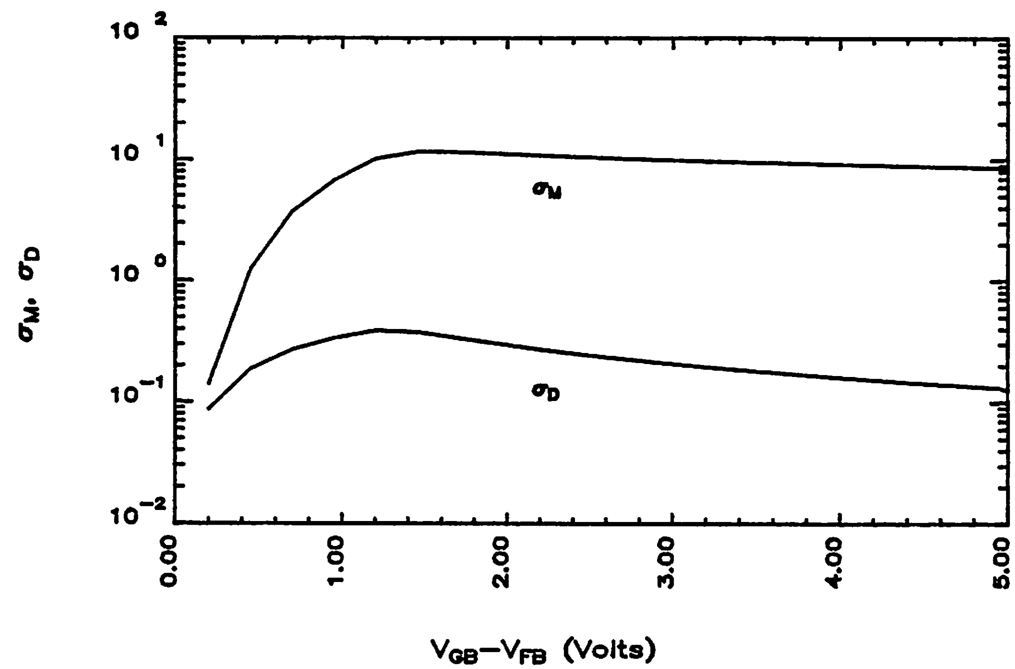


Figure 2.4 σ_{INV} and σ_{DEP}, Weighting Factors of Inversion and Depletion Charges, in the E-B Model,

charge is sensitive to the gate voltage as indicated in the formulation of Q_{DEP}. Conventionally, the drain-conductance modulation is modeled by empirical mobility modulation. In this aspect, the E-B model is less flexible than models using empirical mobility-modulation concept. The I_{DS}-versus-V_{DS} curves of a $50\mu m$ by $9\mu m$ MOSFET are plotted in Figure 2.5. The measured data is compared with both the E-B model and the MOS2 model in SPICE2D, which uses the empirical mobility-modulation approach. At the low bias range of 0 to 0.5 volts V_{DS}, all the second-order effects are negligible and the simulated characteristics are dominated by the basic assumption. The E-B model consistently overestimates the current. The difference increases as the gate voltage increases. The model parameters of this device are chosen so that the models fit adequately in the range of 0 to 10 volts V_{DS} as shown in Figure 2.6. The parameter values are listed in Table 2.1:

Table 2.1

Parameter	MOS2	E-B	Unit
W	50	50	um
L	9	9	um
VTO	0.45	0.45	V
TOX	950	950	A
XJ	2.2	2.2	um
LD	0.75	0.75	
NSUB	6E15	3E15	cm-3
UO	630	550	cm/V-s
UEXP	0.27	-	
UCRIT	60K	-	V/cm

Table 2.1 Device Parameters Used in Figures 2.5 and 2.6

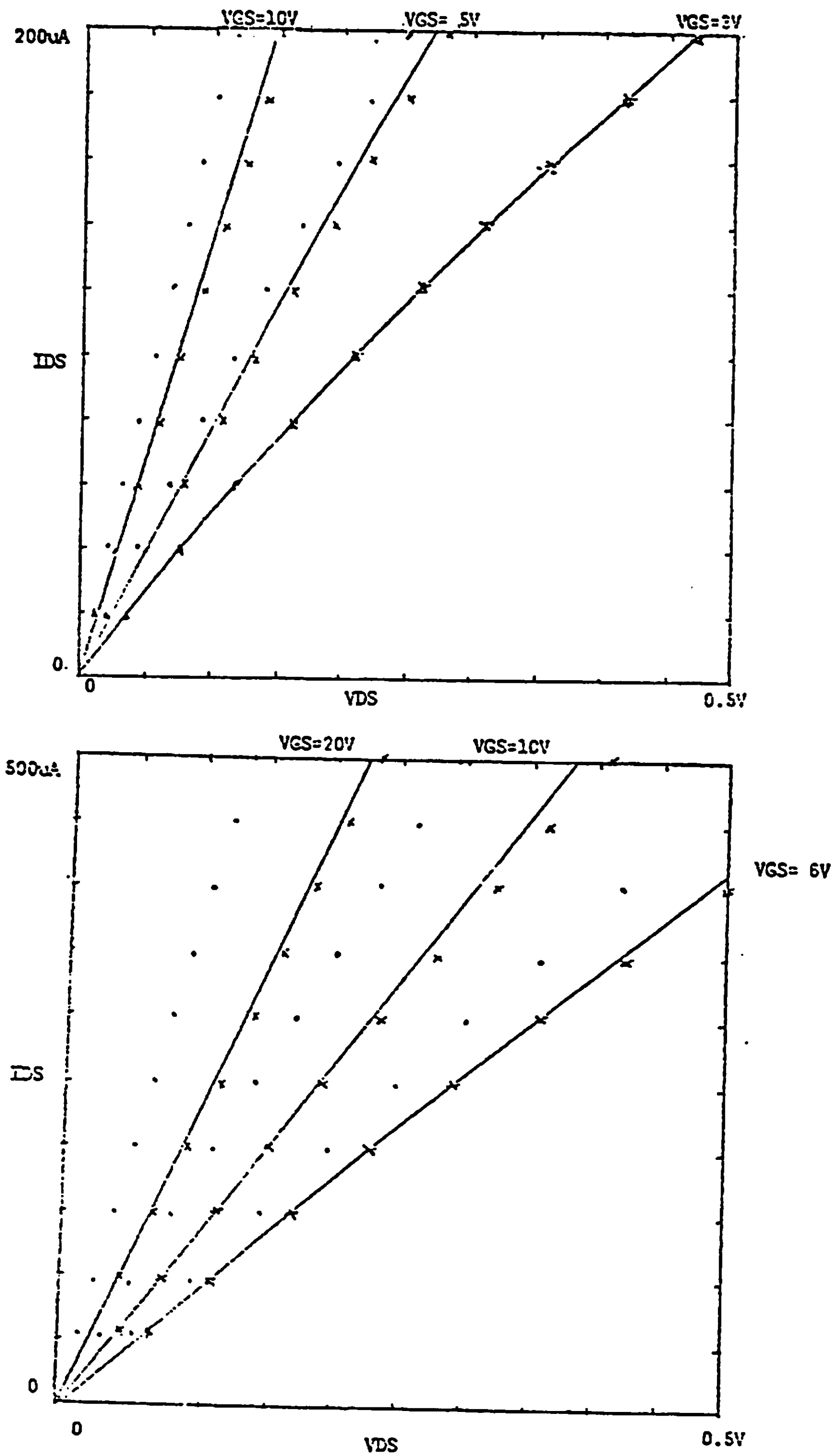


Figure 2.5 I_{DS}-Versus-V_{DS} of the Experimental Measurements and the MOS2 and E-B Models with Parameter V_{GS} in a Low V_{DS} Range,

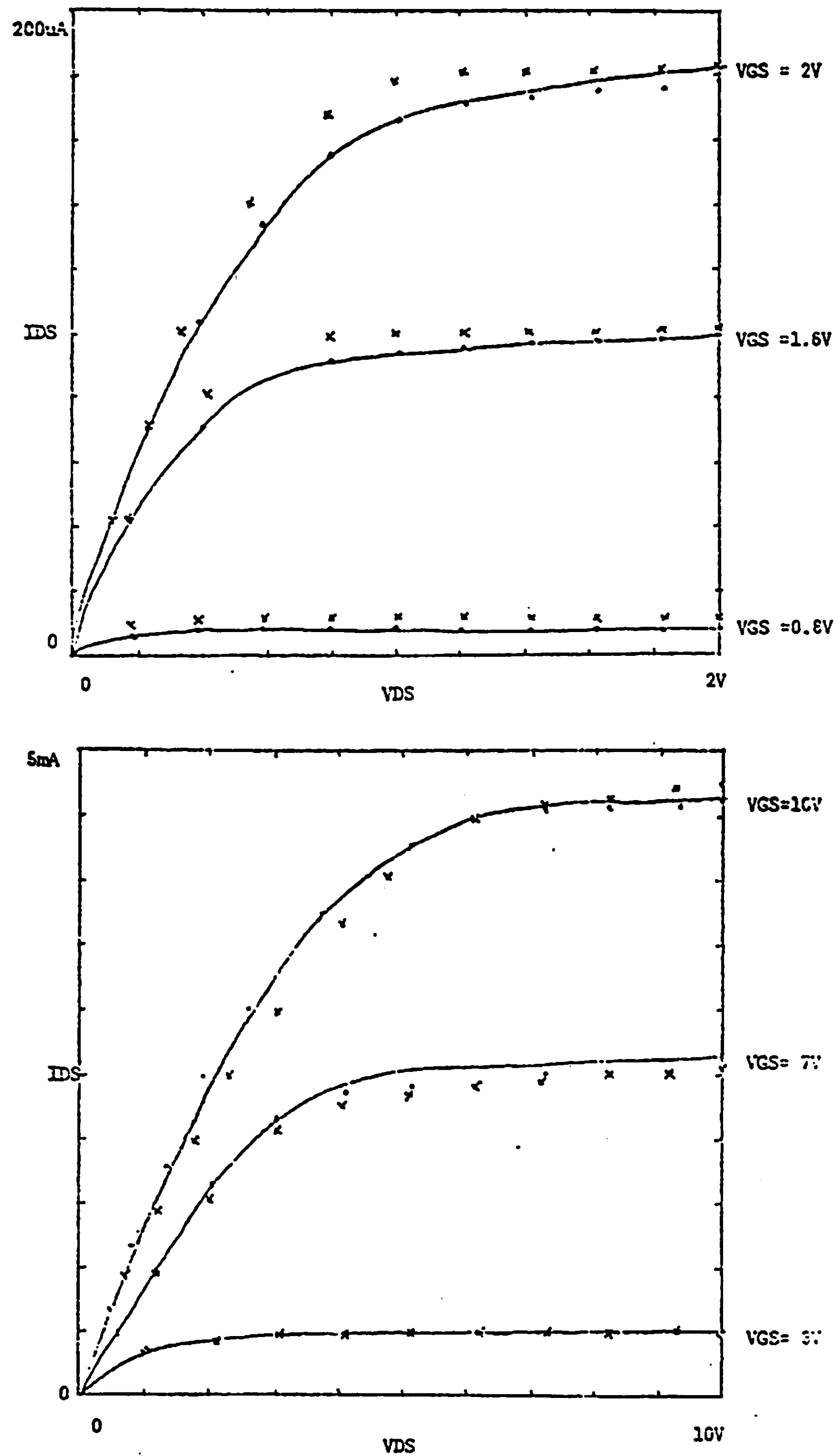


Figure 2.6 I_{DS}-Versus-V_{DS} of the Experimental Measurements and the MOS2 and E-B Models with Parameter V_{GS} in a High V_{DS} Range,

2.2. The Impact of Quantum-Mechanics

Equation (2.21) is a general derivation from the Poisson equation. Quantum-Mechanical statistics and the noticeable wave property in degeneracy induce different carrier distributions and integration results that deviate from the classical ones. A new relationship between the surface potential and the biases is required in order to handle the quantum-mechanical effects properly.

The surface potential measures the conducting carriers. Its distribution determines how the current will flow between the source and the drain. The surface potential predicted by quantum mechanics is more sensitive to the applied gate voltage than the classical statistics predicted.

The quantum-mechanical effects can be attributed to both the highly degenerate surface concentration, which can be properly described only by Fermi-Dirac statistics, and the wave property, which leads to the quantization of the surface energy band. The combined impact of both the statistics and the wave property has been studied by linearizing the surface-potential-well to decouple the Poisson and the Schrödinger equations [27], and using perturbation techniques to include the non-linearity of the potential well in the highly-degenerate case [28].

The maximum of the electron distribution function is located inside the semiconductor instead of at the surface which is predicted by Boltzmann statistics without including the effects of the Schrödinger equation. The peak concentration is much less than the classical one. As reported by the work of Talley et al. [27], the channel thickness decreases as the gate voltage increases, and stays at a finite thickness of approximately $0.01 \mu m$. While classical statistics also predicts a decreasing channel thickness, but the thickness diminishes in the very strong-inversion region.

Pals [29] has measured the characteristics of capacitance versus gate voltage to determine the charge distribution inside a semiconductor. His results show that the quantum-mechanical considerations must be included in the theoretical calculations in order to match the experimental data. The existence of sub-bands has been demonstrated by magneto-resonance measurements [30] on P-channel MOSFETs.

2.2.1. The Wave Property and the Band Quantization

The influence of the wave property has been explored by precise numerical calculations based upon the Schrödinger and Poisson equations. The results of the existing studies are reviewed in the following paragraphs.

When the gate voltage is high enough to induce a degenerate surface carrier concentration, the wave property of the carriers becomes noticeable. The impenetratable potential barrier at the interface forces the interface be a node point of the standing wave pattern associated with the carrier distribution function. As shown in Figure 2.7, the fundamental energy states, which have the highest occupation probability, tend to place the peak carrier concentration close to the center of the surface potential well. So the carrier distribution based upon quantum-mechanical calculations has its peak concentration inside the semiconductor.

As the surface concentration degenerates, the surface conduction band is quantized and splits into sub-bands along a direction normal to the surface. Each of the sub-bands corresponds to a two-dimensional continuum. The carrier motion is quantized along the normal direction, but is continuous along the parallel direction. The quantization effect is enhanced as the surface potential well is narrowed down at an increasing substrate bias. At

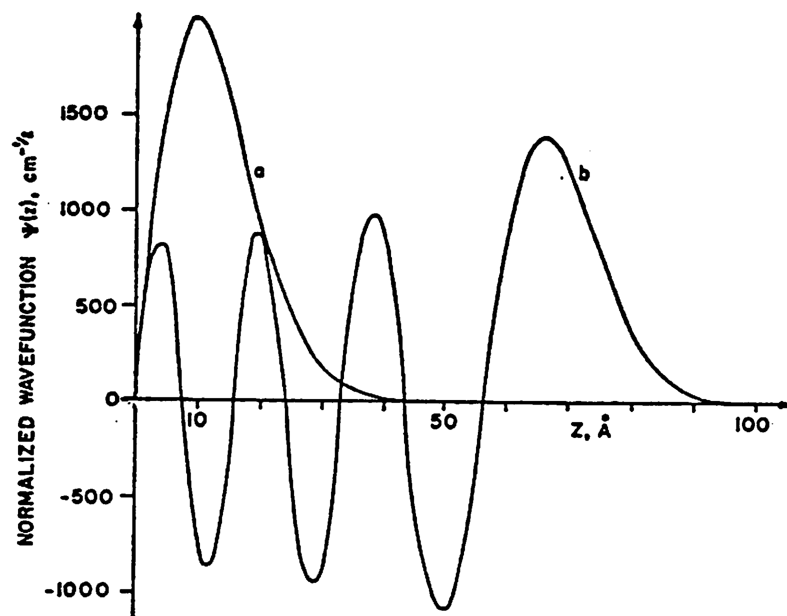


Figure 2.7 The Distribution Function of Inversion Carriers in the Low Energy States (from Reference [2.7]),

room temperature, when the surface just starts to invert, many of the sub-bands are populated and the quantization effect is not prominent. As the gate voltage increases, the surface's normal field strength, the differences among the quantized energy levels, and the relative population of the lowest sub-band all increase. Thus, the quantization effect is profound.

2.2.2. Fermi-Dirac Statistics

The carrier concentration can be expressed as:

$$N = \int_{E_0}^{E_1} N(E)\, F(E)\, dE \tag{2.32}$$

where $N(E)$ and $F(E)$ are the density of states and the distribution probability respectively. If the surface carrier concentration degenerates, the integration should be replaced by the summation over all the sub-bands, $N(E)$ by the corresponding density of each sub-band, and where $F(E)$ is Fermi-Dirac distribution function.

By using the Fermi-Dirac function, neglecting the sub-band quantization and assuming a spherical band structure, the above equation becomes:

$$N(\phi) = \frac{2}{\sqrt{\pi}} N_C F_{1/2}\left[\frac{E_F - E_C + q\left(\phi - \zeta_n\right)}{kT}\right] \tag{2.33}$$

where

$$F_{1/2}(x) = \int_0^{\infty} \frac{\eta^{1/2} d\eta}{1 + e^{\eta - x}} \tag{2.34}$$

and ϕ is the magnitude of band bending and ζ_n is the quasi-Fermi level. If the surface concentration does not degenerate, it can be reduced to:

$$N(\phi) = N_C e^{E_F - E_C + q(\phi - \zeta_n)/kT} \tag{2.35}$$

which is equivalent to Equation (2.14) with $N_o = N_C\, e^{(Ef - Ec)/kT}$. The

relationship between the surface potential and the gate voltage can be derived by substituting the expression $N(\phi)$ for the integrand on the left side of Equation (2.21). By using the approximations suggested by Seiwatz and Green [32], the integration can be reduced to [33]:

$$\int_0^{\phi_s} N(\phi)d\phi = \frac{kT}{q}N_C\frac{2}{\sqrt{\pi}}\frac{2}{3}F_{3/2}\left[E_F - E_C + q\frac{(\phi_s - \zeta_n)}{kT}\right] \tag{2.36}$$

where

$$F_{3/2}(x) = \int_0^{\infty}\frac{\eta^{3/2}d\eta}{1 + e^{\eta - x}} \tag{2.37}$$

The corresponding expression based upon Boltzmann statistics is:

$$\int_0^{\phi_s} N(\phi)d\phi = \frac{kT}{q}N_C\, e^{E_F - E_C + q\frac{(\phi_s - \zeta_n)}{kT}} \tag{2.38}$$

Functions $\frac{2}{\sqrt{\pi}}F_{1/2}(x)$ and e^x, which are counterparts based upon Boltzmann and Fermi-Dirac statistics, are plotted together in Figure 2.8. The quantities $\sqrt{\frac{2}{\sqrt{\pi}}\frac{2}{3}F_{3/2}(x)}$ and $e^{\frac{x}{2}}$, which correspond to the inversion charge density Q_{INV} but are based upon different statistics, are plotted in Figure 2.9. $F_{1/2}$ and $F_{3/2}$ are calculated by the Table-Look-Up method based upon the McDougall-Stoner table [34]. The difference between the results based upon Fermi-Dirac and Boltzmann statistics increases to approximately ten percent as the Fermi level approaches the band edge.

The relationship between the surface potential and the gate voltage can be deduced from Equation (2.21) and the expression Q_{SI} as:

$$V_{GB} = \phi_s + \sqrt{\gamma^2\phi_s + \frac{\gamma^2}{N_{SUB}}\int_0^{\phi_s} N(\phi)d\phi} \tag{2.39}$$

The onset voltage of degeneracy, V_{DEG}, can be defined as the gate voltage at which the Fermi level reaches the band edge. The onset voltage of degeneracy versus the oxide thickness is plotted in Figure 2.10 for cases of both

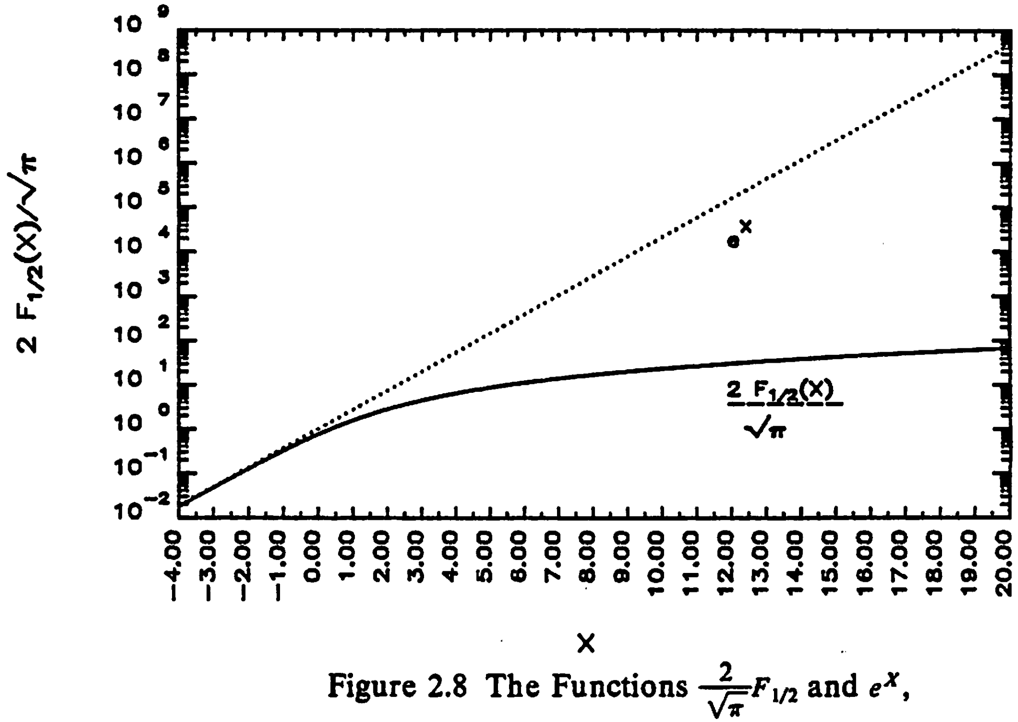


Figure 2.8 The Functions $\frac{2}{\sqrt{\pi}}F_{1/2}$ and e^X,

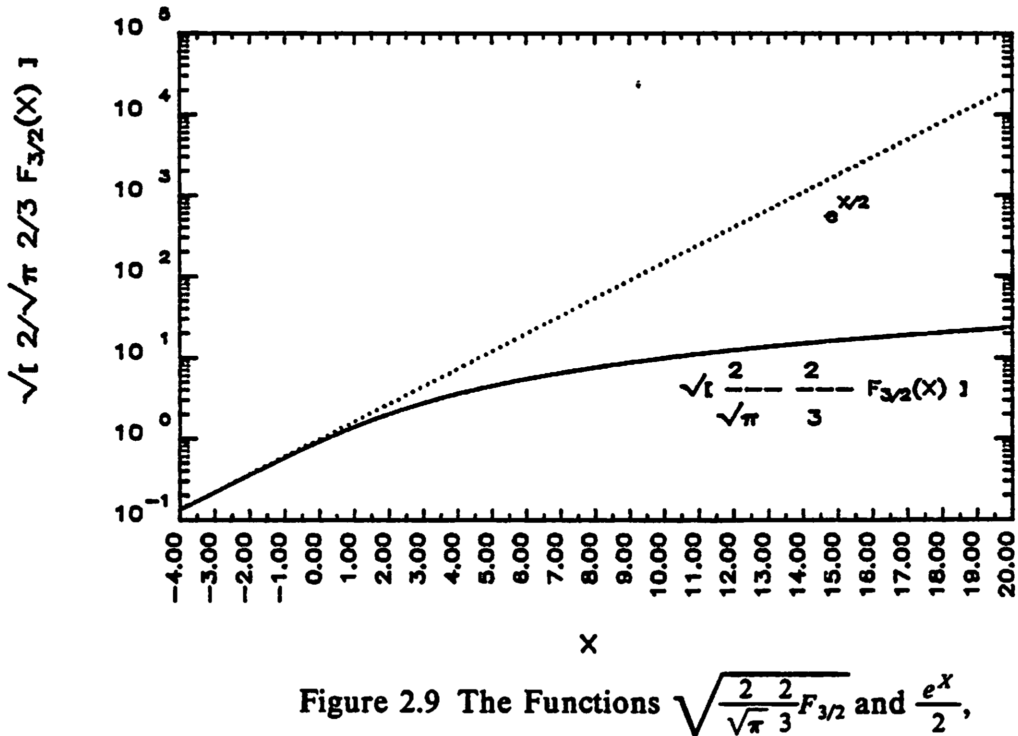


Figure 2.9 The Functions $\sqrt{\frac{2}{\sqrt{\pi}}\frac{2}{3}F_{3/2}}$ and $\frac{e^X}{2}$,

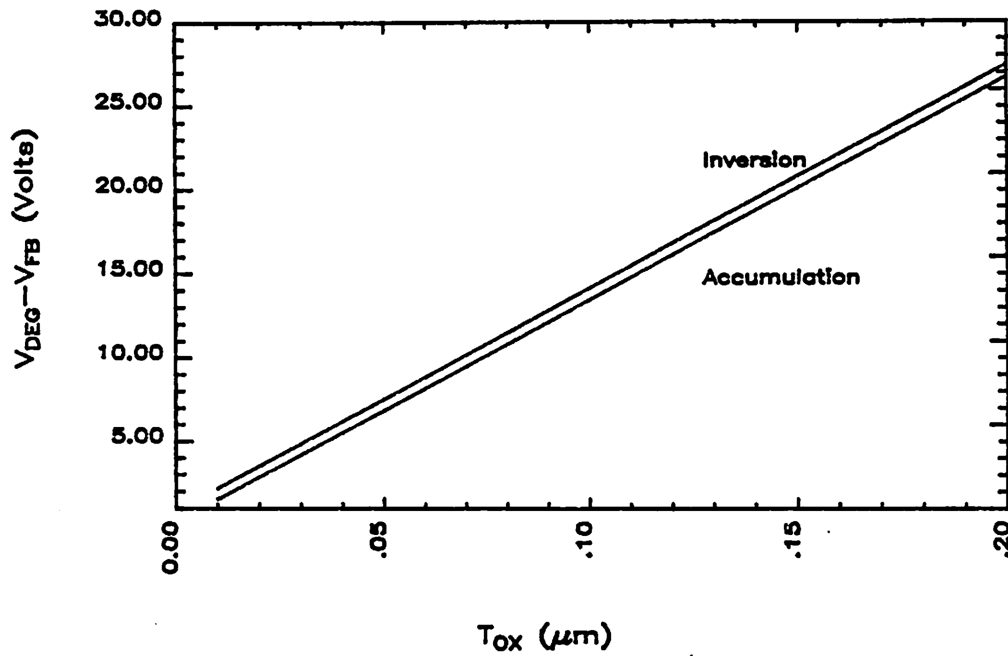


Figure 2.10The Onset Voltage of Degeneracy as Defined by the Coincidence of the Fermi Level with Band Edge,

the inverted and accumulated surfaces.

An accumulated surface, e.g. the surface of a depletion device in the strong-conduction mode, differs from that of an inverted one because there is no depletion charge next to the interface. The degeneracy voltage of an accumulated surface is lower than that of an inverted surface as shown in Figure 2.10.

2.2.3. The Impact on Device Characteristics

By neglecting the surface-scattering effect which modulates the surface mobility, the drain conductance can be expressed as a function proportional to the carrier density per unit area, Q_{INV}, as in Equation (2.30):

$$G_{DS} = -\frac{W}{L} U_{EFF} Q_{INV}(drain)$$

where

$$Q_{INV}(drain) = C_{OX}\left[V_{GB} - V_{FB} - \phi_s\right] - Q_{DEP}(drain) \qquad (2.40)$$

where V_{GS}, which is one order of magnitude larger than ϕ_s, dominates. Transconductance, G_M, can be expressed as:

$$G_M = \frac{W}{L} U_S \int_{\zeta_{n,S}}^{\zeta_{n,D}} \frac{\partial Q_{INV}}{\partial V_{GB}} d\zeta_n \qquad (2.41)$$

$$= \frac{W}{L} U_S \int_{\zeta_{n,S}}^{\zeta_{n,D}} \frac{N(\phi_s - \zeta_n)}{Field_{SURF}(\zeta_n) + \frac{\rho_{SURF}(\zeta_n)}{C_{OX}}} d\zeta_n \qquad (2.42)$$

It depends on the physical properties at the surface of the channel region. The impact of quantum-mechanical statistics on I_{DS}, G_{DS} and G_M is estimated by carrying out the related integrations in Equations (2.18), (2.30) and (2.42). The calculations in the degenerate condition is done by the Table-Look-Up method. The results show that, for the case of $T_{OX} = 0.01\mu m$,

$N_{SUB} = 5\times10^{16} cm^{-3}$, $V_{TO} = 0.5$, $V_{DS} = 0.5$ and $V_{GS} \leq 10$ volts, the quantum-mechanical impact on these device characteristics is limited within 1 percent deviation from the classical one. This finding agrees with Reference [35] upon the capacitive calculation.

A differentiable and integratable empirical expression is fitted into function $F = \sqrt{\frac{2}{\sqrt{\pi}} \frac{2}{3} F_{3/2}}$:

$$F = e^{\frac{X}{2}} \left[a_0 + a_1 X + a_2 X^2 \right] \quad (2.43)$$

where

$$a_0 = 0.924 \quad (2.44)$$

$$a_1 = -0.062 \quad (2.45)$$

$$a_2 = -0.012 \quad (2.46)$$

Its functional performance and percentage deviation are plotted in Figures 2.11 and 2.12.

A new relationship between the surface potential and the terminal voltages, which correctly reflects the impact of the quantum-mechanical effects, can be derived only by including the effects of the wave property and band quantization.

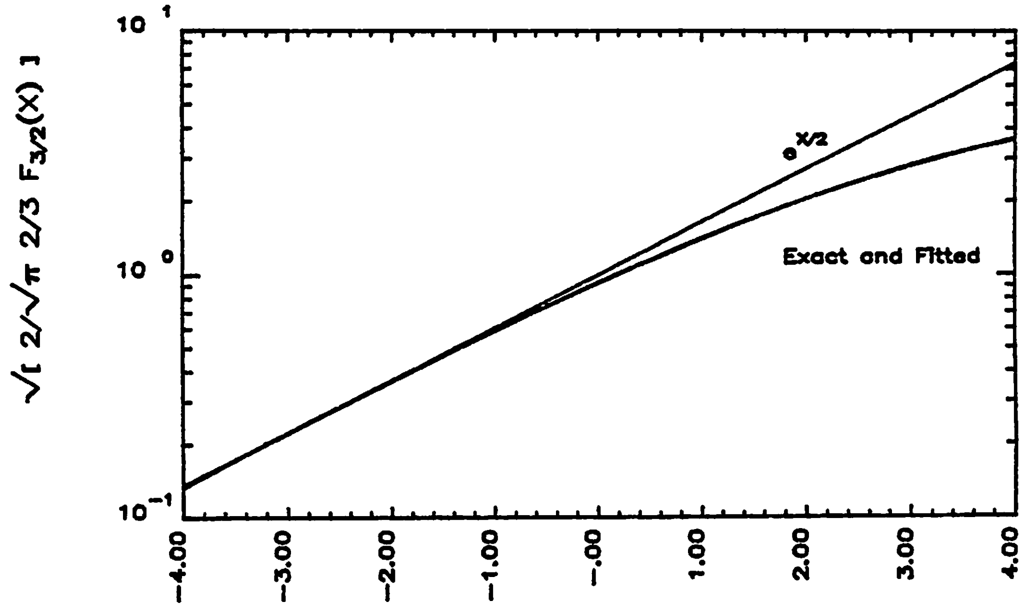


Figure 2.11The Exact and Approximate Functional Curves of $\sqrt{\frac{2}{\sqrt{\pi}}\frac{2}{3}F_{3/2}}$,

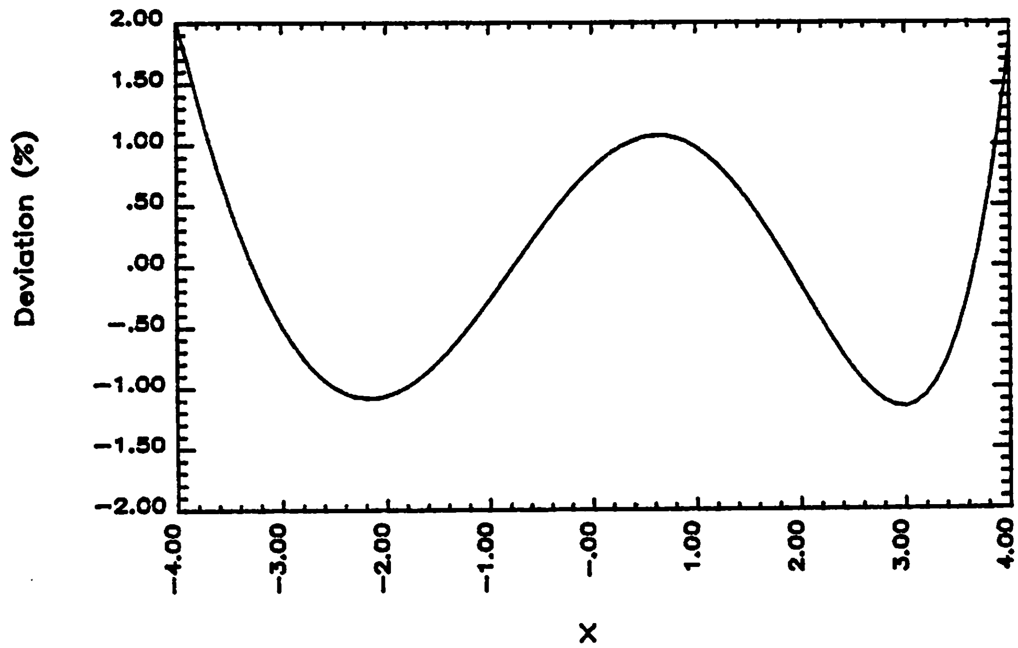


Figure 2.12The Percentage Deviation between the Exact and Approximate Expressions of Function $\sqrt{\frac{2}{\sqrt{\pi}}\frac{2}{3}F_{3/2}}$,

CHAPTER 3

A Unified Approach to MOSFET Modeling

For today's MOSFET applications, an adequate CAD model, besides addressing the small-geometry effects, should also be able to simulate consistently the operations of both strong and weak inversions. Much of the literature on MOSFET modeling concentrates only on device characteristics either below or above the threshold voltage [31,36]. Some unified theories and models [17,21,26] have been proposed which require internal iterations to achieve a solution for the characteristics. These models have slow computational speed. Other models [15] attempt to join the characteristics based upon different theories of specific operational ranges. If the physical implications are not correctly perceived, these models will be accurate in only one of the two regions.

This chapter presents a unified approach to MOSFET modeling without invoking internal computational iterations. The proposed approach is based upon the recognition of the transition region between weak and strong inversions. It establishes an efficient CAD model covering overall characteristics. Section 3.1 describes the inadequacy of existing unified models. Sections 3.2 and 3.3 present the strong- and the weak-inversion models upon which the proposed model of transition region is based. Section 3.4 gives the definition and establishes the significance of the transition region. Model equations are developed. Section 3.5 describes the influence of fast-surface states. A comparison with existing unified models demonstrates the validity of the new model for CAD applications.

3.1. General Overview

The drain-to-source conduction of a MOSFET is induced and modulated by the gate voltage. The drain current consists of both diffusion and drift components. The diffusion current dominates in strong inversion, while the drift current dominates in weak inversion. In the previous chapter, the general expression of drain current, Equation (2.18), including both components, has been derived in terms of an integral relationship with the quasi-Fermi level as the integrating parameter:

$$\begin{aligned} I_{DS} &= -\frac{W}{L} U_{EFF} \int_0^{V_{DS}} Q_{INV} d\zeta_n \\ &= -\frac{W}{L} U_{EFF} \int_0^{V_{DS}} \left[Q_{SI} - Q_{DEP} \right] d\zeta_n \end{aligned} \quad (3.1)$$

In order to carry out the integration, the density of the total charge inside the semiconductor, Q_{SI}, has to be partitioned into the inversion charge Q_{INV} and the fixed surface depletion charge Q_{DEP}, and related to the terminal voltages.

The total semiconductor charge is balanced by the charge residing on the gate, and can be related to the terminal voltages, as derived in the previous chapter, by a manipulation of the Poisson equation, as in Equation (2.22):

$$\frac{1}{2}\left[\frac{Q_{SI}}{\epsilon_{si}}\right]^2 = \frac{qN_{SUB}}{\epsilon_{si}} Q_o \quad (3.2)$$

where

$$Q_o = \left[1 - e^{-\frac{2q\phi_F}{kT}}\right]\phi_s + \frac{kT}{q} e^{-\frac{q(2\phi_F + \zeta_n)}{kT}}\left[e^{\frac{q\phi_s}{kT}} - 1\right] + \frac{kT}{q}\left[e^{-\frac{q\phi_s}{kT}} - 1\right] \quad (3.3)$$

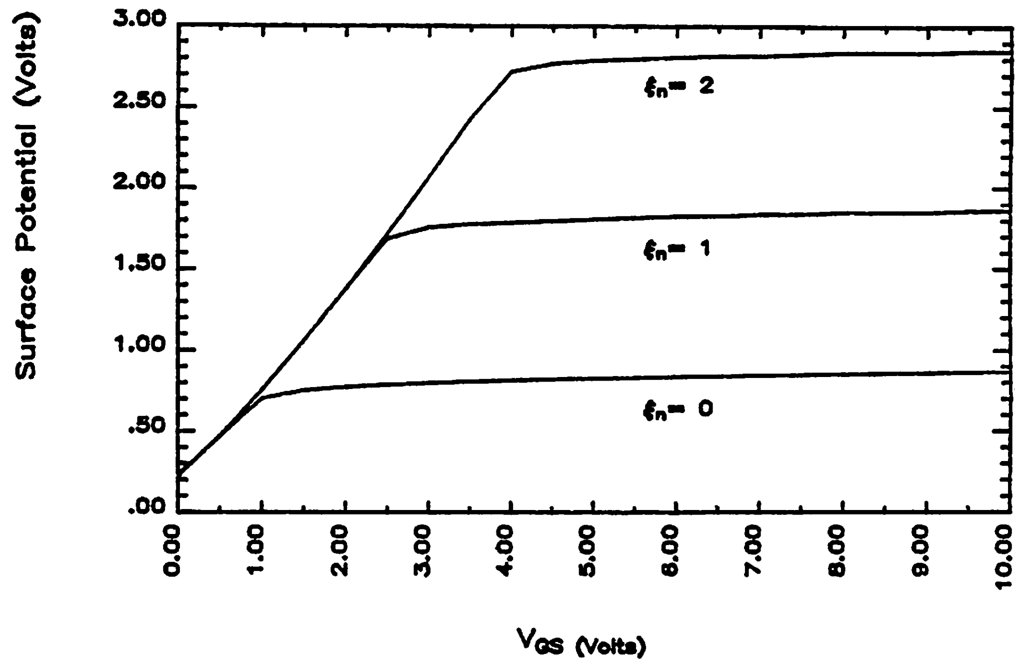
The difference between Equations (2.22) and (3.2) is in the inclusion of terms corresponding to the concentration of holes and depleted donors.

The diffusion current is proportional to the gradient of carrier density, while the drift current is proportional to the product of the carrier density and the electric field strength. In other words, these two current components are controlled by either the gradient or the magnitude of the surface potential, ϕ_s. Existing unified theories generally formulate the characteristic equations in terms of the surface potential which is a controlling variable in Equation (3.2).

The relationship between ϕ_s and the terminal voltages as predicted by Equation (3.2) is plotted in Figure 3.1 in which the ϕ_s-versus-V_{GS} curves are plotted with parameters ζ_n and V_{BS}. The curves asymptotically approach straight lines in the extreme of either strong or weak inversion. In these two extremes, the transcendental Equation (3.2) can be approximated by explicit analytical equations which can be used to develop the models of both regions.

Equation (3.2) is used by all the unified theories to relate the quasi-Fermi level to terminal voltages [17,21,26]. The approaches differ in the way they divide Q_{SI} into Q_{INV} and Q_{DEP}, thus leading to different results. In general, Equation (3.2) has to be solved by iteration, especially in the weak-inversion region where ϕ_s is almost constant over the entire integration domain of Equation (3.1). An accurate value of ϕ_s is required because the drain current is calculated as the difference between two very close quantities. Therefore, most unified theories require a well-converged solution of Equation (3.2) for the weak-inversion characteristics.

The unified model which is developed by Baccarani et al. [17], together with the model based upon the double integration [26], the conventional strong-inversion model [31] and the weak-inversion model by Swanson et al. [36] have been programmed on a Hewlett-Packard 1000 minicomputer to compare the computational speed (in second per operation), in linear,



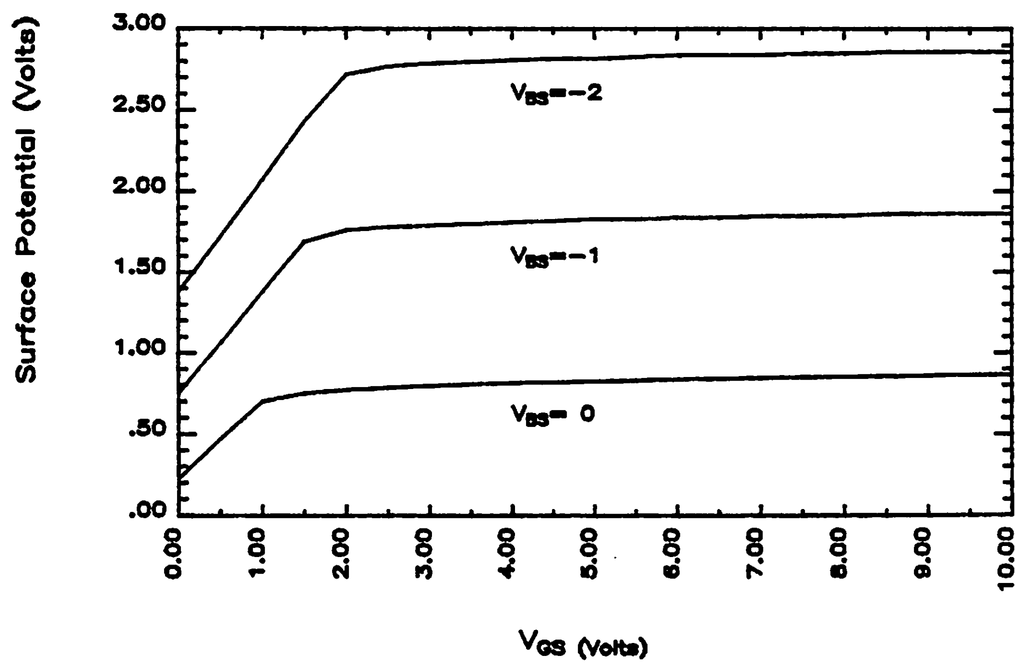


Figure 3.1 ϕ_s-versus-V_{GB} with Parameters ζ_n and V_{BS},

saturation, and weak-inversion regions. The results are summarized in Table 3.1:

Table 3.1			
Model	Linear	Saturation	Weak-Inv
P-S	5.23 sec/op	5.09 sec/op	5.22 sec/op
B-R-S	5.08 msec /op	3.03 msec/op	3.64 msec/op
Strong	0.85 msec/op	1.06 msec/op	-
Weak	-	-	0.68 msec/op

Table 3.1 Comparison of Computational Speeds

The computational speed of the iterative approach is three orders of magnitude greater than that of the double-integration approach, while that of the explicit approaches is five times as fast as the iterative approach. As the model evaluation consumes a very high percentage of computational time in VLSI circuit simulations, the explicit approach is much more desirable than the others.

The Pao-Sah theory [26] uses a rudimentary integral formulation,

$$I_{DS} = \frac{W}{L} U_{EFF} q \int_0^{V_{DS}} d\zeta_n \int_0^{\phi_s(\zeta_n)} \frac{N(\phi)}{E(\phi)} d\phi \tag{3.4}$$

where approximations are kept to a minimum. However, the result is cumbersome for CAD applications. The El-Mansy-Boothroyd theory [21] partitions the charge based upon overall weighting factors:

$$I_{DS} = \frac{W}{L} U_{EFF} \int_0^{V_{DS}} d\zeta_n \sigma_{INV} Q_{INV} \tag{3.5}$$

Although allowing for the influence of the spatial distribution, this treatment gives an unrealistic result because of using the weighted average. The Baccarani-Rudan-Spadini theory [17] decouples Q_{DEP} from Q_{INV} by relating Q_{DEP} to ϕ_s by a square-root relationship. It is equivalent to assuming that the inversion charge distribution has no influence on the potential distribution.

$$I_{DS} = \frac{W}{L} U_{EFF} C_{OX} \int_0^{V_{DS}} \left[V_{GB} - V_{FB} - \phi_s(\zeta_n) - \gamma\sqrt{\phi_s(\zeta_n)} \right] d\zeta_n \quad (3.6)$$

This result agrees well with the Pao-Sah theory, especially in the weak-inversion region where the depletion charge dominates.

The other method of providing a model covering wide operational ranges is to attach exponential tails with measured weak-inversion slopes to the strong-inversion characteristics at a point slightly above the threshold voltage [15].

The threshold voltage used in model evaluation can be determined by:

(a) plotting $(\frac{I_{DS}}{V_{DS}})$-versus-V_{GS}, measured at fixed low V_{DS} and fixed V_{BS},

(b) extrapolating the linear portion of the curve to zero,

(c) interpreting the interception on the V_{GS} axis, V_{GS}^o, as a point corresponding to pseudo zero inversion charge density,

(d) relating V_{TH} to V_{GS}^o through the conventional current equation in a rearranged format:

$$V_{TH} = V_{GS}^o - \frac{V_{DS}}{2} - \frac{L \times I_{DS}}{W \times V_{DS} U_{EFF} C_{OX}} \quad (3.7)$$

The other parameters, N_{SUB}, ϕ_F, and γ, can be derived from the extrapolated V_{TH}'s at different V_{BS}'s:

$$N_{SUB} = \frac{\left[\gamma C_{OX}\right]^2}{2q\epsilon_{si}} \quad (3.8)$$

$$\phi_F = \frac{kT}{q} \ln\left[\frac{N_{SUB}}{N_I}\right] \quad (3.9)$$

$$\gamma = \frac{V_{TH}(V_{BS2}) - V_{TH}(V_{BS1})}{\sqrt{\phi_F - V_{BS2}} - \sqrt{\phi_F - V_{BS1}}} \tag{3.10}$$

Iterative computation is required to obtain a set of consistent parameters.

In order to compare the basic assumptions of different modeling approaches, the Pao-Sah theory is used to generate ideal characteristics; these characteristics are used to extract model parameters for the other models. The physical parameters used in the Pao-Sah model are listed in Table 3.2 together with the extrapolated parameters to be used for the other models:

Table 3.2

Dev	Mdl -	NSUB (cm-3)	TOX (um)	GAMMA (sqrt V)	VTO (V)	VFB (V)
No.1	Intr.	5.0E14	0.105	0.39	0.26	-0.57
	Extr.	3.1E14	0.105	0.36	0.41	-0.38
No.2	Intr.	5.0E15	0.105	1.24	0.26	-1.41
	Extr.	3.5E15	0.105	1.17	0.41	-1.19
No.3	Intr.	2.0E16	0.105	2.48	0.26	-2.95
	Extr.	1.9E16	0.105	2.41	0.42	-2.37
No.4	Intr.	5.0E15	0.05	0.59	0.26	-0.88
	Extr.	3.2E15	0.05	0.54	0.41	-0.68
No.5	Intr.	5.0E15	0.20	2.36	0.26	-2.32
	Extr.	3.7E15	0.20	2.30	0.42	-2.09

Table 3.2 Intrinsic and Extrapolated Parameters

The extrapolated parameter values differ from the intrinsic ones. The most significant difference is between the value of V_{TH} and the value of V_{FB}. Though the values differ by only about $0.2V$, the difference implies that the extrapolated V_{TH} does not correspond to a point at which ϕ_s equals $2\phi_F$ as generally assumed. If the same parameters are used in the companion weak-inversion tail of the joint model with a single break point, the quality of the weak-inversion model is sacrificed.

The ideal characteristics with intrinsic parameters based upon the Pao-Sah theory [26], the simple strong-inversion model [31] with extrapolated parameters, the simple weak-inversion model [36] with intrinsic parameters, and the strong-weak-inversion model with single break point [15] with extrapolated parameters are plotted together in Figure 3.2 in which V_{DS} equals 0.05 V and 5.0 V. The method of choosing a single break point fails to recognize the existence of a transition region where none of the specialized theories apply. It results in an erroneous prediction of the current in weak inversion if it is matched in strong inversion, and vice versa.

All the weak-inversion models apply to the weak-inversion region only, and all the strong-inversion models apply to the strong-inversion region only. A single break point approach is not sufficient to retain the accuracy in both original theories.

3.2. Strong-Inversion Region

Strong inversion is characterized by either a high V_{GS} or a low quasi-Fermi level. At this extreme, the right hand side of Equation (3.2) is dominated by the term of inversion charge and ϕ_s can be approximated as:

$$\phi_s = 2\phi_F + \zeta_n + \frac{kT}{q} \ln\left[\frac{q}{kT}\left[\left[\frac{V_{GB} - \phi_s}{\gamma}\right]^2 - \phi_s\right]\right] \tag{3.11}$$

$$= \phi_{BI} + \zeta_n$$

where

$$\phi_{BI} = 2\phi_F + \frac{kT}{q} \ln\left[\frac{q}{kT}\left[\left[\frac{V_{GB} - \phi_s}{\gamma}\right]^2 - \phi_s\right]\right] \tag{3.12}$$

ϕ_s is linearly related to the quasi-Fermi level ζ_n and logarithmically related to V_{GB}. Its dependence on V_{GB} is small and ϕ_{BI} stays close to $2.5\phi_F$ in the

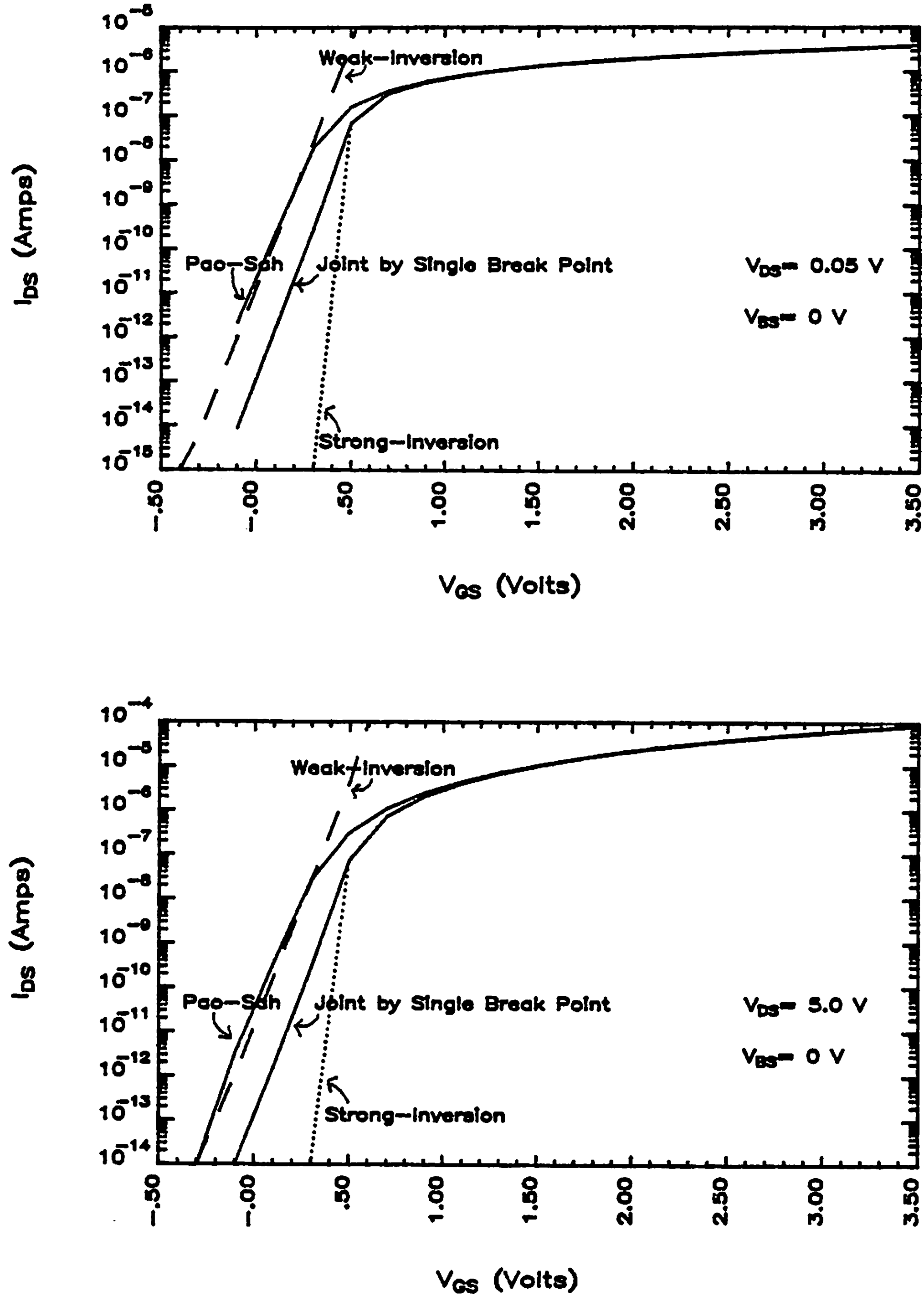


Figure 3.2 The Pao-Sah Model and the Approximations in Strong- and Weak-Inversion Regions at V_{DS} 0.05 V and 5.0 V,

strong-inversion region. Therefore the ϕ_s used inside the logarithm can be approximated by $(2.5\phi_F + \zeta_n)$ without losing accuracy.

As pointed out in the previous chapter, when V_{GB} is sufficiently high, Equation (3.2) no longer holds because the Boltzmann statistics can not apply to the degenerate case. A new equation is proposed in the previous chapter to replace Equation (3.2) in order to include quantum-mechanical-statistical effects. The break point joining the non-degenerate and degenerate cases is defined as follows:

$$V_{DEG} = V_{FB} + \phi_F + \frac{E_G}{2q} + \zeta_n \tag{3.13}$$

As concluded in the previous chapter, the correction factor due to degenerate statistics is small and it is important only at a very high V_{GS}. In most of the practical operational range, the device characteristics are subject to the strong influence of surface-mobility modulation. The degeneracy impact of sub-band splitting can be included in empirical equations designed for mobility modulation. Thus this effect is neglected in the following derivation of model equations.

The terms are plotted in Figure 3.3 for comparison. ϕ_s and the contribution of the depletion charge are approximately constant in the strong-inversion region. Q_{DEP} can be approximated as:

$$Q_{DEP} = \gamma C_{OX}\sqrt{\phi_s} \tag{3.14}$$

The current equation can be derived accordingly:

$$I_{DS} = \frac{W}{L} U_{EFF} C_{OX} I^o \tag{3.15}$$

where

$$I^o = \left[V_{GB} - V_{FB} - \frac{\phi_{s,D} + \phi_{s,S}}{2}\right]\left[\phi_{s,D} - \phi_{s,S}\right] - \frac{2}{3}\gamma\left[\phi_{s,D}^{\frac{3}{2}} - \phi_{s,S}^{\frac{3}{2}}\right] \tag{3.16}$$

This formulation is very similar to the conventional one [31] and the difference is minor.

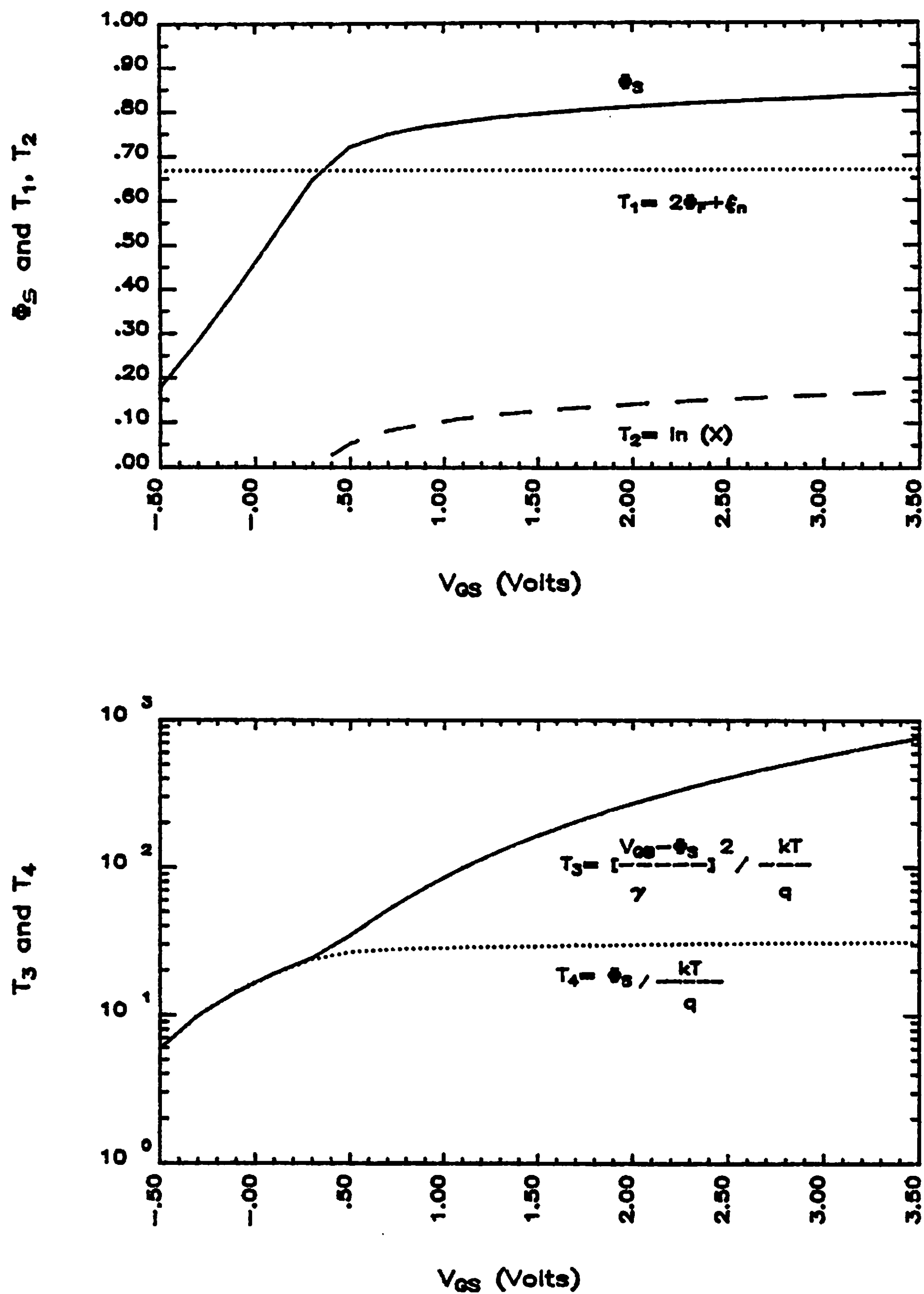


Figure 3.3 The Contributions to ϕ_s from Various Terms: T_1, T_2, T_3, and T_4,

3.3. Weak-Inversion Region

When V_{GB} is low and/or the quasi-Fermi level, ζ_n, is high, ϕ_s is approximately linearly proportional to V_{GB} and very insensitive to the quasi-Fermi level, as illustrated in Figure 3.1. In Equation (3.2), the term corresponding to the minority carrier concentration in weak inversion, which depends on the quasi-Fermi level, is so small that it can be ignored. When ϕ_s is smaller than $(2\phi_F + \zeta_n)$ and the term of the minority carrier concentration is neglected in Equation (3.2), the expression ϕ_s becomes:

$$\phi_s = \frac{\gamma^2}{2} + V_{GB} + \gamma\sqrt{\frac{\gamma^2}{4} + V_{GB}} \tag{3.17}$$

This approximation fails at $\phi_s \geq (2\phi_F + \zeta_n)$ where the exponent of the minority carrier term in Equation (3.2) becomes positive and the magnitude of the term increases rapidly.

In the weak-inversion region, the depletion charge is dominant and can be expressed as:

$$Q_{DEP} = -\gamma C_{OX}\sqrt{\phi_s - \frac{kT}{q}} \tag{3.18}$$

and the total charge density inside the semiconductor can be approximated as:

$$Q_{SI} = \gamma C_{OX}\sqrt{\phi_s - \frac{kT}{q}}\left[1 + \frac{1}{2}\frac{kT/q}{\phi_s - kT/q}e^{\frac{q(\phi_s - \zeta_n - 2\phi_F)}{kT}}\right] \tag{3.19}$$

The density of the inversion charge equals the difference between the magnitudes of Q_{SI} and Q_{DEP}:

$$Q_{INV} = Q_{SI} - Q_{DEP} \tag{3.20}$$

$$= C_D\frac{kT}{q}e^{\frac{q(\phi_s - \zeta_n - 2\phi_F)}{kT}} \tag{3.21}$$

where

$$C_D = \sqrt{\frac{q\,\epsilon_{si} N_{SUB}}{2\left[\phi_s - kT/q\right]}} \tag{3.22}$$

The drain current is dominated by diffusion. The configuration of the channel region of a MOSFET in weak inversion and the base region of a bipolar transistor is similar. Accordingly, the weak-inversion current can be formulated as:

$$I_{DS} = WD\frac{Q_{SRC} - Q_{DRN}}{L} \tag{3.23}$$

$$= \frac{W}{L} U_S C_D \frac{kT}{q}^2 e^{\frac{q(\phi_s - 2\phi_F - V_{SB})}{kT}} \left[1 - e^{-\frac{qV_{DS}}{kT}}\right] \tag{3.24}$$

3.4. Join Together

The weak-inversion model is accurate in the region where ϕ_s is equal to or less than $(2\phi_F + \zeta_n)$, while the strong-inversion model is good in the region where the inversion charge is the dominant charge component. By properly defining the transition region where neither of these two models is valid, the characteristics in the weak- and strong-inversion regions are joined together through an empirical transition characteristics to provide an accurate and efficient CAD model over the overall operational range.

The approximated ϕ_s of both strong and weak inversions is plotted in Figure 3.4, together with the exact solution, over the range of $(V_{GB} - V_{2\phi_F})$ between -1 and +4 volts. $V_{2\phi_F}$ is the gate voltage at which the surface potential equals $(2\phi_F + \zeta_n)$. The boundaries of the transition region can be defined as follows:

(a) weak-inversion boundary: $Vweak$ where surface potential $\phi_{s,WEAK} = (2\phi_F + \zeta_n)$; This is equivalent to the condition $\frac{\partial Q_{INV}}{\partial \phi_s} = \frac{\partial Q_{DEP}}{\partial \phi_s}$

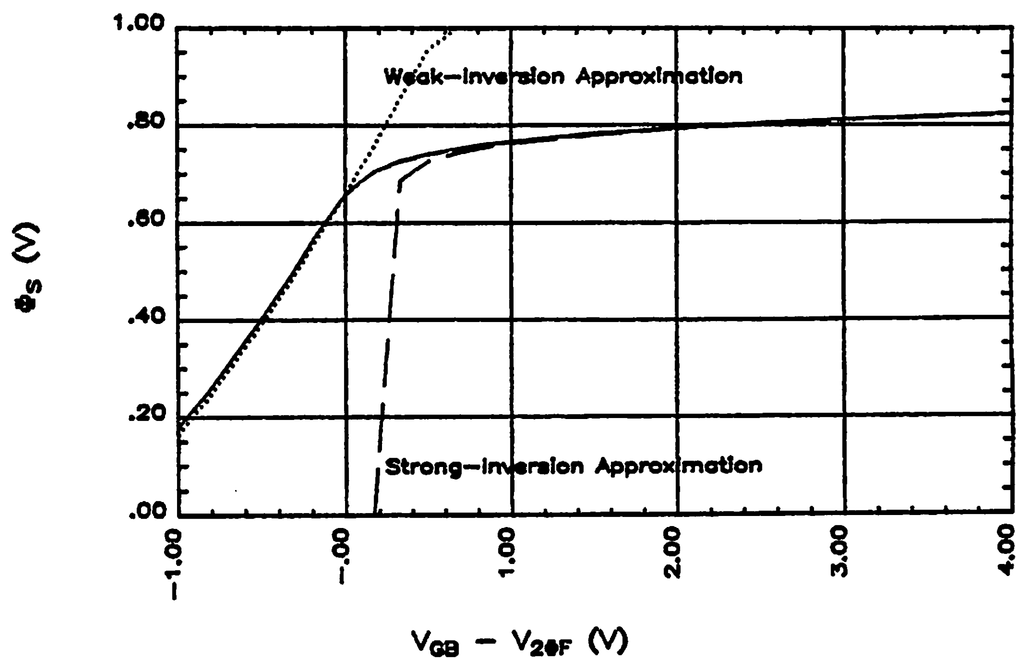


Figure 3.4 The Approximate and Exact ϕ_s in Strong- and Weak-Inversion Regions,

[37]. In other words, in this condition, a variation of ϕ_s induces the same amount of both the inversion and depletion charges. As indicated in the previous section, once ϕ_s is greater than $(2\phi_F + \zeta_n)$, the inversion charge increases exponentially and the depletion charge is no longer the dominant factor in determining the surface potential.

(b) strong-inversion boundary: V_{strong} where the surface potential is $\phi_{s,STRONG}$ so that the contribution to ϕ_s from Q_{INV}, $\phi_{s,INV}$ is ten times greater than that from Q_{DEP}, $\phi_{s,DEP}$. $\phi_{s,INV}$ and $\phi_{s,DEP}$ are the first and third terms inside the square root at the right side of Equation (3.2):

$$\phi_{s,INV} = \frac{kT}{q} e^{\frac{q(\phi_s - 2\phi_F - \zeta_n)}{kT}} \quad (3.25)$$

$$\phi_{s,DEP} = \phi_s \quad (3.26)$$

By assigning $X = \frac{\phi_{s,INV}}{\phi_{s,DEP}}$, it is reduced to:

$$\phi_{s,STRONG} = 2\phi_F + V_{SB} + \frac{kT}{q}\ln(X\frac{q\,\phi_{s,STRONG}}{kT}) \quad (3.27)$$

The corresponding gate voltage, V_{STRONG}, is:

$$V_{STRONG} = V_{FB} + \phi_{s,STRONG} + \gamma\sqrt{(1 + X)\phi_{s,STRONG}} \quad (3.28)$$

$\phi_{s,STRONG}$ can be estimated by replacing the $\phi_{s,STRONG}$ inside the logarithm by $(2\phi_F + \zeta_n)$.

Inside the transition region, the sensitivity of ϕ_s-versus-V_{GB}, $(\partial\phi_s/\partial V_{GS})(V_{GB} - V_{FB})/\phi_s$, changes from greater than one percent to about ten percent. The following expression is proposed to approximate the relationship of ϕ_s to V_{GS} in the transition region:

$$\phi_s = a + b(V_{GB} - V_{WEAK})^n \quad (3.29)$$

Once the exponent n, the sensitivity coefficient, is determined, the coefficients a and b can be deduced from the boundary conditions. n can best be approximated as $\frac{1}{2}$ to reflect the average sensitivity within the transi-

tion region. For cases of $5.0\times10^{14}\leq N_{SUB}\leq5.0\times10^{16}$ and $0.05\mu m\leq T_{OX}\leq0.2\mu m$, the difference between V_{WEAK} and V_{STRONG} is approximately $0.6V$. The resultant ϕ_s together with the exact solution and the percentage deviation are plotted in Figure 3.5 for comparison. The deviation between the approximation and the exact solution is less than 10 percent.

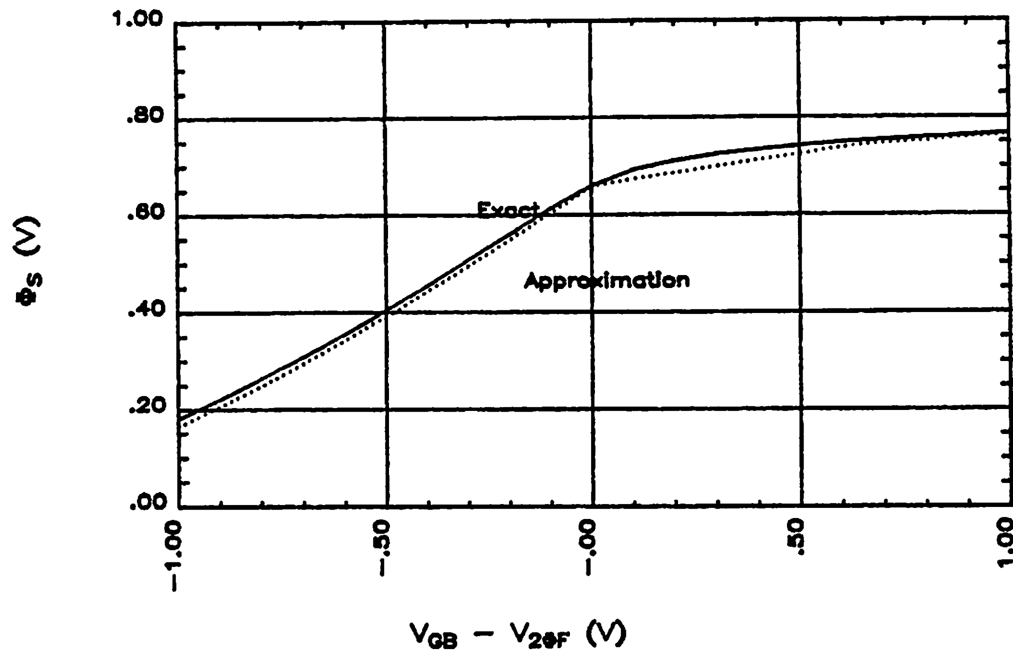
The relationship of V_{GB} to the logarithm of drain currents is very similar to that of ϕ_s-versus-V_{GB}. The formulation of the drain current in the transition region is complicated by the dependence of ϕ_s on the quasi-Fermi level ζ_n which is the integrating variable in the current equation. Instead of dividing the integral into three parts, which corresponds to partitioning the channel into strong-, weak-inversion and transition regions, the weak- and strong-inversion characteristics are joined through the same transition region by a similar equation. The comparisons between the current-voltage characteristics based upon different approaches, at $V_{DS} = 0.05V$ and $V_{BS} = 0V$, are plotted in Figure 3.6 with parameters N_{SUB} and T_{OX}. The transition region is about the same as the one in the relationship between ϕ_s-versus-V_{GB}. The drain current in the transition region is expressed in terms of the currents at the break points.

$$I_{DS} = e^{\ln(I_{WEAK}) + b(V_{GB} - V_{WEAK})^{\frac{1}{2}}} \qquad (3.30)$$

where

$$b = \frac{\ln(I_{STRONG}) - \ln(I_{WEAK})}{(V_{STRONG} - V_{WEAK})^{\frac{1}{2}}} \qquad (3.31)$$

Compared with the exact ideal characteristics in Figure 3.7, the overall agreement between the exact and joint solutions is good. The computational speed of this model is as fast as that of simple models in both weak- and strong-inversion regions. Only in the transition region is there a penalty in computational speed.



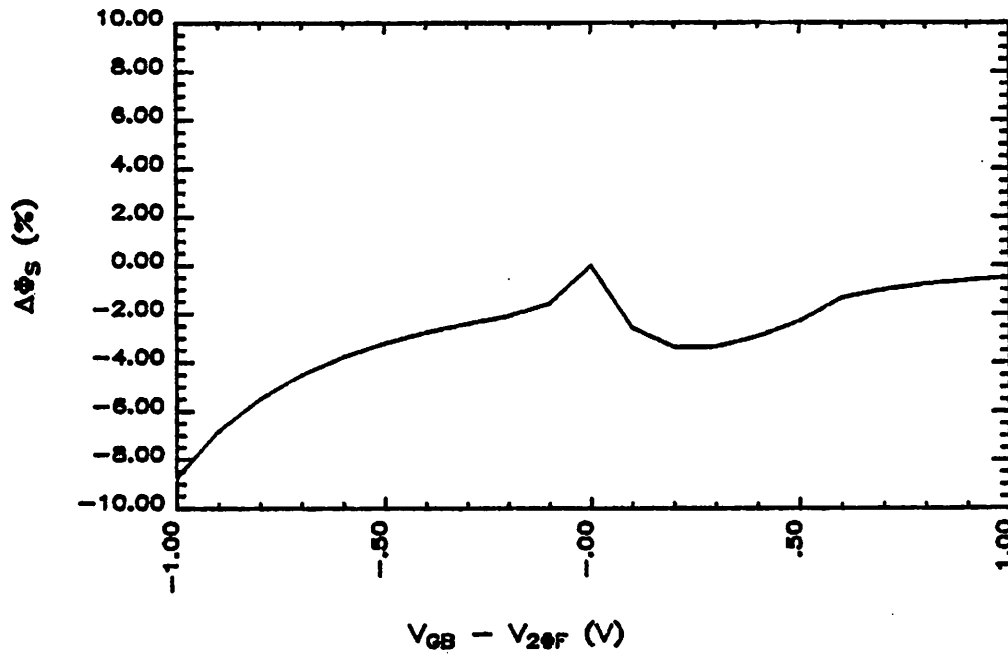
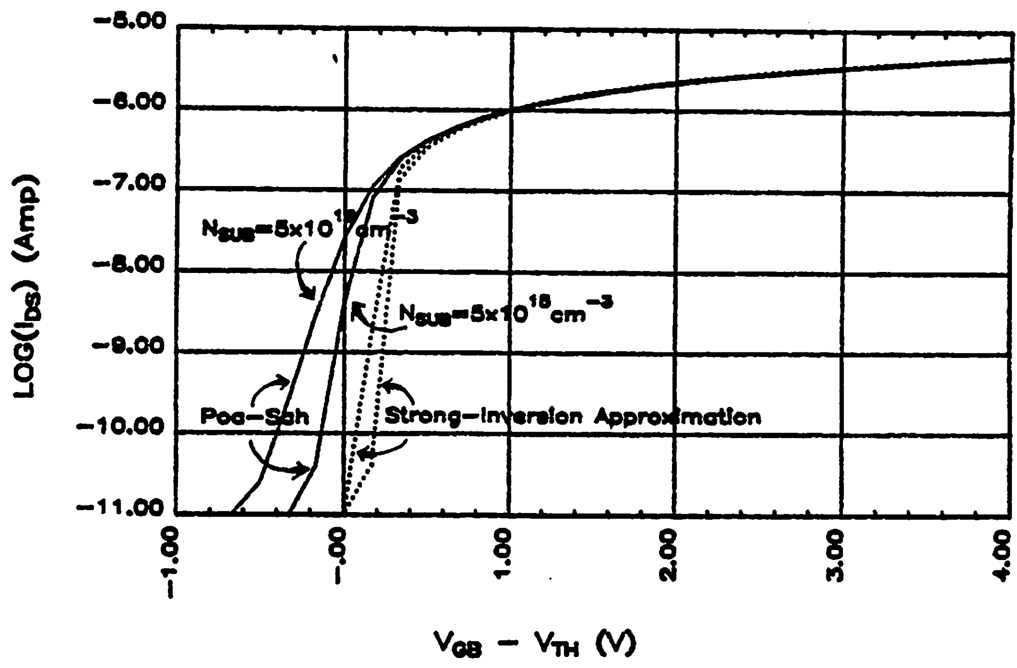


Figure 3.5 The Exact and Joint Curvers of ϕ_s-versus-V_{GB} and the Percentage Deviation,



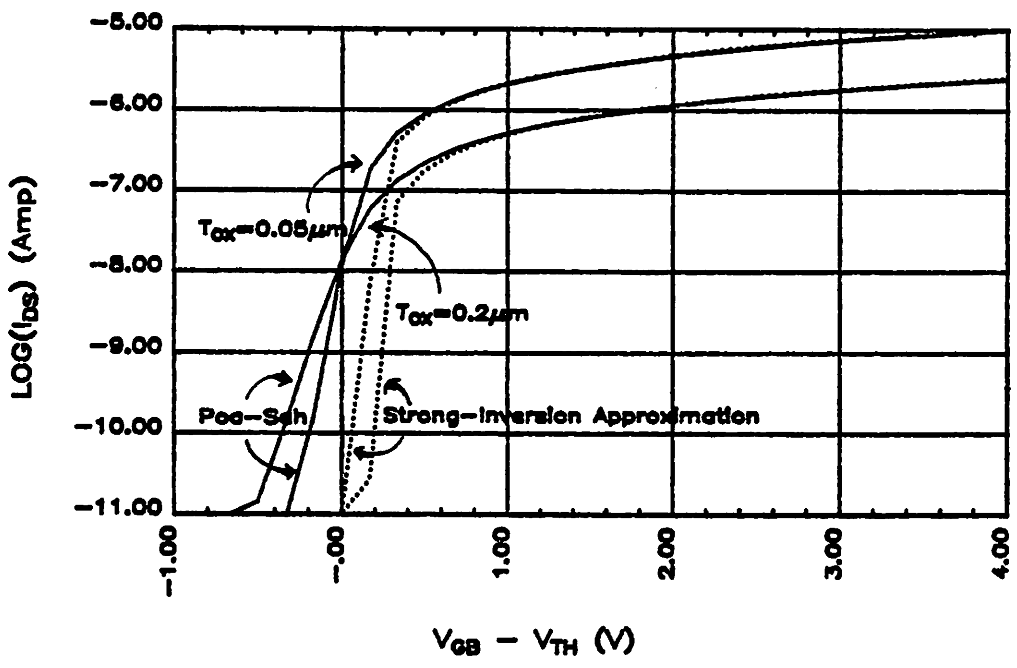


Figure 3.6 The Exact and Approximate Curves of I_{DS}-versus-V_{GS} in Strong-inversion Regions with Parameters N_{SUB} and T_{OX},

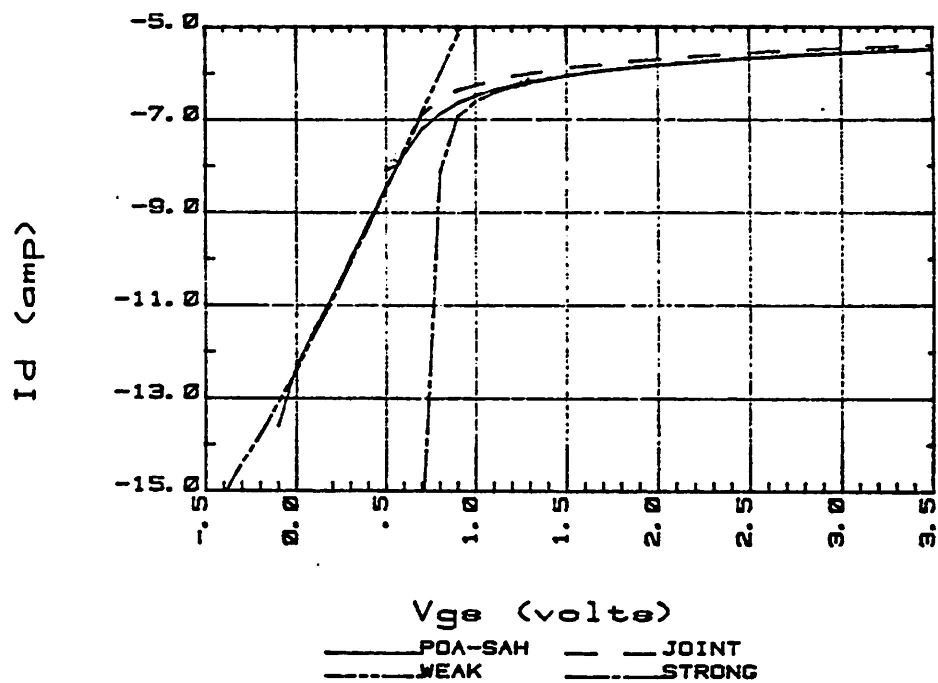


Figure 3.7 The Exact and Joint Curves of I_{DS}-versus-V_{GB},

3.5. The Influence of Fast-Surface States

The fast-surface-state density is used by van Overstraeton et al. [38] and Swanson et al. [36] as a parameter to characterize the weak-inversion region. Fast-surface states are the surface states whose lifetime is so short as to be filled and/or emptied fast enough to follow the variation in ϕ_s induced by changes in the applied biases. The fast-surface states are induced by the broken bonds at the surface resulting from the interruption of the crystalline structure [39]. They distribute themselves almost uniformly over the center of the energy gap, with the peak densities near the band edges. The detailed distribution function differs from material to material. Experiments [40] show that the density of fast-surface states is about 10^{10} to 10^{11} $cm^{-2}eV^{-1}$ in the central region.

In the presence of fast-surface states, the Q_{SI} in the left side of Equation (3.2) becomes:

$$Q_{SI} = C_{OX}\left[V_{GB} - V_{FB} - \phi_s + \frac{qN_{FS}}{C_{OX}}\left[\phi_s - \zeta_n\right]\right] \tag{3.32}$$

Figure 3.8 shows how the relationships of ϕ_s-versus-V_{GB} and I_{DS}-versus-V_{GB} vary with the fast-surface-state density. Both relationships are based upon the Pao-Sah theory with Q_{SI} defined by Equation (3.32).

The presence of fast-surface states widens the transition region and lowers the current. This situation is equivalent to lower the threshold voltage. If the fast-surface-state density is abnormally high, say 10^{12}, the turn-on characteristics are softened and the weak-inversion slope is reduced. Fast-surface states have a significant effect only in the weak-inversion region. The approximated ϕ_s in the weak-inversion region is updated to include the effect of fast-surface states:

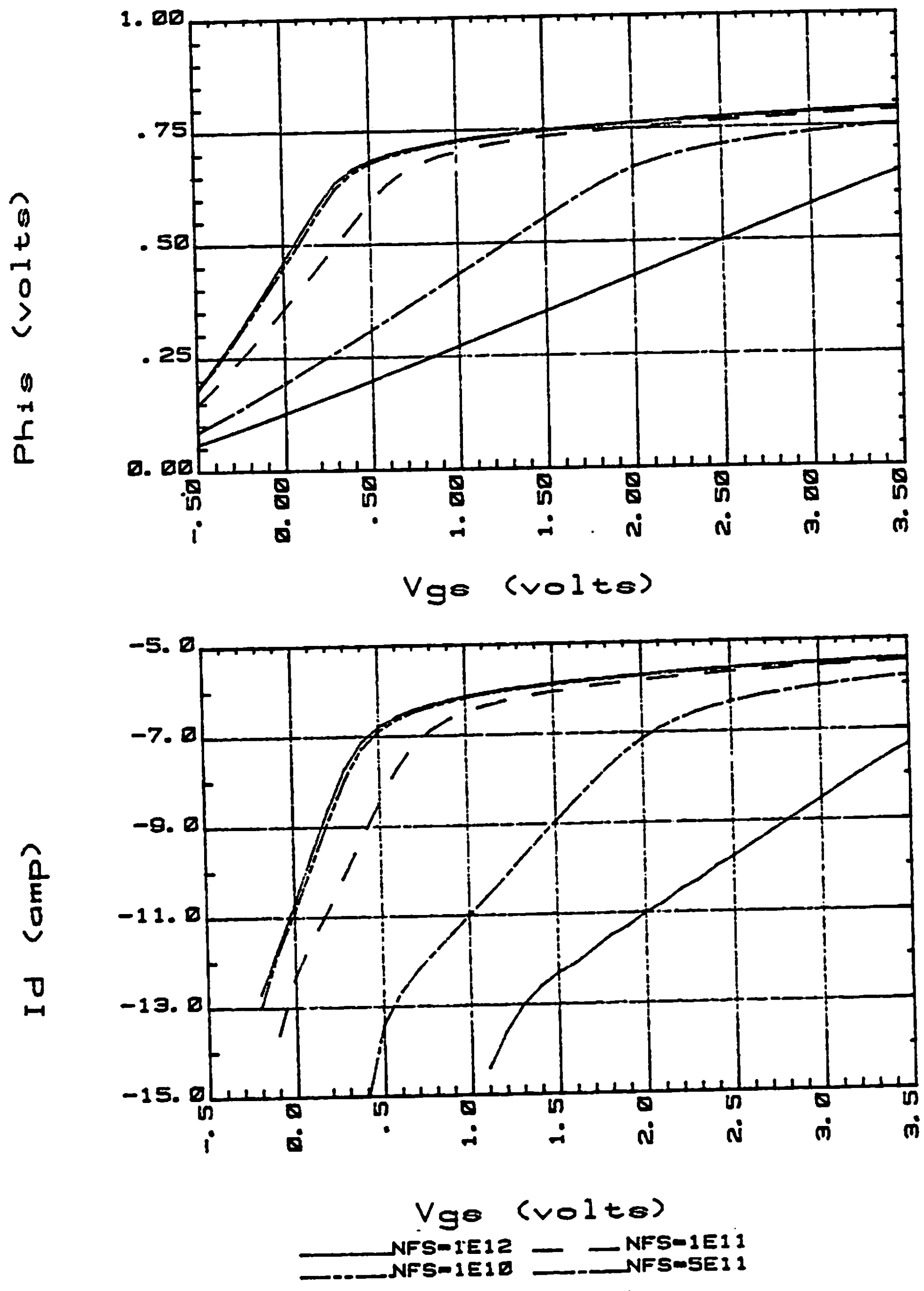


Figure 3.8 The Curves of ϕ_s-versus-V_{GB} and I_{DS}-versus-V_{GB} with Parameter N_{FS},

$$\phi_s = \frac{V_{GB} - V_{FB}}{1 + \alpha} + \frac{1}{2}\left[\frac{\gamma}{1 + \alpha}\right]^2 - \frac{\gamma}{1 + \alpha}\sqrt{\frac{1}{4}\left(\frac{\gamma}{1 + \alpha}\right)^2 + \frac{V_{GB} - V_{FB}}{1 + \alpha}} \qquad (3.33)$$

where

$$\alpha = \frac{qN_{FS}}{C_{OX}} \qquad (3.34)$$

The current equation in weak inversion is modified accordingly. The weak-inversion slope, $\frac{kT}{q}\frac{\partial \ln(I_{DS})}{\partial V_{GB}}$, becomes:

$$\frac{kT}{q}\frac{\partial \ln(I_{DS})}{\partial V_{GB}} = \frac{1}{1 + \alpha}\left[1 - \frac{1}{\sqrt{1 + 4\left(V_{GB} - V_{FB}\right)\frac{1 + \alpha}{\gamma^2}}}\right] \qquad (3.35)$$

The emphasis in this chapter is placed upon the transition between the strong- and weak-inversion regions, i.e. the region close to the threshold voltage. The device characteristics are better described by the flatband voltage, V_{FB}, than by the threshold voltage, V_{TH}. Once the device operates in the strong-inversion region, the effect of surface-mobility modulation, which is not covered in this chapter, can not be ignored and will be presented in Chapter 6.

CHAPTER 4

Two-Dimensional Simulations of Small-Geometry MOSFETs

With decreasing transistor dimensions, it has become more difficult to describe MOS transistors with equations that are simple enough for hand calculations or programmable calculators and yet retain sufficient accuracy to provide useful information about the device characteristics. In a small-geometry MOSFET both the impurity and potential distributions are extremely inhomogeneous. The numerical solution of two-dimensional potential and current-continuity equations is required to determine their characteristics. A thorough solution including every possible effect can be obtained using maxi-computers.

However, a more limited computer program with interactive capacity is also needed. Such a program, if sufficiently fast and efficient, can interactively provide iterative solutions which can then be used to obtain the optimal device structure. This requirement limits the solutions to that of the two-dimensional impurity and potential distributions, in other words, to the handling of the weak-inversion or weak-injection approximation for an MOS transistor. Since VLSI devices are geared toward low-voltage and low-power applications, these characteristics are of critical importance.

Program TWIST (TWo-dimensional Interactive Simulation of MOS Transistors) has been developed using a minicomputer together with graphics terminals to simulate the characteristics of weak inversion and weak-injection punchthrough by the solution of the two-dimensional Poisson equation. Graded mesh, modified Gummel's algorithm, and Successive-Over-Relaxation iteration, together with a by-pass scheme, are implemented. The desired high-speed interactive feature and the graphics representation of all

data are demonstrated. Program TWIST is used in later chapters to study the weak inversion and weak-injection punchthrough characteristics.

The use of this program allows optimized device structures to be developed which then deserve more elaborate simulations involving the complete solution of both the potential and the transport aspects, which presently consumes approximately 50 times more computational time than the approach presented here. In a working hierarchy of CAD tools, structural and impurity parameters can be obtained from process simulators [41-42]. TWIST can then be used to optimize and develop semi-empirical models of small-geometry devices. At that point, a full two-dimensional potential and current-continuity solution would be justified [43] for the derivation of device parameters suitable for circuit-oriented simulators [11-13].

Section 4.1 gives an overview of the structure of Program TWIST. Section 4.2 describes the generation of impurity profiles and graded meshes on which the analysis can be based. Section 4.3 presents the basic physical equations and the boundary conditions used in the program. Section 4.6 describes the iteration algorithm used to solve the Poisson equation, and evaluates its performance. Section 4.7 presents the equations used to characterize devices from the self-consistent potential solution. The TWIST user's guide is presented in Appendix A, an example input together with its SUPREM input in Appendix B, the corresponding console record in Appendix C, and the program list in Appendix E.

4.1. Overview of Program TWIST

Program TWIST requires 65K 16-bit words on a Hewlett-Packard 1000 F-series computer. 32K words are used by the EMA (Extended Memory Area) to handle the data arrays. The present setup of the system is shown in Figure 4.1. The HP2648A graphics terminal provides interactive communication and graphics displays of the simulation results. These results can also be drawn on the four-color plotter HP9872A and/or a graphics hard copy unit.

The following device structures can be handled:

(a) conventional MOSFETs with uniform substrates,

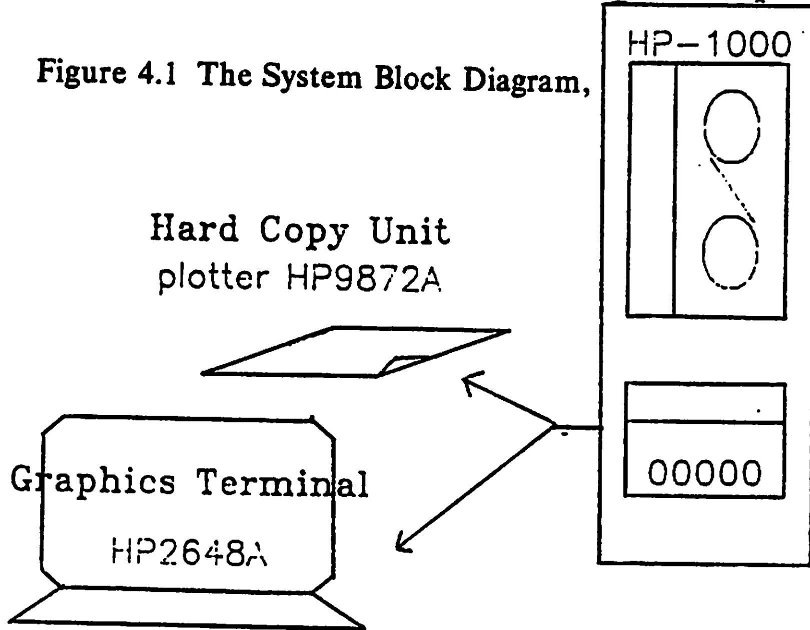
(b) enhancement and depletion MOSFETs with single or double channel implants,

(c) MOSFETs with asymmetric channel implants

The oxide covering the device to be simulated does not need to be uniform in thickness. The gate electrode can be arbitrarily located. The widths of the source and drain regions may differ.

Figure 4.2 shows the flowchart of Program TWIST. The program is divided into eight parts, the root and seven segments. The root, TWIST, is the executive program which controls the overall function. The first segment, GETPA, reads in data either from the console interactively or from a pre-defined file. The second segment, SETPA, initializes both the impurity distribution and the graded mesh, and preprocess the parameters to be used in the analysis. The third segment, SOLVE, iterates the Poisson equation using the finite-difference method at the given bias; the resolution and the accuracy can be controlled interactively. From the fourth through seventh

Figure 4.1 The System Block Diagram,



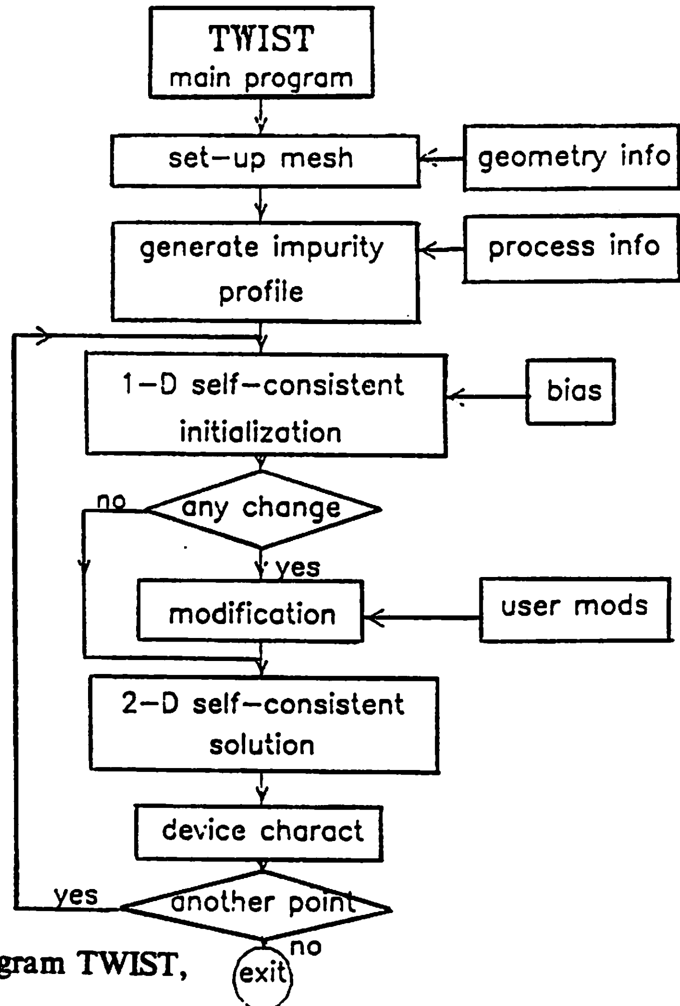


Figure 4.2

The Flow Chart of Program TWIST,

segments, OUTP1, OUTP2, OUTP3 and OUTP4, handle the graphics and numerical outputs; both can also be routed to hard copy units, i.e. either the plotter or the line printer.

4.2. Graded Mesh and Impurity Distribution

In a small-geometry MOSFET, both the potential and impurity distributions are extremely inhomogeneous. Large gradients exist in the immediate vicinity of the source and drain junctions and of the interface between oxide and silicon. In these regions, the density of the grid points on which the finite-difference equations are based should be high to ensure accuracy. The point density in the remote regions can be relatively low. For the mesh-setup purpose, the horizontal cross section of the device is divided into three regions: source, channel and drain, to which Mock's algorithm [19] is applied. The Y-constant in Mock's equation is changed to 0.05 to get reasonable mesh sizes in the surface region. A typical mesh layout is shown in Figure 4.3.

Up to three ion-implantation steps can be used to tailor the impurity profile. The first implant always covers the whole device as either the well implant of the CMOS/DMOS process or the threshold voltage implant of the NMOS process. The second one may cover only part of the device as required by the DMOS process, or the whole device as the threshold voltage implant of the CMOS process, the double implant of the NMOS process to suppress the source-to-drain punchthrough, and/or the depletion implant of the depletion-NMOS process. The third implant is the source/drain implant; it is allocated to the user-defined source/drain regions.

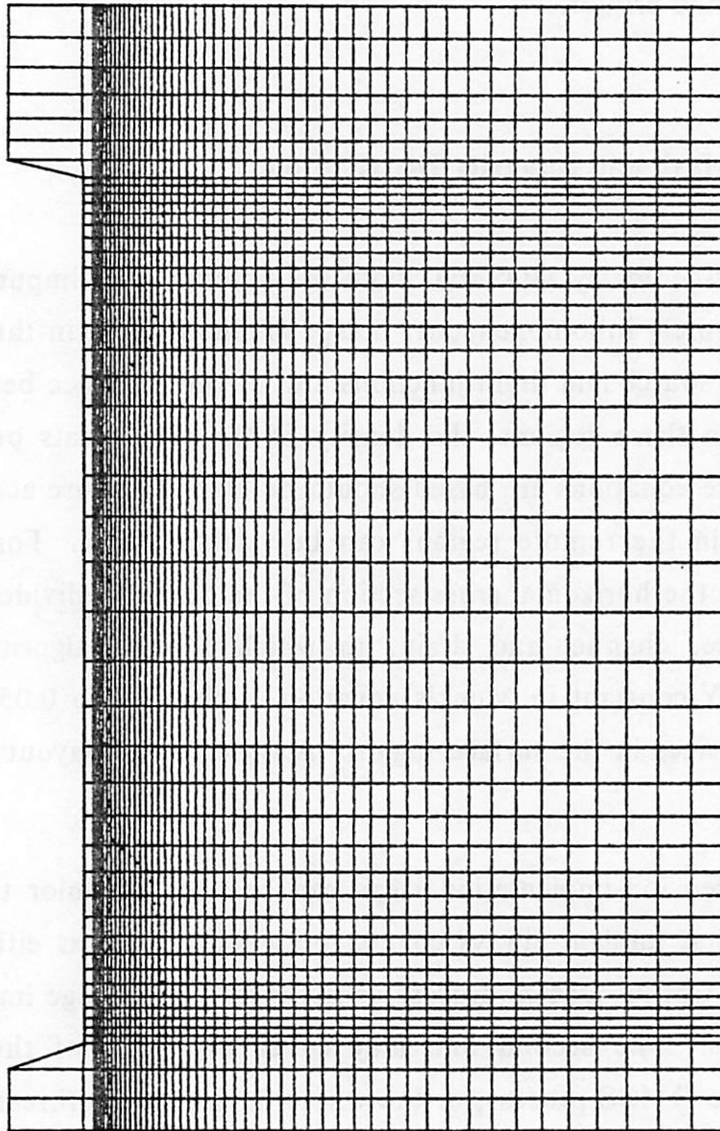


Figure 4.3 The Mesh Setup of Program TWIST,

The impurity profile generation can be either analytical or numerical. In the analytical mode, the ion-implantation profile assumes a Gaussian distribution. The two-dimensional redistribution, due to later high-temperature-process steps, is included. The expression used is [44-45]:

$$N(x,y,t) = \frac{Dose}{2\sqrt{\pi A}}\left[\Omega(y,t) + \Omega(-y,t)\right]\times\left[1 - erf\left[\frac{x - x_J}{\sqrt{4Dt}}\right]\right] \tag{4.1}$$

where

$$\Omega(y,t) = e^{-\frac{(y - R_P)^2}{A}}\left[1 + erf\left[B + Cy\right]\right] \tag{4.2}$$

The coefficients A, B and C are:

$$A = 2\Delta R_P^2 + 4Dt \tag{4.3}$$

$$B = \frac{R_P}{\Delta R_P}\left[\frac{2Dt}{A}\right]^{\frac{1}{2}} \tag{4.4}$$

$$C = \frac{\Delta R_P}{2\sqrt{DtA}} \tag{4.5}$$

The error function in the expression is evaluated by its polynomial approximation [46].

In the numerical mode, Equation (4.2) is replaced by the one-dimensional linear interpolation of the output from Program SUPREM [41]; the lateral two-dimensional redistribution is still based upon Equation (4.1); the standard deviation is estimated from the SUPREM result.

4.3. Basic Equations and Boundary Conditions

Under weak-inversion and/or weak-injection conditions, the equations of the current continuity and the electron-hole-recombination effect are ignored. The equations used to describe the physical mechanism inside the semiconductor are:

$$\nabla^2\phi = -\frac{q}{\epsilon_{si}}\left[N_D - N_A + P - N\right] \tag{4.6}$$

$$N = \frac{N_I^2}{N_{SUB}} e^{\frac{q(\phi - \phi_{F_N})}{kT}} \tag{4.7}$$

$$P = N_{SUB}\, e^{-\frac{q(\phi - \phi_{F_P})}{kT}} \tag{4.8}$$

The first one is the Poisson equation of potential ϕ. The last two are the electron and hole densities based upon Boltzmann statistics. The variables ϕ_{F_N} and ϕ_{F_P} are the quasi-Fermi levels of electrons and holes, respectively.

The interior of the oxide is assumed to be free of charges. The Laplace form of the Poisson equation is used to describe the oxide potential distribution:

$$\nabla^2\phi = 0 \tag{4.9}$$

The potential in the neutral substrate, the reference potential, is assigned zero. The potentials in the neutral source and drain regions are:

$$\phi(src/drn) = \frac{kT}{q}\ln\left[\frac{N(x,y)}{N_A}\right] + V_{APP} \tag{4.10}$$

where V_{APP} is the reverse bias voltage applied across the source/drain to the substrate junction.

Figure 4.4 is the cross section of a device with specified boundary conditions. The potentials at the two boundary planes at the left and right are determined by the self-consistent solution of the one-dimensional Poisson equation:

$$\nabla_y^2\phi = -\frac{q}{\epsilon_{si}}\left[N_D - N_A + P - N\right] \tag{4.11}$$

The boundary potentials at points A and B are calculated by Equation (4.10), i.e. they are neutral. The boundary potentials at points C and D may be zero, i.e. neutral, or be extrapolated from the potential of their neighbors based upon the quadratic equation:

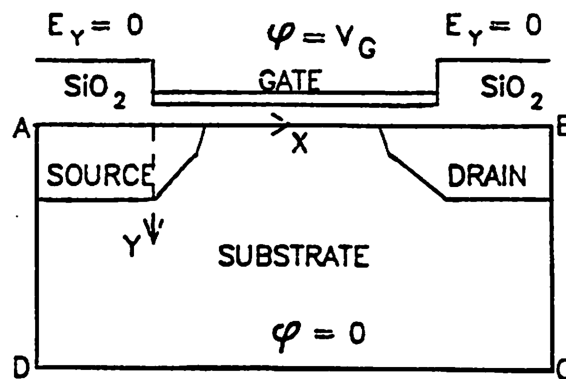


Figure 4.4 The Boundary Conditions of Program TWIST,

$$\phi_o = \frac{qN_{SUB}}{2\epsilon_{si}}\left[W_o - D_{y_o}\right]^2 \tag{4.12}$$

where W_o is the width of the depletion region to be sustained by the potential, ϕ_1, at the neighboring grid point:

$$W_o = \sqrt{\frac{2\epsilon_{si}\phi_1}{qN_{SUB}}} \tag{4.13}$$

D_{y_o} is the mesh size at point C/D. This equation is based upon the assumption of complete depletion together with a zero electrical field in the horizontal direction.

The lower boundary plane is treated in the same way as that for points C and D which has been described. The mesh points in the lower boundary plane are either in the neutral substrate where the potential is zero, or in the completely-depletion region where the potential will be calculated from Equation (4.12).

The top boundary plane consists of the gate electrode and the exposed oxide. The potential at the gate electrode is the gate voltage. The boundary at the exposed oxide serves as a reflection plane of the potential distribution. This is equivalent to assume zero normal electrical fields at the surface, i.e. no charge on the exposed oxide.

4.4. Quasi-Fermi-Level

The distribution of the electron and hole quasi-Fermi levels determines the direction and the magnitude of the total current density. The quasi-Fermi-level distribution can be solved together with the potential distribution from the Poisson equation and the current-continuity equation. If the Poisson equation stands alone, as it does in TWIST, only one unknown, the

potential, can be solved. But quasi-Fermi levels are required for the calculation of carrier densities as shown in Equations (4.7) and (4.8). Thus we need an algorithm to assign the quasi-Fermi levels at each mesh point.

The assignment of quasi-Fermi levels must satisfy the following criteria:

(a) the quasi-Fermi level should be constant along the direction without current flow,

(b) the quasi-Fermi levels of both electrons and holes are the same in the neutral regions,

To facilitate the explanation of the quasi-Fermi-level assignment, the definitions of source, drain and channel regions are clarified first. The source, drain, and channel regions differ from those defined by the user because of the two-dimensional impurity redistribution. In the case of an enhancement MOSFET, the region between the surface P-N junction and the neighboring boundary plane is defined as a source/drain region. In the case of a depletion MOSFET, the "surface junction" is defined as the turning point of surface impurity distribution, i.e. the point at which the second-order gradient of impurity distribution along the surface changes the sign. The region between the junctions is the channel region.

In the source and drain regions, only the y-direction correlation of the quasi-Fermi-level distribution is considered. The quasi-Fermi-level assignment is based upon the one-dimensional theory of a P-N junction in y-direction. In the neutral substrate, where the net charge density is less than two thirds of the impurity density, both the electron and hole quasi-Fermi levels are assigned to the substrate bias, zero. In the neutral source and drain, the quasi-Fermi levels are assigned to the bias voltages on the junctions, i.e. V_{SB} and V_{DB} respectively. In the depletion region, where the net charge density is greater than two thirds of the impurity density, E_{Fp}, the

hole quasi-Fermi level, of an N-channel MOSFET is assigned to the substrate potential, and E_{F_N}, the electron quasi-Fermi level, is assigned to V_{SB} or V_{DB}.

In an N-channel enhancement MOSFET, E_{Fp} in the channel region is always assigned to the substrate potential, because the hole current is negligible. In the neutral region next to the substrate boundary, E_{F_N} also stays at the substrate potential level. The quasi-Fermi levels will separate only if the surface depletion region exists. In the surface depletion region, two different cases must be considered. In the initialization, E_{F_N} always assumes the bias on the source junction. During the two-dimensional iterations, the horizontal correlation must be included. In a horizontal cross section, E_{F_N} is assigned regionally constant and located at either the source or drain bias, as shown in the band diagram in Figure 4.5. E_{F_N} is at the source level until, at any given depth y, the partial derivative of the electron energy in the x direction becomes negative: in other words, the partial derivative of the potential, ϕ, becomes positive. The boundary of the drain-controlled depletion region is assigned at the place where the potential assumes a further drop of $\frac{kT}{q}$ from the barrier potential. Beyond this point, E_{F_N} is assigned to the drain level. This transition will cause the electron density to drop abruptly at the location where the drain control sets in. Actually, the concentration of the electrons which are injected from the source would drop linearly, in the absence of recombination mechanism, to the boundary of drain-control.

In the channel region of a depletion MOSFET, the junction between the surface and the substrate causes the quasi-Fermi levels to separate. A neutral region may exist between the surface and the junction, and the quasi-Fermi levels also join here.

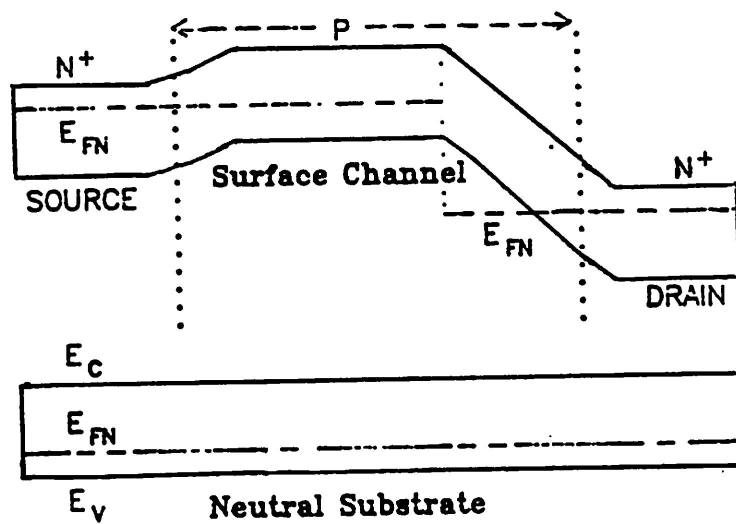


Figure 4.5 The Energy-Band Diagram of an Enhencement N-Channel MOS-FET,

4.5. Potential Initialization

Initialization is necessary and critical for the two-dimensional numerical solution of the Poisson equation. The estimated initial values determine the convergence speed to a large degree. The algorithms which converge fast require closely-estimated and smoothly-distributed initial values.

The potential distribution in a small-geometry MOSFET is inherently two-dimensional, especially in the drain-controlled depletion region which occupies a major portion of the device. The problem is further complicated by the extreme inhomogeneity of impurity distributions. Few of the existing theories can model the potential distribution in this region adequately by analytical expression. Instead of strictly abiding by the theoretical predictions of surface potential distribution, TWIST uses an empirical approach to provide quickly-evaluated, smoothly-distributed initial values. The resultant convergent speed demonstrates its validity.

In the initialization for the low V_{DS} case, the device is partitioned into five vertical domains as shown in Figure 4.6: channel, source and drain, and source- and drain-controlled depletion domains. In the channel, source and drain domains, the potential distributions can be described by the one-dimensional Poisson equation, while in the source- and drain-controlled depletion domains, the potential is a two-dimensional function of both x and y coordinates. The widths of the source- and drain-controlled depletion domains are estimated using the equations described in References [47-48].

In the channel, source and drain domains, one-dimensional self-consistent potential distributions are solved at the left and right boundary planes and the middle cross section of the channel region. These one-dimensional solutions are then assigned to the entire regions to which they belong.

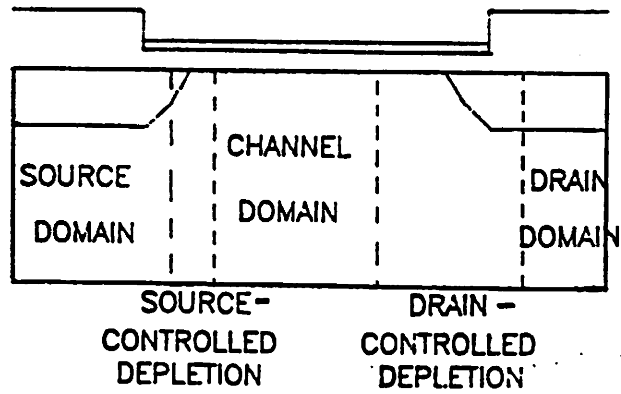


Figure 4.6 The Cross Section of a MOSFET, Divided into Five Domains for Potential Initialization in Cases of Low V_{DS},

By using these potential solutions as boundary conditions, the potentials in the source- and drain-controlled depletion domains are assigned using the equation:

$$\phi(x,y) = \phi_1(y) + \left[\phi_2(y) - \phi_1(y)\right] \times \left[1 - \sqrt{\frac{\phi_{s2} - \phi_{s1}}{\phi_2(y) - \phi_1(y)}} \left[\frac{x - x_1}{x_2 - x_1}\right]\right]^2 \tag{4.14}$$

for $\phi(x,y)$ lying between $\phi_1(y)$ and $\phi_2(y)$, and x between x_1 and x_2, where ϕ_{s1} and ϕ_{s2} are the surface potentials at the two boundary planes respectively, and $\phi_1(y)$ and $\phi_2(y)$ are the potentials in the two boundary planes. Whenever the expression gives a value greater than $\phi_2(y)$, $\phi(x,y)$ is limited to $\phi_2(y)$.

The one-dimensional self-consistent potential distributions in the channel, source and drain domains are solved using the following initial conditions:

(a) In the source and drain domains, the initial potential is based on the one-dimensional P-N junction theory with a uniform substrate. The depletion region is totally allocated inside the substrate.

(b) In the channel domain, the surface potential of either enhancement or depletion channel is estimated using approximate closed-form solutions of the Poisson equation in the condition of either strong or weak inversion. The depth of the surface depletion region is estimated and the potential is assigned accordingly.

In devices with very short channel lengths and moderate drain biases, the channel domain does not even exist. The drain- and source-controlled depletion domains merge together, as shown in Figure 4.7. The following scheme is designed to initialize this extreme situation:

(a) Determine the width of the source- and drain-controlled depletion domains based upon analytical expressions [47-48].

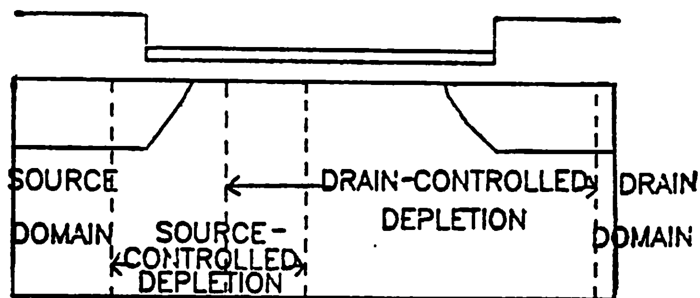


Figure 4.7 The Cross Section of a MOSFET, Divided into Three Domains for Potential Initialization in Cases of High V_{DS},

(b) Limit the drain-controlled depletion domain between the source and drain junctions.

(c) Assign the potential in the drain-controlled depletion domain using Equation (4.14).

(d) Use the middle of the overlapping region as the boundary of the source-controlled depletion domain.

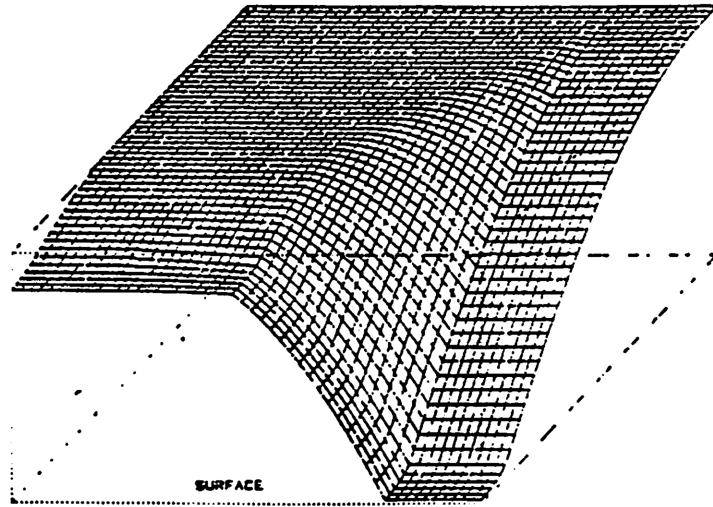
(e) Use the already assigned potentials at the boundaries as the boundary conditions and assign the potentials in the source-controlled depletion domain accordingly.

After the potential distribution of the entire device are initialized, the user may be allowed to modify it. But the auto-initialization results are adequate in most cases tried to date. Figure 4.8 shows the initial and final negative-potentials, which are directly proportional to the electron energies, of a 0.8 μm device with a channel implant and biases at $V_{GB} = 0.1V$, $V_{DB} = 5V$, $V_{SB} = 0V$.

4.6. Iteration Algorithm and Program Performance

The self-consistent two-dimensional potential distribution is then solved using an iterative method. The resultant potential, field and free carrier distributions can be displayed. The surface- and punchthrough-barrier potentials, the injection locations and the surface depletion regions are determined and displayed.

The Poisson equation in a five-point finite-difference format is solved by the Successive-Over-Relaxation algorithm with a modified alternating-



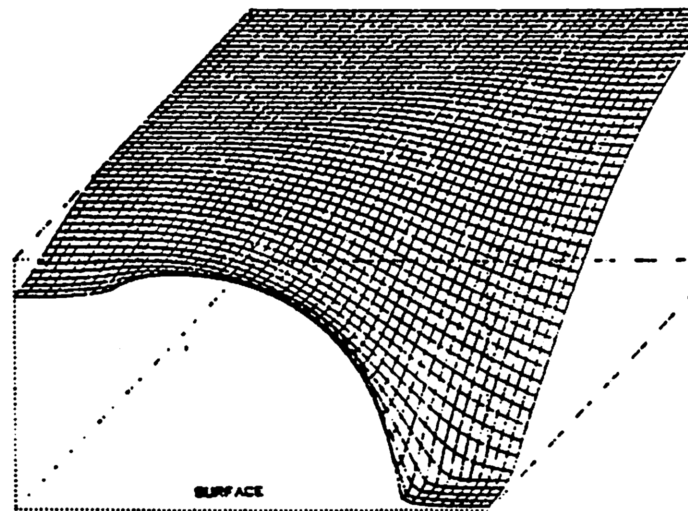


Figure 4.8 The Initialized and Resulting Negative Potential Distributions,

direction method. The mesh is scanned column by column along the horizontal direction. But the scanning direction is reversed every other iteration loop to ensure that the most recent iteration results are used to update the potential at the currently scanned point. The Gummel algorithm is modified to use the potentials of the four immediate neighbors as boundary conditions and to carry out the Newton-Raphson iteration of the potential at each point. Because the correlation is quite localized, the points which converge quickly are detected and skipped in later iteration loops to increase speed. Typically, by the third iteration, about half of the total mesh points of a uniform-mesh setup and one third of the total mesh points of a graded mesh setup are by-passed. More than two thirds of the mesh points are by-passed in later iteration loops. The computational time per iteration loop varies due to the by-pass scheme. The average computational time per two-dimensional iteration loop averages 1-2 seconds for a 50 by 50 mesh setup with $2mV$ resolution.

4.7. Device Characteristics

In the weak-inversion and/or the weak-injection region, the drain current is mainly a diffusion current injected from the source over the regional potential barrier and collected by the drain. So the current is formulated as:

$$I_{DS} = qD\frac{N_I^2 W}{N_{SUB} W_B} \int_0^{W_B} e^{\frac{q(\phi_B(y) - \phi_{SRC})}{kT}} dy \quad (4.15)$$

where W_B is the "base width", ϕ_B is the local barrier potential and ϕ_{SRC} is the source potential. Given a self-consistent potential distribution, the drain diffusion current can be calculated from the barrier potentials at the interface, in the buried channel, and/or at the saddle point at the punchthrough

ridge, and the "base width". In small structures, the barrier may be just a point in the potential profile. In these cases, the "base width" and the depth of the base cross section are calculated as the dimensions of the regions in which the potentials differ from the barrier potential by less than one or two $\frac{kT}{q}$. Although this leaves the base width ambiguous to some extent, the most dominant factor in the current equation is the exponential term depending on the barrier potentials. The larger part of design optimization rests on the control of the various local barrier potential.

CHAPTER 5

The Punchthrough

With recent technological developments in both the accuracy of process control and the fine structure of lithography patterns, the scaled-down MOSFETs promise a higher integration density and a faster switching speed. The scaling approach [49], which requires the reduction of both physical and electrical dimensions in proportion, has practical and physical limitations due to technical constraints and the non-linear relationship between geometrical and physical parameters. One of the most important problems in designing small-geometry MOSFETs is the punchthrough between the source and the drain. It is the result of the barrier lowering due to the merging of the source and drain depletion regions.

Once the punchthrough condition is reached, the current flowing from the source to the drain increases significantly as V_{DS} increases. This additional current can be viewed as an undesirable component to be avoided, or exploited as part of the conduction current in novel applications of MOSFETs [50]. Both approaches require a thorough understanding of punchthrough.

In this chapter, the punchthrough phenomenon is demonstrated by two-dimensional device simulation and theoretical analysis. Section 5.1 describes the close correlation between the punchthrough of the source and the drain and the static-feedback from the drain to the gate. Section 5.2 describes the nature of punchthrough and the locus of the injection point by the results of two-dimensional simulations using Program TWIST. Section 5.3 presents a theoretical analysis of punchthrough, based upon the assumption of uniform substrate doping.

5.1. Static Feedback and Punchthrough

Static-feedback and punchthrough effects are usually cited as two different characteristics associated with short-channel MOSFETs. As a matter of fact, the fundamental mechanisms of these two effects are very similar. Both phenomena can be described as a modulation of the potential barrier between the source and the drain by the drain voltage, V_{DS}, when the channel length is sufficiently short. The static feedback from the drain to the gate is observed as a shift in the threshold voltage due to V_{DS}, and punchthrough is observed as an abnormal current which strongly depends on V_{DS} at a medium or high V_{DS}.

Figure 5.1 shows the surface potential distribution versus the normalized channel length, as simulated by TWIST, in devices of channel lengths ranging from $20\mu m$ to $1.5\mu m$, with a uniform substrate concentration of $2.0\times10^{15} cm^{-3}$, biased at $V_{GS} - V_{FB} = 0.6V$, $V_{DS} = 0.0V$ and $V_{BS} = 0.0V$. Under such bias condition, these devices operate in the weak-inversion mode. In a long-channel device, the barrier is wide and flat and its height can be predicted by the one-dimensional Poisson equation in the direction normal to the channel. The barrier width is reduced as the channel length is shortened. In a device of intermediate channel length, the barrier height is the same as that of a long-channel device; however, the source and drain depletion regions fill most of the channel region. A shift in the threshold voltage is observed and can be explained by the overall charge conservation, as modeled by Yau and Lee respectively [51-52]. With a shorter channel length, the source and drain depletion regions merge. The barrier width is reduced to a single point and the height is lowered. In this operational mode, the surface conduction current in weak-inversion, which is independent of the drain voltage in a long-channel device, increases as V_{DS} increases. This phenomenon is called the static-feedback effect; the gate and

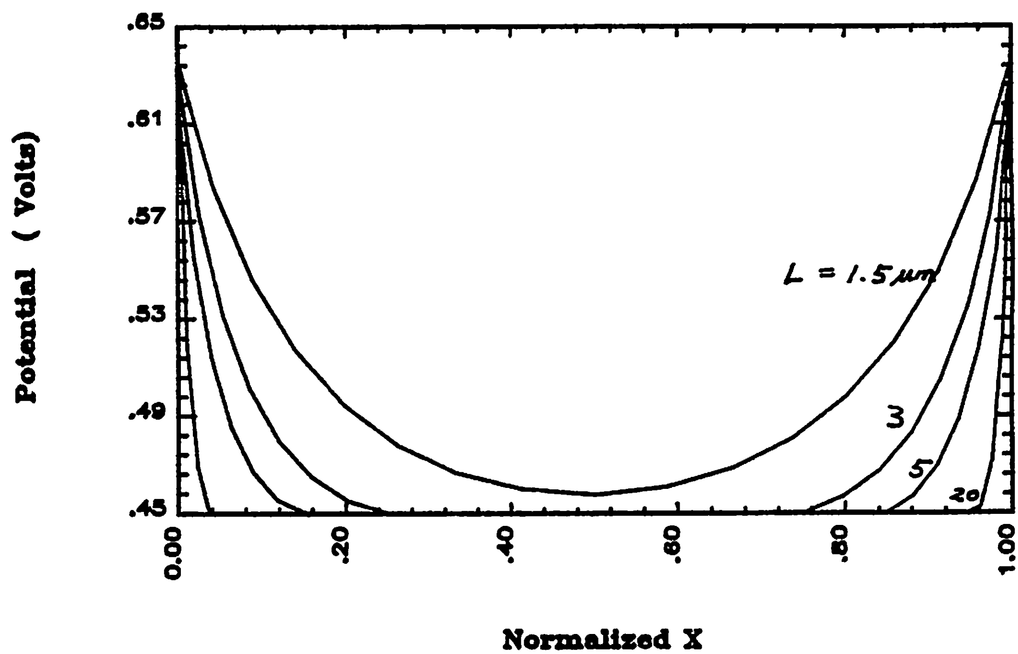


Figure 5.1 The Surface-Potential Distributions versus Normalized Channel Length with Parameter L,

the drain are coupled together. Both the gate and the drain have a direct control over the barrier. Figure 5.2 shows the surface potential distribution versus the normalized channel length in a device $1.5\mu m$ long, biased at the same V_{GS} and V_{BS} as in Figure 5.1 and at $V_{DS} = 0.0V, 2.0V$ and $4.0V$. The potential barrier seen from the source is lowered as V_{DS} increases.

The merging of the source and drain depletion regions leads to the lowering of the barrier between them. Under certain bias conditions, the barrier is deep inside the substrate and provides an alternate current path to the surface channel. Because of this extra substrate component, the drain current in the punchthrough mode is more dependent on V_{DS} than the drain current in the weak-inversion mode. Because the gate is shielded from the barrier by the depletion charge, the gate-control over the buried barrier is weaker than that in the weak-inversion mode. The carriers are injected from the source over the buried barrier and flow along the edge of the barrier minimum in the vertical cross sections. The injection barrier is located at the saddle point where the potential is the minimum in the horizontal direction and the maximum in the vertical direction.

The barrier height is lowered as V_{GS} and/or V_{DS} increase and raised as V_{SB} increases. The buried injection barrier moves toward the surface as V_{GS} and/or V_{SB} increase, and away from the surface as V_{DS} increases. The operational mode shifts gradually from punchthrough to static feedback as the injection point moves toward the surface and vice versa. In the low current region, both operational modes are barrier-controlled. The shift between static feedback and punchthrough is demonstrated by a series of two-dimensional simulations of a device with $L = 1\mu m$, $T_{OX} = 0.065\mu m$, $x_J = 0.5\mu m$, and $N_{SUB} = 0.75\times10^{15} cm^{-3}$. Figures 5.3, 5.4, and 5.5 show the equal-potential contours. The injection point is the point at which the equal-potential lines meet or form closed circles. Figure 5.3 shows the

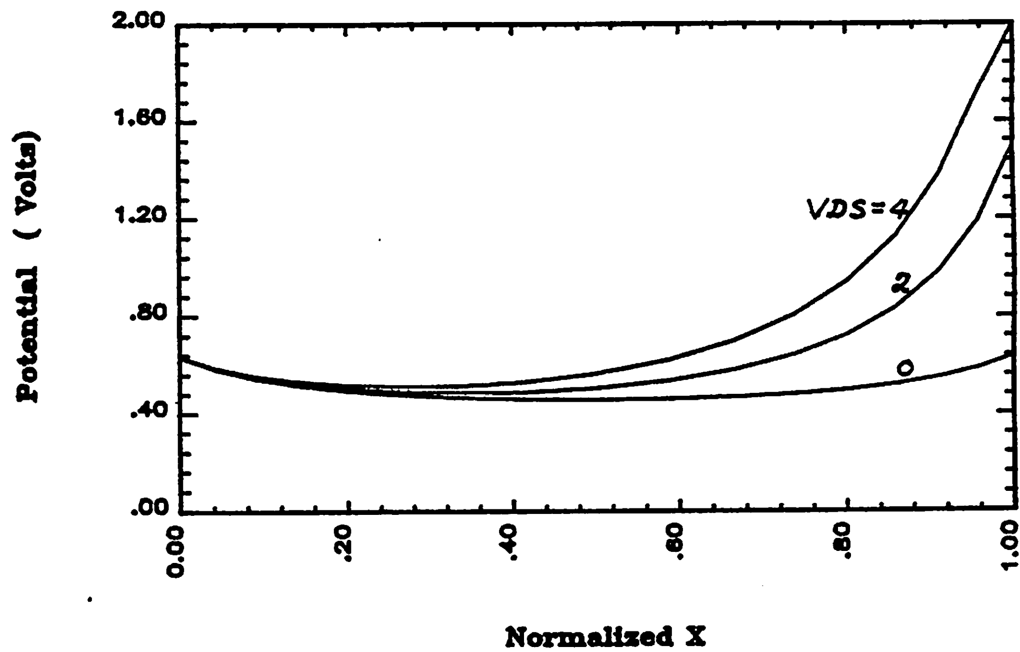
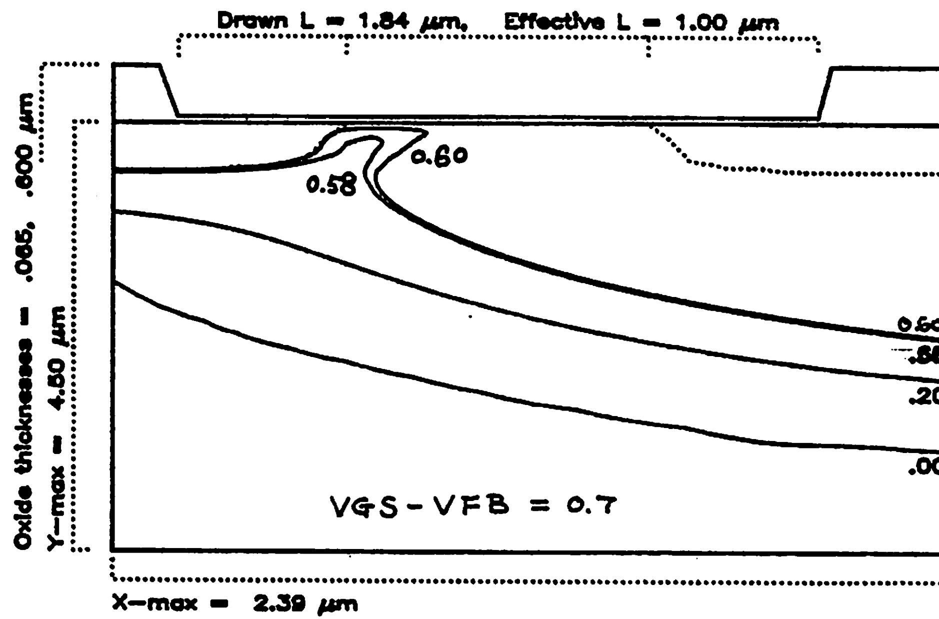


Figure 5.2 The Surface-Potential Distributions versus Normalized Channel Length with Parameter V_{DS},



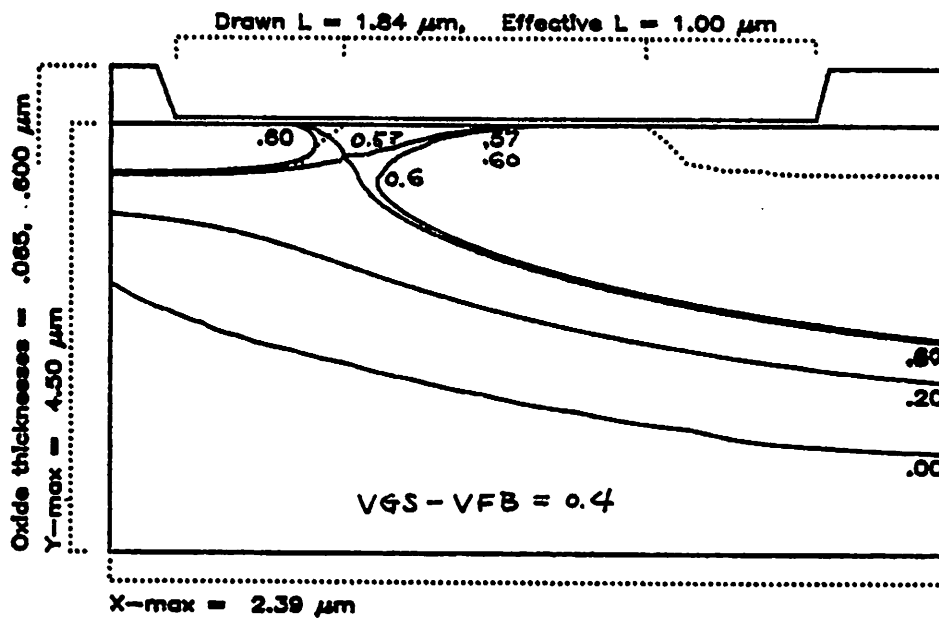
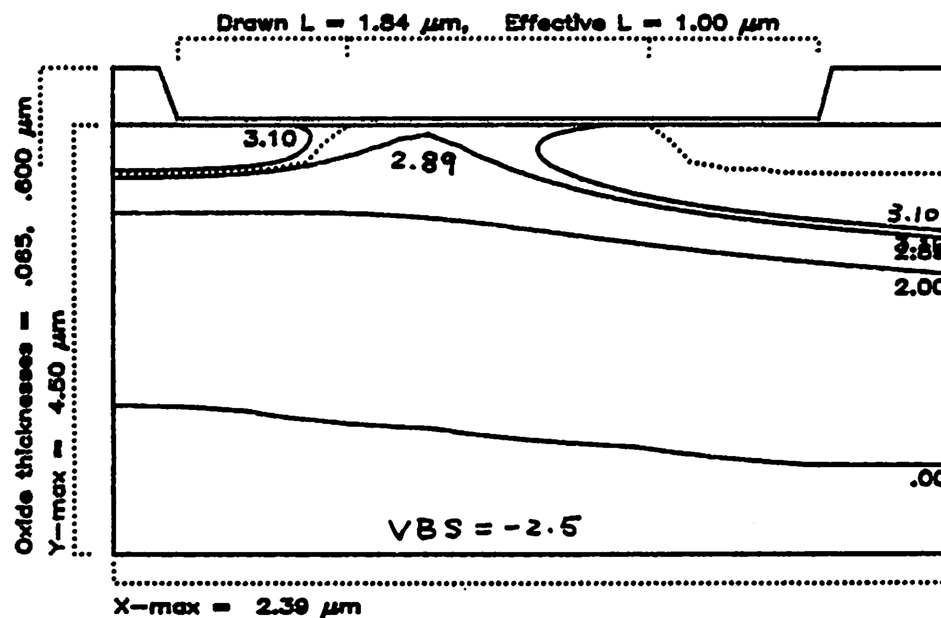


Figure 5.3 The Equal-Potential Contours at $V_{DS} = 4V$ and $V_{BS} = 0V$ with V_{GS} - V_{FB} = $0.7V$ and $0.4V$,



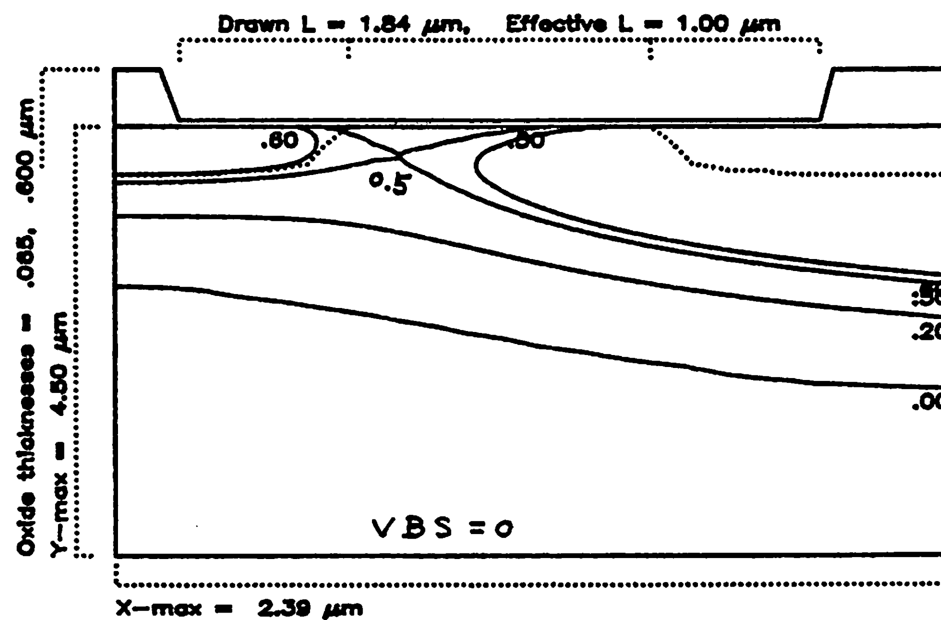


Figure 5.4 The Equal-Potential Contours at $V_{DS} = 2V$ and $V_{GS} - V_{FB} = 0.4V$ with $V_{BS} = -2.5V$ and $0V$,

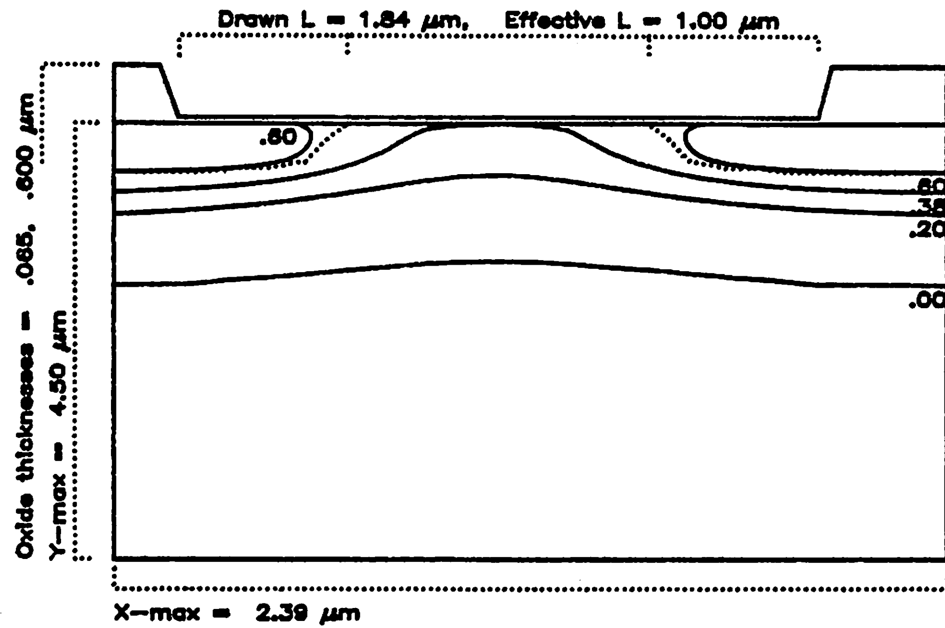


Figure 5.5 The Equal-Potential Contours at $V_{DS} = 0.0V$, $V_{GS} - V_{FB} = 0.4V$ and $V_{BS} = 0V$,

equal-potential contours at $V_{DS} = 4V$ and $V_{BS} = 0V$ with $V_{GS} - V_{FB}$ at 0.7 V and 0.4 V. The injection point moves from (x,y)=(0.554 μm, 0.0 μm) to (x,y)=(0.484 μm, 0.386 μm) and the barrier potential moves from 0.630 V to 0.567 V as V_{GS} moves from 0.7 V to 0.4 V. Figure 5.4 shows the contour plots at $V_{DS} = 2V$ and $V_{GS} - V_{FB} = 0.4V$ with V_{BS} at -2.5 V and 0.0 V. The injection point moves away from the surface as V_{BS} decreases and the height of the injection barrier is lowered. Figure 5.5 shows the contour plot of the same device at $V_{DS} = 0.0V$, $V_{BS} = 0.0V$ and $V_{GS} - V_{FB} = 0.4$. By comparing it with Figure 5.4, one sees that the injection point moves toward the surface and the barrier height increases as V_{DS} decreases.

5.2. The Saddle Point

When a long-channel MOSFET is biased at $V_{DS} = 0.0V$, $V_{BS} = 0.0V$ and $V_{GS} = V_{FB}$, the channel region assumes a constant potential from the surface through the substrate. In a short-channel MOSFET with the same process parameters at the same bias, the flatband configuration is modified by the merged depletion regions. Inside the overlapped depletion region, charges are shared between the source and drain junctions. The more the charges are shared, the lower the potential barrier is sustained. The barrier height is lowered and the potential distribution at the channel center is no longer flat. Since more charges are shared at the surface than inside, the deviation from the flatband condition is largest at the surface. However, the gate voltage tends to hold the potential at the flatband. The gate influence decreases as it penetrates the substrate. The combined effects of the gate voltage and the merging of the depletion regions result in a potential maximum in the vertical cross sections as shown in Figure 5.6(a). The horizontal minimum of the potential distribution in each cross section is located at the center of the device because V_{DS} is zero. Thus the resulting potential distribution looks like a

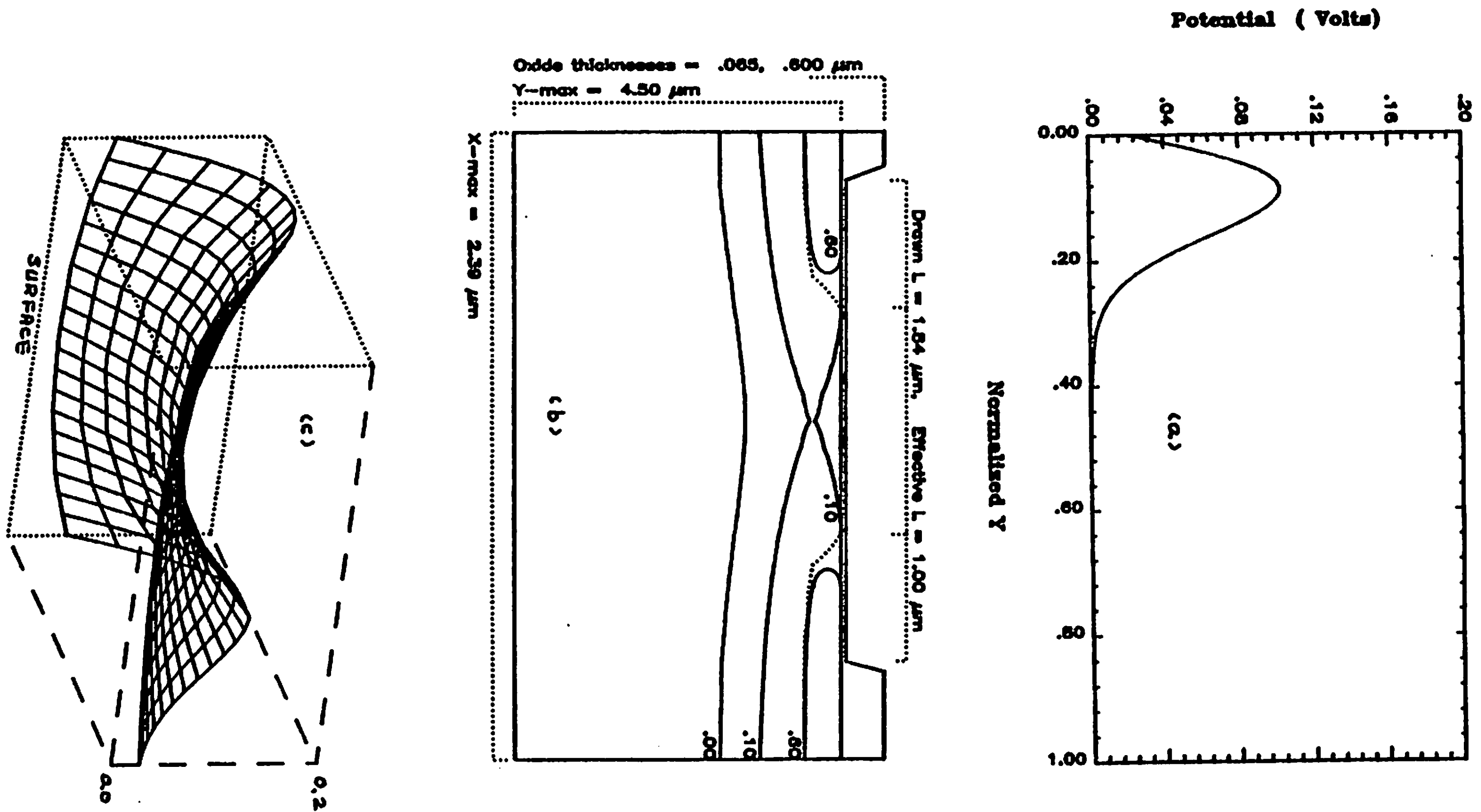


Figure 5.6 Potential Distributions at $V_{DS} = 0V$, $V_{BS} = 0V$ and $V_{GS} - V_{FB} = 0V$, (a) Potential Distribution in the Normal Cross Section of the Channel Center, (b) Equal-Potential Contours, (c) Three-Dimensional Potential Distribution,

saddle. The equal-potential contours and the three-dimensional potential distribution are shown in Figures 5.6(b) and (c) respectively.

The saddle point's location and its potential can be modulated by the bias of the device as shown in Figures 5.3, 5.4, and 5.5. The locus of the saddle points and the barrier potential and height are listed in Table 5.1 as a function of biases:

Table 5.1						
VDS (V)	VGS (V)	VSB (V)	Saddle (V)	Height (V)	XB um	YB um
1.0	.40	0.0	.422	.460	.718	.192
2.0	.40	0.0	.501	.381	.632	.297
3.0	.40	0.0	.548	.334	.554	.336
4.0	.40	0.0	.567	.315	.484	.336

Table 5.1 Properties of the Saddle Point as a Function of Biases

5.3. Quasi-One-Dimensional Analysis

As illustrated in the previous section, in either the static-feedback or punchthrough mode, the influences of V_{GS}, V_{BS} and V_{DS} are coupled together. The potential distribution is inherently two-dimensional. The author handles the analysis by decoupling the two-dimensional Poisson equation into a simpler one-dimensional format and introducing quasi-empirical parameters to model two-dimensional effects on the onset voltage of punchthrough.

One- and two-dimensional charged systems differ in charge sharing. In a two-dimensional system, the charge supports the potential differences in both the x and y directions while in a one-dimensional system, the charge

supports the potential difference in only one direction. In MOSFETs in the barrier-controlled mode, the potential barrier in either direction is supported by only part of the depletion charge between the source and the drain. This can be modeled by replacing the doping concentration in the pseudo depletion region by a higher pseudo concentration. The pseudo depletion region can be penetrated only by applying a higher bias across the source and drain junctions than predicted by one-dimensional theories.

In the two-dimensional Poisson equation:

$$\nabla_x^2\phi + \nabla_y^2\phi = -\frac{\rho(x,y)}{\epsilon_{si}} \tag{5.1}$$

The left side of Equation (5.1) can be replaced by $\frac{qN_A}{\epsilon_{si}}$ in the case of an N-channel MOSFET. The term of the second-order derivative in the vertical direction, y, contributes to the pseudo density in the pseudo depletion region. In the punchthrough mode, the second-order derivative at the saddle point can be approximated as:

$$\nabla_y^2\phi = \frac{1}{L_D}\left[\frac{V_B - \phi}{W_1} - \frac{\phi - \phi_s}{W_0}\right] \tag{5.2}$$

$$= -\frac{1}{L_D}\left[\frac{\phi}{W_1} + \frac{\phi - \phi_s}{W_0}\right] \tag{5.3}$$

where W_1 and W_0 are the widths of the depletion regions below and above the saddle point in the vertical cross section. They can be approximated by $W_1 = \sqrt{\frac{2\epsilon_{si}\phi}{qN_A}}$ and $W_0 = \sqrt{\frac{2\epsilon_{si}(\phi - \phi_s)}{qN_A}}$ respectively. L_D is the Debye length, $\sqrt{\frac{2\epsilon_{si}\, kT/q}{qN_A}}$. After substituting W_1, W_0 and L_D in the above equation, $\nabla_y^2\phi$ can be expressed as:

$$\nabla_y^2\phi = -\frac{qN_A}{2\epsilon_{si}}\left[\sqrt{\frac{\phi}{kT/q}} + \sqrt{\frac{\phi - \phi_s}{kT/q}}\right] \tag{5.4}$$

This is a negative quantity. By moving this term to the right side of Equa-

tion (5.1), the equation can be rewritten as:

$$\nabla_x^2 \phi = \frac{qN_A}{\epsilon_{si}} - \nabla_y^2 \phi \tag{5.5}$$

$$= \frac{qN_A}{\epsilon_{si}} \left[1 + \frac{1}{2} \left[\sqrt{\frac{\phi}{kT/q}} + \sqrt{\frac{\phi - \phi_s}{kT/q}} \right] \right] \tag{5.6}$$

The two-dimensional effect is equivalent to an increase in the effective substrate doping concentration. By defining the scaling factor of the substrate doping concentration, F, as:

$$F = 1 + \frac{1}{2} \left[\sqrt{\frac{\phi}{kT/q}} + \sqrt{\frac{\phi - \phi_s}{kT/q}} \right] \tag{5.7}$$

The Poisson equation then assumes a one-dimensional format:

$$\nabla_x^2 \phi = \frac{qN_A}{\epsilon_{si}} F \tag{5.8}$$

By using an effective doping density, $N_A F$, the one-dimensional concept of punchthrough can be applied.

As illustrated in the one-dimensional band diagram in Figure 5.7, when the source and drain depletion regions merge and the barrier potential is lowered so that it is less than the junction built-in potential, the source is essentially forward biased and the punchthrough current starts to flow. At the onset of punchthrough, the relationship of the depletion widths to each other is formulated, based upon the assumption of complete depletion:

$$W_D + W_S = L \tag{5.9}$$

where W_D and W_S are the depletion region widths of the source and drain junctions:

$$W_S = \sqrt{\frac{2\epsilon_{si}\phi_J}{qN_A F}} \tag{5.10}$$

$$W_D = \sqrt{\frac{2\epsilon_{si}\left[\phi_J + V_{PT}\right]}{qN_A F}} \tag{5.11}$$

The onset voltage of punchthrough, V_{PT}, can be expressed as:

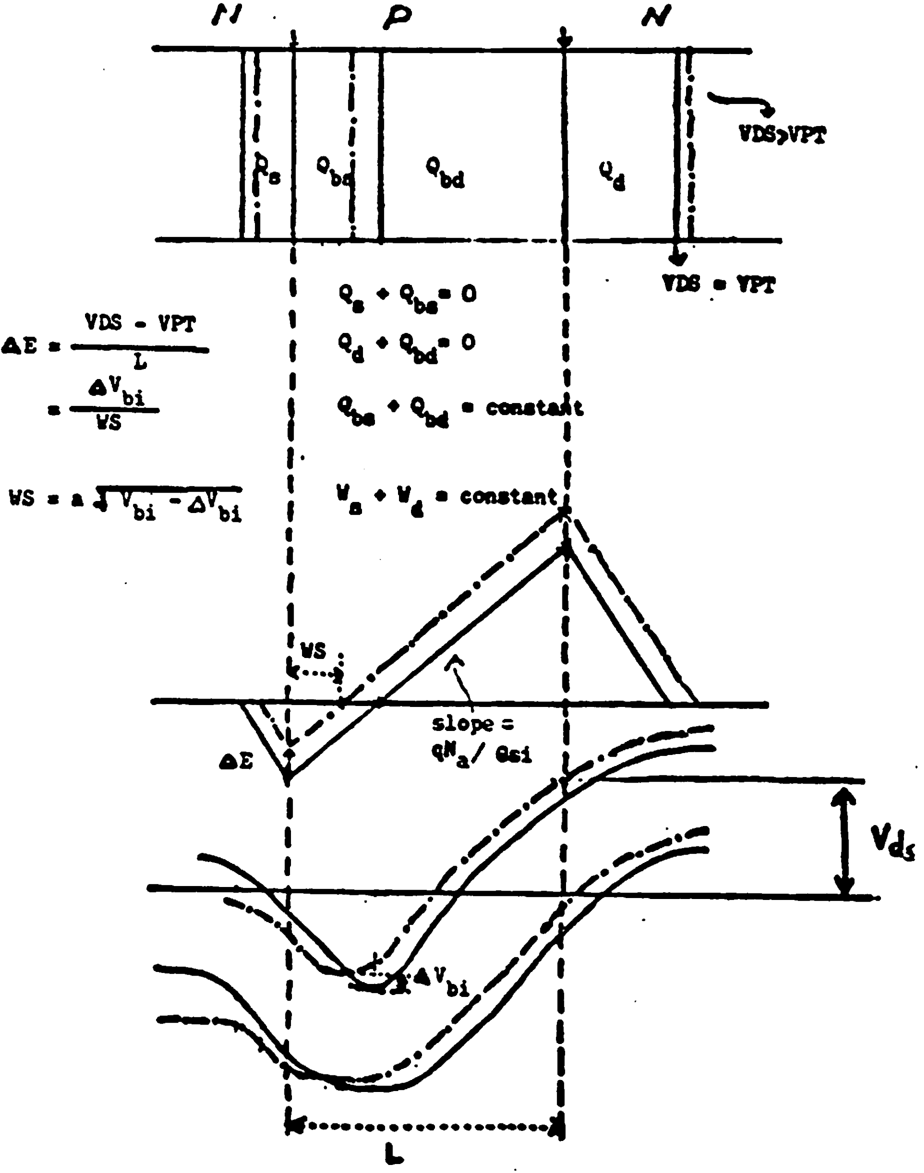


Figure 5.7 The One-Dimensional Potential Distribution between the Source and Drain Junctions at Punchthrough,

$$V_{PT} = \left[\alpha L\right]^2 F - 2\alpha L\sqrt{\phi_J F} \tag{5.12}$$

where

$$\alpha = \sqrt{\frac{2\epsilon_{si}}{qN_A}} \tag{5.13}$$

By substituting the explicit expression F in Equation (5.13), V_{PT} becomes:

$$V_{PT} = \alpha L^2\left[1 + \frac{1}{2}\left[\sqrt{\phi_J + V_{SB}} + \sqrt{\phi_J + V_{SB} - \phi_s}\right]\right] \tag{5.14}$$

$$- 2\alpha L\sqrt{\phi_J\left[1 + \frac{1}{2}\left[\sqrt{\phi_J + V_{SB}} + \sqrt{\phi_J + V_{SB} - \phi_s}\right]\right]}$$

ϕ_J is the barrier height between the source and the drain at the onset of punchthrough. It is a constant, while ϕ_s is the value of the surface potential with the reference potential at V_B. In the barrier-controlled mode, the surface is weakly inverted and the surface potential, ϕ_s, can be approximated as a linear function of V_G:

$$\phi_s = aV_{GB} + b \tag{5.15}$$

$$\phi_s = a\left[V_{GS} - V_{BS}\right] + b \tag{5.16}$$

The second square-root term in the expression F depends only on V_{GS}. As illustrated in the previous section, V_{GS} and V_{BS} can affect the barrier potential and its location. In other words, they modulate the pseudo substrate doping density.

A major feature of Equation (5.12) is the dependence of V_{PT} on V_{BS} and V_{GS}. If the two-dimensional effect factor, F, is ignored, that the depletion regions have just merged, the intrinsic onset voltage of punchthrough, V_{PT}^o, is independent of V_{BS} and V_{GS}:

$$V_{PT}^o = \left[\alpha L\right]^2 - 2\alpha L\sqrt{\phi_J} \tag{5.17}$$

An empirical formulation of V_{PT} is proposed based upon Equation (5.12):

$$V_{PT} = \alpha + \beta\sqrt{V_{SB} + \phi_J} + \delta\sqrt{V_{GS} - V_{FB}} \tag{5.18}$$

Three empirical factors, α, β and δ are introduced.

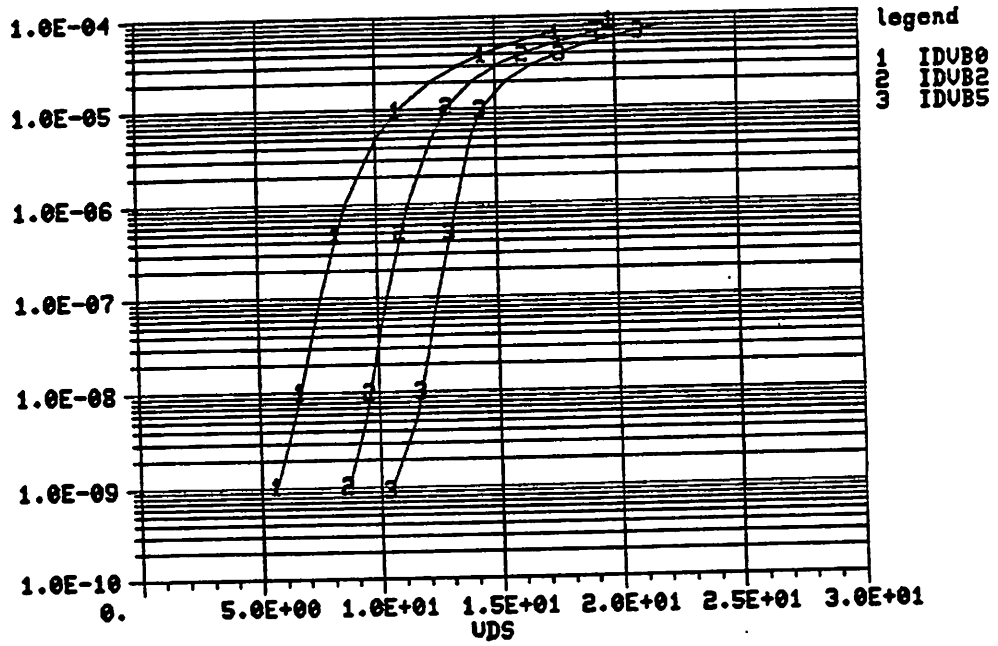
Experimentally, V_{PT} can be defined as the drain voltage at which the normalized drain current, $\frac{L}{W} I_{DS}$, is below $10^{-9}amp$ and V_{GS} is below V_{FB}. A low V_{GS} is chosen to suppress the surface current. Figure 5.8 shows the measured I_{DS}-versus-V_{DS} of device A with parameter V_{GS} and V_{BS}. Figures 5.9 shows the characteristics of device B. V_{PT}-versus-V_{GS} with parameter V_{BS} and V_{PT}-versus-V_{BS} with parameter V_{GS} of device A are plotted in Figure 5.10. The characteristics of device B are plotted in Figure 5.11. The device parameters and punchthrough coefficients which are obtained by linear regression are listed in Table 5.2:

Table 5.2			
Param	Unit	Dev. A	Dev. B
W	um	50	50
L	um	1.53	1.8
Na	1E15 cm-3	1.78	0.75
Tox	um	0.08	0.065
Xj	um	0.65	0.5
Vto	V	0.05	-0.2
VFB	V	-0.86	-1.01
alpha	V	-0.08	4.79
beta	sqrt V	3.93	4.23
delta	sqrt V	1.94	3.6

Table 5.2 Device Parameters and Punchthrough Coefficients

The limitation in circuit design is determined by the maximal allowable leakage current. The onset of punchthrough conduction can be used as a limiting voltage in circuit-design applications.

DEVICE A AT VGS=-2



DEVICE A AT VBS=0

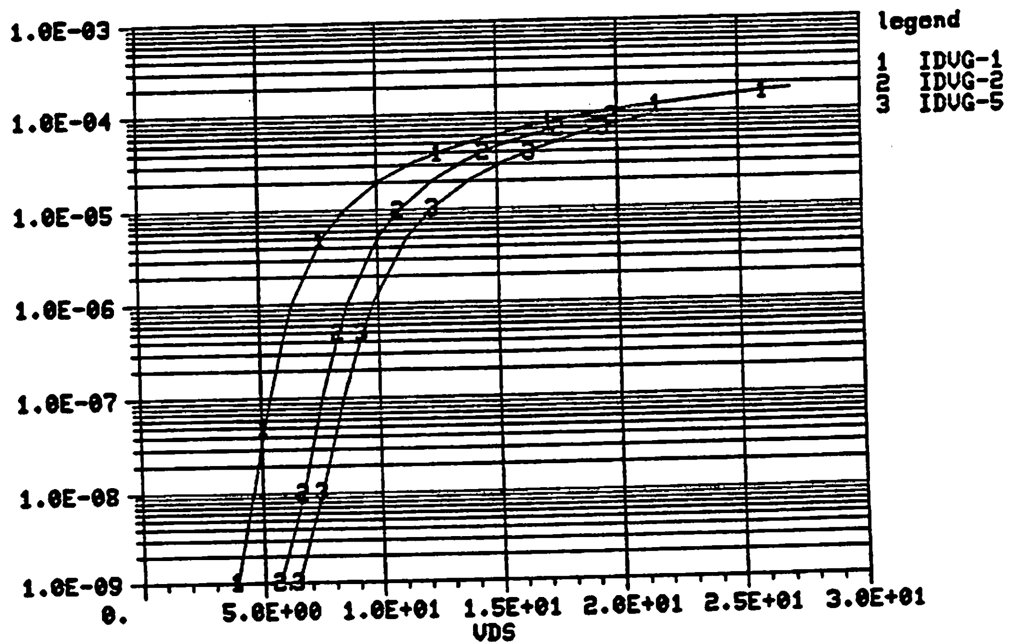
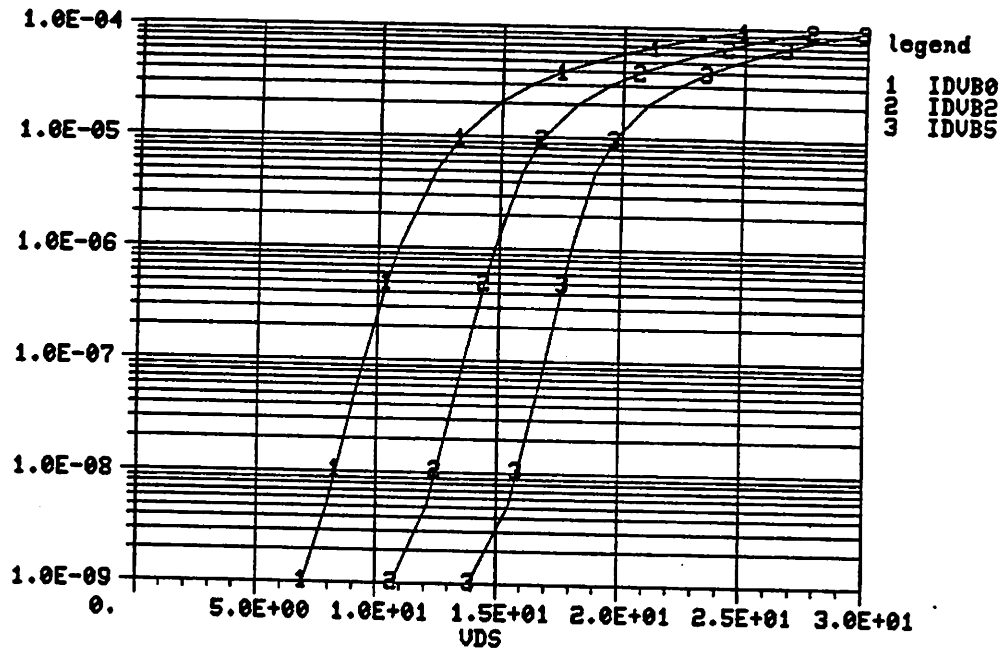


Figure 5.8 The Measured I_{DS}-versus-V_{DS} with Parameters V_{BS} and V_{GS} of Device A,



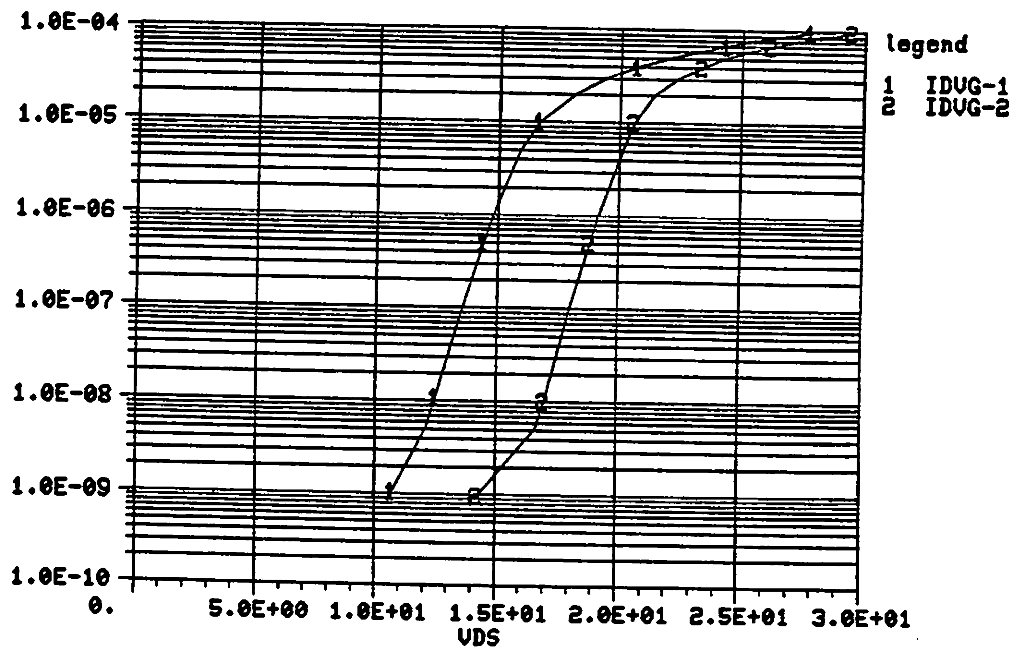
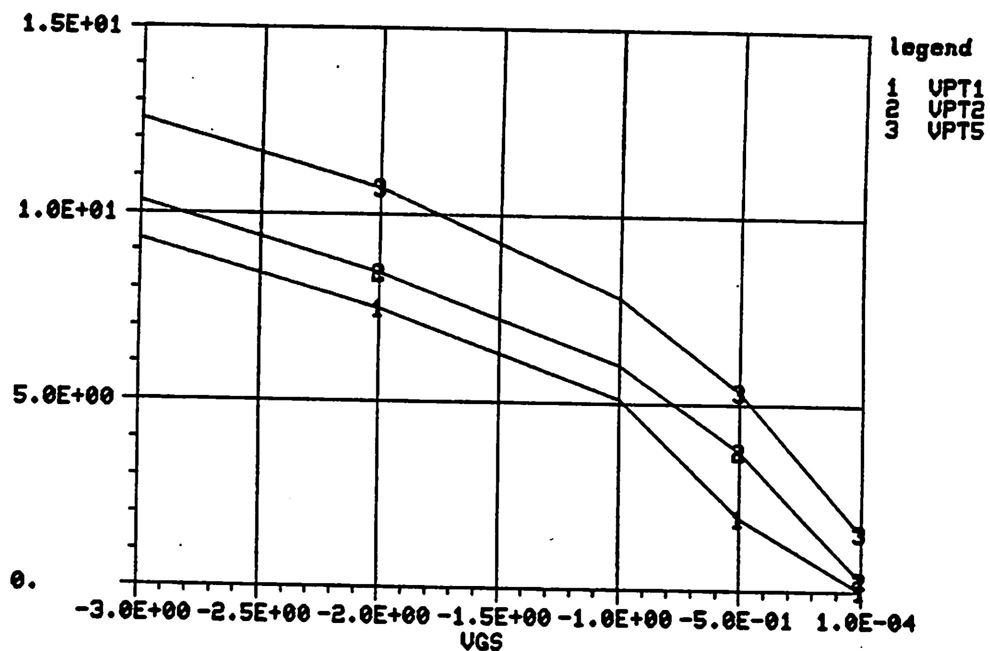


Figure 5.9 The Measured I_{DS}-versus-V_{DS} with Parameters V_{BS} and V_{GS} of Device B,



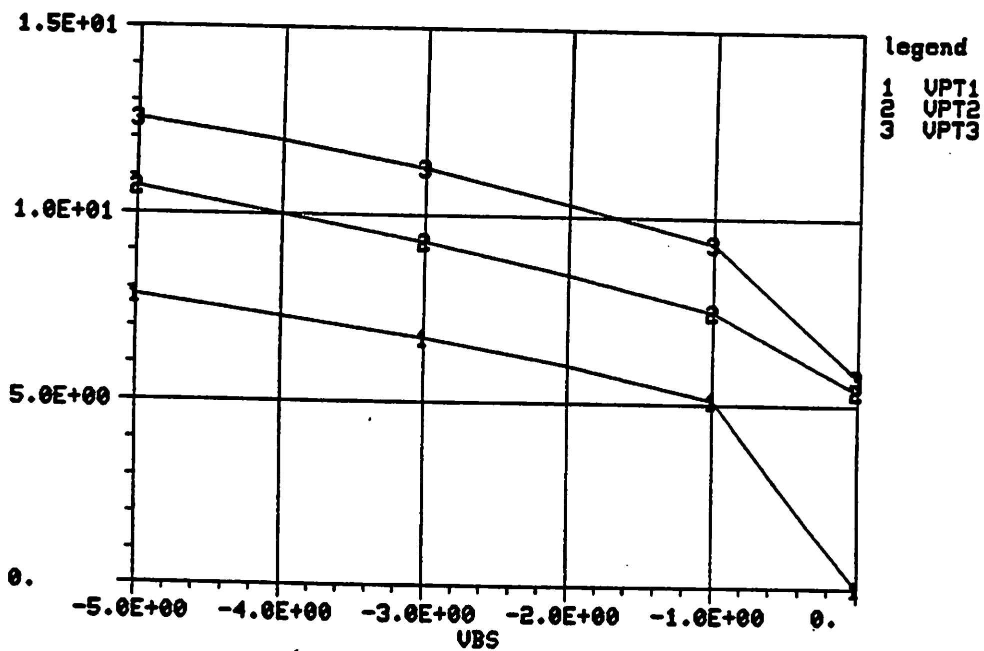
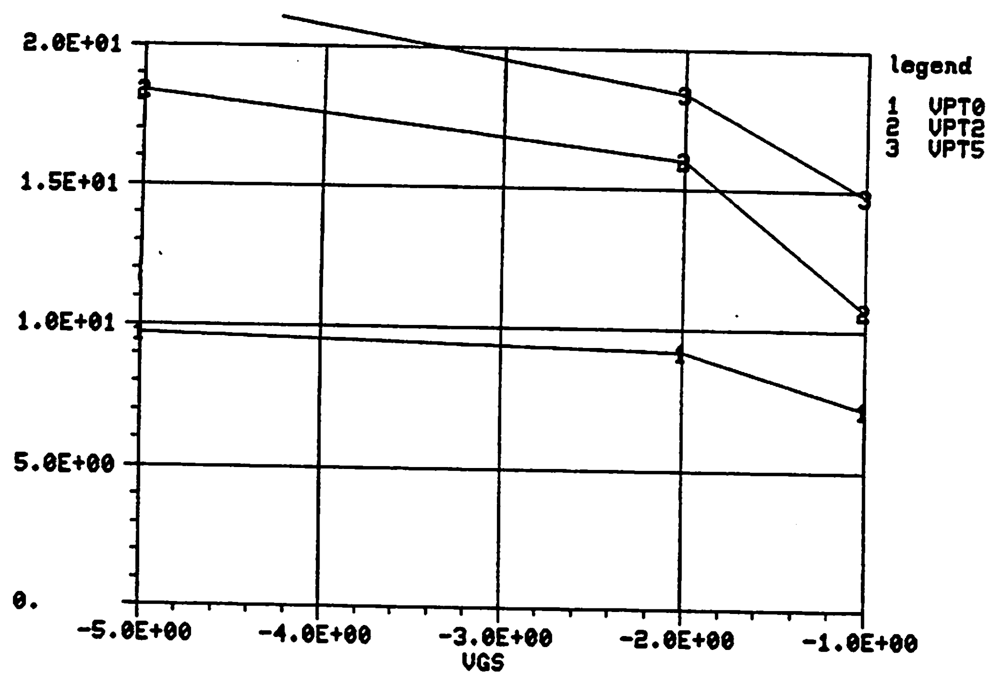


Figure 5.10 V_{PT}-versus-V_{GS} with Parameter V_{BS} and V_{PT}-versus-V_{BS} with Parameter V_{GS} of Device A,



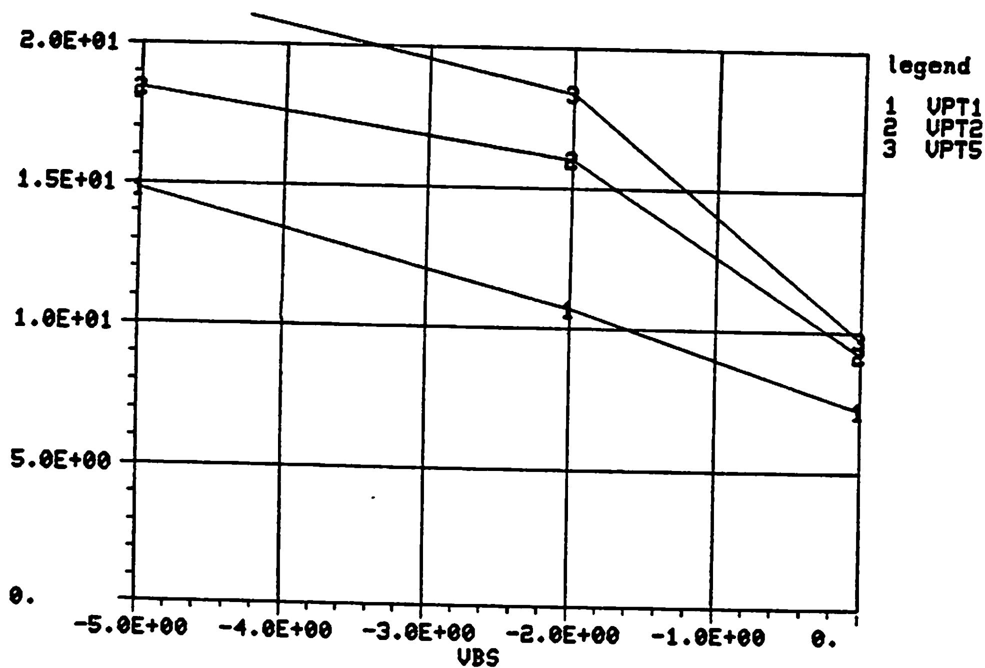


Figure 5.11 V_{PT}-versus-V_{GS} with Parameter V_{BS} and V_{PT}-versus-V_{BS} with Parameter V_{GS} of Device B,

CHAPTER 6

Small-Geometry Effects and the Semi-empirical Model MOS3

Besides the punchthrough phenomenon, advances in process technology to achieve small device geometries also bring into prominence other effects which do not appear in the characteristics of the devices of long or even medium channel lengths. In very small MOSFETs, the electrical dimensions, which determine the device characteristics, could not always scale proportionally as the other parameters. For example, the depletion-layer width of a P-N junction is related to the built-in potential of the junction by a square-root expression. The built-in potential depends logarithmically on the substrate doping density. Thus the depletion-layer width does not scale linearly with the doping density. Keeping the operational voltage constant during scaling, instead of strictly following the scaling rules, causes high-field phenomena such as the velocity saturation of hot electrons.

A small-geometry MOSFET with $L \leq 2\mu m$ and $W \leq 2\mu m$ can be characterized by the following features:

(a) Threshold-voltage sensitivity to the length and the width of the device due to the two-dimensional nature of the potential distribution;

(b) Threshold-voltage sensitivity to the drain voltage due to the drain-induced lowering of the barrier;

(c) Relaxed transition between the linear and saturation regions due to the velocity saturation of hot electrons;

(d) Lowered saturation voltage and current due to the velocity saturation of hot electrons.

This chapter describes the MOS3 model which was developed and implemented in the circuit simulation program SPICE2 (Version 2G.1) to address the above problems and attain computational efficiency. The companion capacitance model, which conserves charge [53], is also derived and implemented.

As demonstrated in the previous chapter, the ultimate difficulty in developing a model for small devices is the correct treatment of the two-dimensional configuration. A complete analytical solution is too complex for the application in circuit simulations in even the simpliest ideal case, not to mention the more complicated two-dimensional simulations. A semi-empirical modeling approach is a compromise between simulation accuracy and computational efficiency. In addition, if the model equations can be writtened in a simple format, they will allow easy parameter extraction, a property which is as critical as the accuracy of the model itself.

The semi-empirical approach to MOSFET modeling proceeds as follows:

(a) Perform two-dimensional analyses, taking into account the details of the device configuration;

(b) Derive semi-empirical relationships between parameters based upon the linearization of the two-dimensional simulation results. The resulting model is accurate only within the range of process parameter variations for which two-dimensional results are investigated.

The threshold voltage is the most critical parameter and is investigated in Section 6.1. The threshold-voltage sensitivity of small devices is emphasized. Section 6.2 presents the basic current equation upon which the model is based. In Sections 6.3 to 6.7, second-order effects, including mobility modulation, velocity saturation, channel-length modulation and capacitances, are described in sequence. The last section compares the MOS2 and

MOS3 models. Examples of circuit simulations are included to demonstrate the validity of the semi-empirical model.

6.1. Threshold Voltage

The threshold voltage, V_{TH}, of a long-channel device, say $L>20\mu m$, depends on the substrate bias. Their relationship can be predicted by applying the charge-conservation principle to a vertical cross section of the device in the middle of the channel. This cross section is bounded by the gate and the substrate. In the channel:

$$Q_g = -\left[Q_{DEP} + Q_{INV}\right] \tag{6.1}$$

and $Q_{INV} = 0$ at $V_{GS} = V_{TH}$. Q_g, Q_{DEP} and Q_{INV} are the gate, depletion and inversion charges respectively. The result is:

$$V_{TH,L} = V_{FB} + \phi + \sqrt{2\epsilon_{si} q N_{SUB}(\phi + V_{SB})} \tag{6.2}$$

where $V_{TH,L}$ is the threshold voltage of long-channel devices, V_{FB} the flatband voltage, ϕ the surface potential at threshold and V_{SB} the source to substrate bias. This expression is not applicable for small devices whose threshold voltages should be determined by the overall charge conservation [48,51-52] of the entire channel.

In the lengthwise cross section of a device, the field lines which originate in the depletion charges near the ends of the channel terminate at the source or the drain, instead of the gate. Thus, the effective threshold voltage is reduced as the channel length is shortened.

On the other hand, the depletion region actually extends beyond the edges of the channel widthwise. The extra charges at the edges sustain field lines which terminate at the gate and result in a higher threshold voltage as

compared to a wide device. The threshold voltage increases as the channel width decreases.

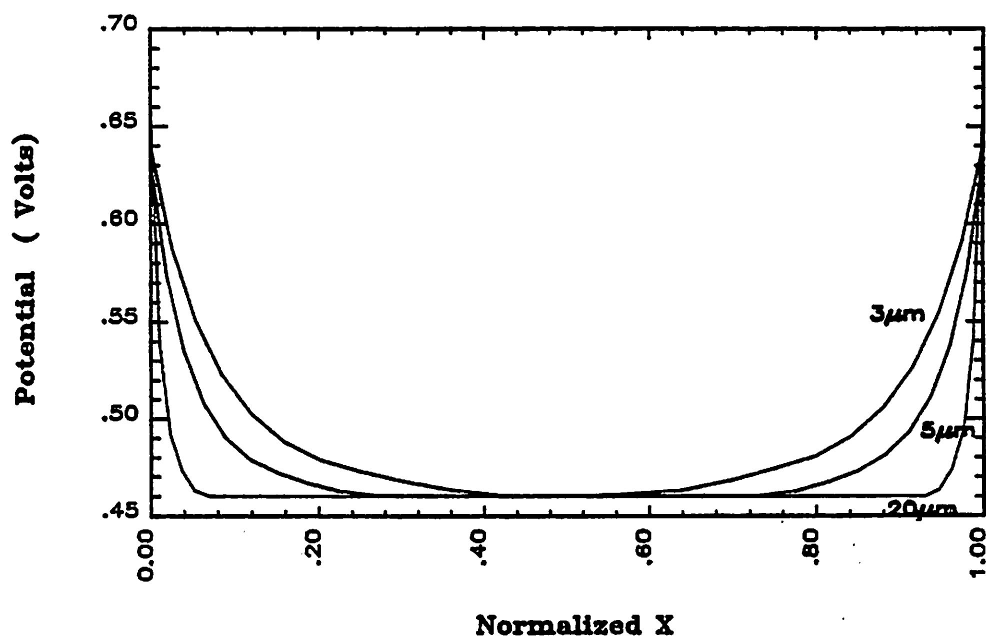
In the case of a long, wide device, the percentage of the charge at the edges is negligible. In a short and/or narrow device, this percentage is significant enough to make the shift in the threshold voltage visible. In MOS3, the effects of the channel length and width and the drain voltage on the threshold voltage are decoupled.

6.1.1. Short-Channel Effects

In a device of intermediate channel length, say $5\mu m$, the surface potential barrier is the same as that of a long-channel device, say $20\mu m$. However, the source and drain depletion regions at the edges occupy most of the channel cross section. The surface potential at the edge is higher than that at the corresponding region of a long-channel device and results in a higher concentration of conduction carriers, as shown in Figure 6.1. The average channel conductance is higher than that of a long-channel device, and the effective threshold voltage is lowered. By assuming a trapezoidal partition [51] of the depletion charge, and applying the cylindrical-junction approximation [48], one can formulate the correction factor of the threshold voltage as:

$$F_S = 1 - \frac{x_J}{L}\left[\left[\frac{W_C}{x_J} + \frac{L_D}{x_J}\right]\sqrt{1 - \left(\frac{W_P/x_J}{1 + W_P/x_J}\right)^2} - \frac{L_D}{x_J}\right] \tag{6.3}$$

which is used to multiply the depletion-charge term, the square root, in Equation (6.2). The threshold-voltage shift due to the short-channel effect with $N_{SUB} = 5.0\times10^{15}cm^{-3}$, $T_{OX} = 0.1\mu m$, $x_J = 0.5\mu m$, and a channel length of $2\mu m$ to $20\mu m$, is plotted in Figure 6.2. The difference between the plane-



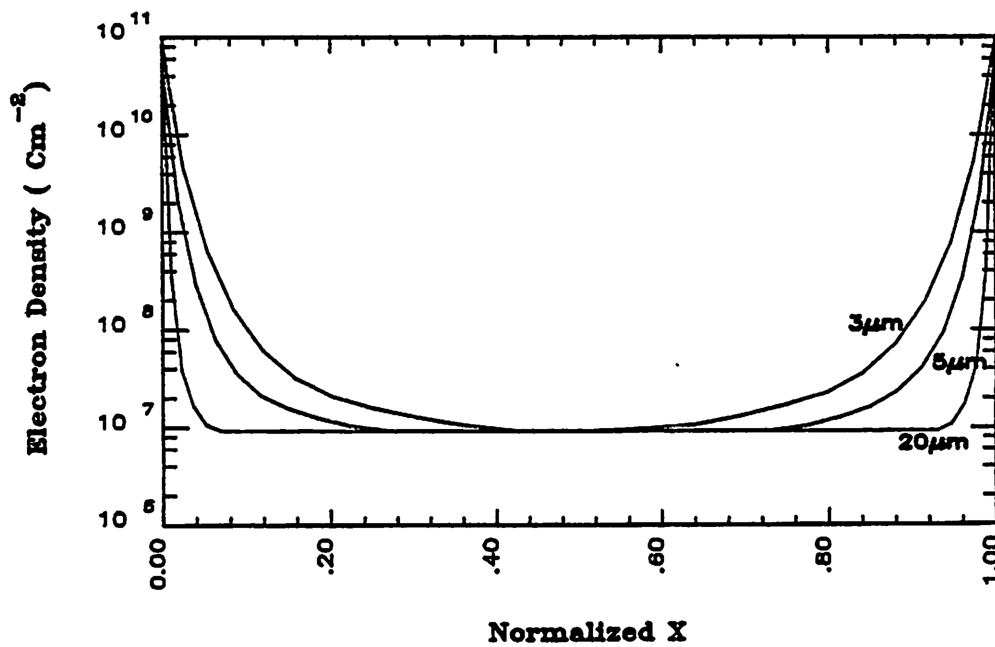


Figure 6.1 Surface Potential and Carrier Density versus Normalized Channel Lengths,

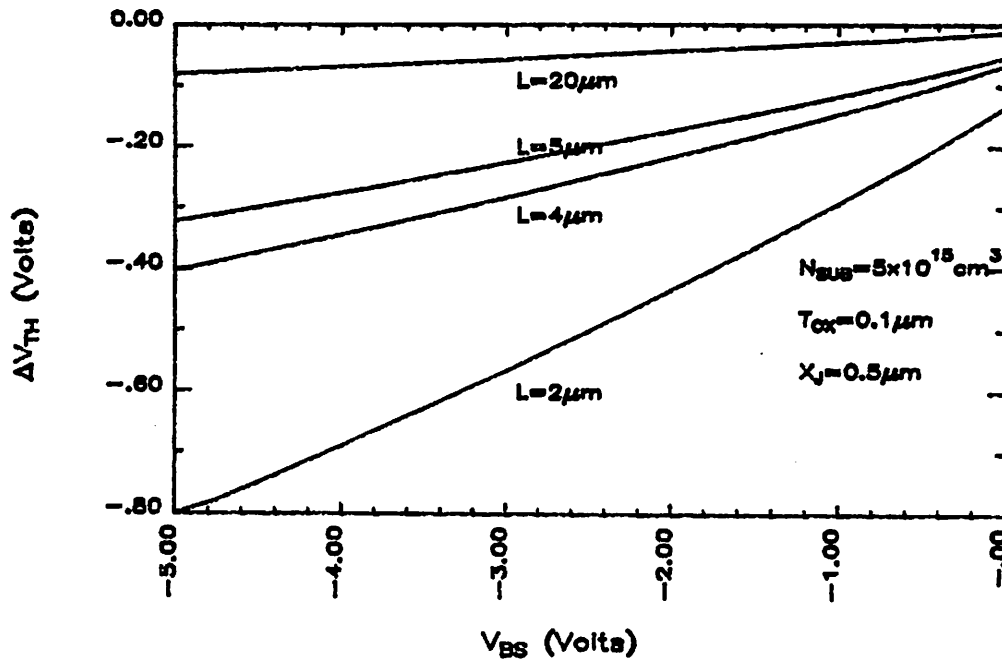


Figure 6.2 The V_{TH} Shift versus V_{BS} with Channel Length as Parameter,

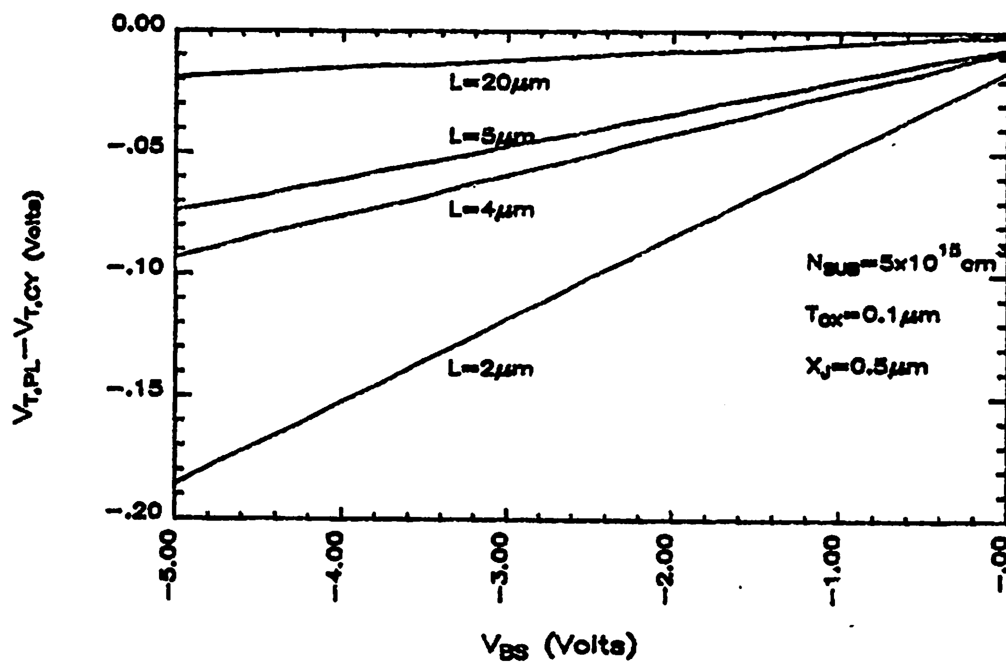


Figure 6.3 The V_{TH}'s of the Plane and Cylindrical Junction Approximations,

junction approximation [51] and the cylindrical-junction approximation [48] is plotted in Figure 6.3. This difference is important in the case of low V_{GS} operation in which the device characteristics are sensitive to the exact value of the threshold voltage. The difference increases as the deviation between the predicted depletion-region widths increases, i.e. when the substrate doping concentration is lower, the oxide thicker, and the junction made more shallow.

In the case of a shorter channel length, the source and drain depletion regions merge and the barrier maximum becomes a single point. The barrier height is lowered and can be further modulated by the drain voltage, as shown in Figure 5.2. An extra term which is linearly proportional to V_{DS} has been added to Equation (6.2). The proportional constant is inversely related to the oxide capacitance and the cube of the channel length [54]. Parameter η is designed to allow flexibility:

$$\Delta_L V_{TH} = \frac{\eta A}{C_{OX} L^3} V_{DS} \tag{6.4}$$

where A is an empirical constant whose value varies with the process parameters.

6.1.2. Narrow-Channel Effect

By assuming cylindrical distributions of the depletion charge at the edges, a correction factor can be formulated as:

$$\Delta_W V_{TH} = DELTA \frac{\pi \epsilon_{si}}{C_{OX} W} \left[\phi + V_{SB} \right] \tag{6.5}$$

Other edge effects, such as the existence of a field implant, non-planarity due to a LOCOS process, and the correction of the cylindrical field distribution, are included by introducing the empirical parameter, DELTA. This

parameter is characterized by threshold-voltage measurements.

6.1.3. The Model Equations

From the above, V_{TH} is formulated as:

$$V_{TH} = V_{FB} + \phi - F_D V_{DS} + \gamma F_S \sqrt{\phi - V_{BS}} + F_N \left[\phi - V_{BS}\right] \quad (6.6)$$

where F_D is the static-feedback coefficient:

$$F_D = \eta A \frac{L^{-3}}{C_{OX}} \quad (6.7)$$

A is an empirical constant:

$$A = 8.15 \times 10^{-22} \ (meterfarad) \quad (6.8)$$

and gamma is the body-effect coefficient without any correction:

$$\gamma = \frac{\sqrt{2\epsilon_{si} q N_{SUB}}}{C_{OX}} \quad (6.9)$$

F_S is the correction factor due to the short-channel effect as defined in Equation (6.3). L_D is the lateral-diffusion length, W_P the depletion-layer width of a plane junction, and W_C the depletion-layer width of a cylindrical junction and:

$$\frac{W_C}{x_J} = d_0 + d_1 \frac{W_P}{x_J} + d_2 \left[\frac{W_P}{x_J}\right]^2 \quad (6.10)$$

where d_0, d_1 and d_2 are empirical constants with the following values [48]: $d_0 = 0.0631353$, $d_1 = 0.8013292$, $d_2 = 0.01110777$. In Equation (6.6) F_N is the correction coefficient of narrow-channel effect:

$$F_N = DELTA \frac{\pi \epsilon_{si}}{C_{OX} W} \quad (6.11)$$

6.2. Basic Drain-Current Equation

The drain current can be expressed as:

$$I_{DS}^{o}(x) = WQ_{INV}(x)v(x) \tag{6.12}$$

where $N(x)$ is the carrier density per unit area at location x:

$$Q_{INV}(x) = C_{OX}\left[V_{GS} - V_{TH}(x)\right] \tag{6.13}$$

$v(x)$ is the carrier drift velocity:

$$v(x) = U\frac{dV(x)}{dx} \tag{6.14}$$

$V_{TH}(x)$ is the effective threshold voltage at location x:

$$V_{TH}(x) = V_{FB} + \phi(x) + \gamma\sqrt{\phi(x)} \tag{6.15}$$

Since the drain current is a constant independent of the location, Equation (6.12) can be integrated from the source to the drain to provide:

$$I_{DS}^{o} = \frac{W}{L}\int_{0}^{V_{DS}} Q_{INV}(x)dV(x) \tag{6.16}$$

The carrier density can be approximated by linearizing the expression of the effective threshold voltage with respect to the source:

$$V_{TH}(x) = V_{TH}(src) + \left[1 + F_B\right]V(x) \tag{6.17}$$

where

$$F_B = 0.5\frac{\gamma}{2\sqrt{\phi + V_{SB}}} \tag{6.18}$$

The 0.5 factor is used to count the effect of the other higher order terms which are dropped for simplicity. The expressions of the carrier and current densities become:

$$Q_{INV} = C_{OX}\left[V_{GS} - V_{TH} - \left[1 + F_B\right]V(x)\right] \tag{6.19}$$

$$I_{DS}^{o} = \frac{W}{L}U\,C_{OX}\left[V_{GS} - V_{TH} - \frac{\left[1 + F_B\right]}{2}V_{DS}\right]V_{DS} \tag{6.20}$$

A similar expression has been developed by other workers [55].

The above equation is an approximation of the drain-current expression which has been widely used in text books [31,56]:

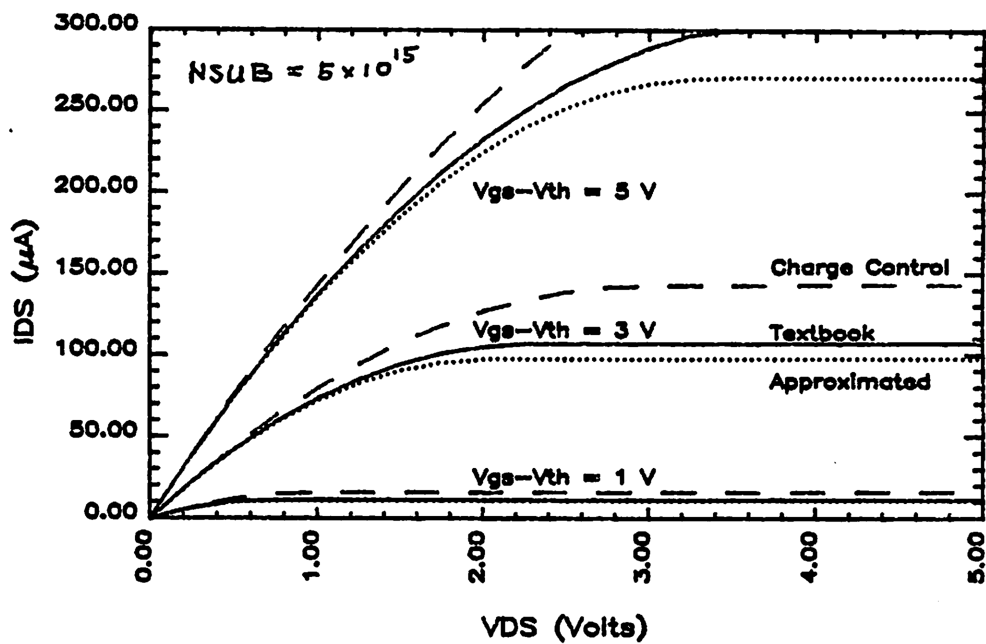
$$I_{DS}^{o} = \frac{W}{L} U\ C_{OX} \left[\left[V_{GS} - V_{FB} - \phi - \frac{V_{DS}}{2} \right] V_{DS} - \frac{2}{3}\gamma \left[\left[\phi + V_{DB} \right]^{\frac{2}{3}} - \left[\phi + V_{SB} \right]^{\frac{2}{3}} \right] \right] \quad (6.21)$$

These two equations differ in their linearization of the threshold-voltage expression. A comparison between Equations (6.20) and (6.21) is plotted in Figure 6.4 for the cases of $L = 5\mu m$, $W = 50\mu m$, $T_{OX} = 0.65\mu m$, $U_O = 600cm^2/V{-}sec$, $V_{TO} = 1V$ and $N_{SUB} = 5.0\times10^{15}cm^{-3}$ and $5.0\times10^{15}cm^{-3}$. A simpler expression based upon charge-control analysis [56] is also plotted in the same figure for comparison:

$$I_{DS}^{o} = \frac{W}{L} U\ C_{OX} \left[V_{GS} - V_{TH} - \frac{V_{DS}}{2} \right] V_{DS} \quad (6.22)$$

These three characteristics are approximately the same in the low V_{DS} range. The difference between the charge-controlled model and the text book model is much larger than that between the MOS3 and text book models. The characteristics at a higher V_{DS} are governed by second-order effects, such as the hot-electron effect, the current saturation, and the channel-length modulation. The basic current equation is critical only in the low V_{DS} operational range of a short-channel device. Thus Equation (6.20) can be applied to a small-geometry device without compromising accuracy.

The simplicity of Equation (6.20) leads to an explicit formulation of the saturation voltage which is derived later. Because the V_{DS} dependent term, η, in the threshold-voltage expression represents the average influence of the drain voltage upon the surface potential, it is treated as a constant throughout the integration in the derivation of current equation,



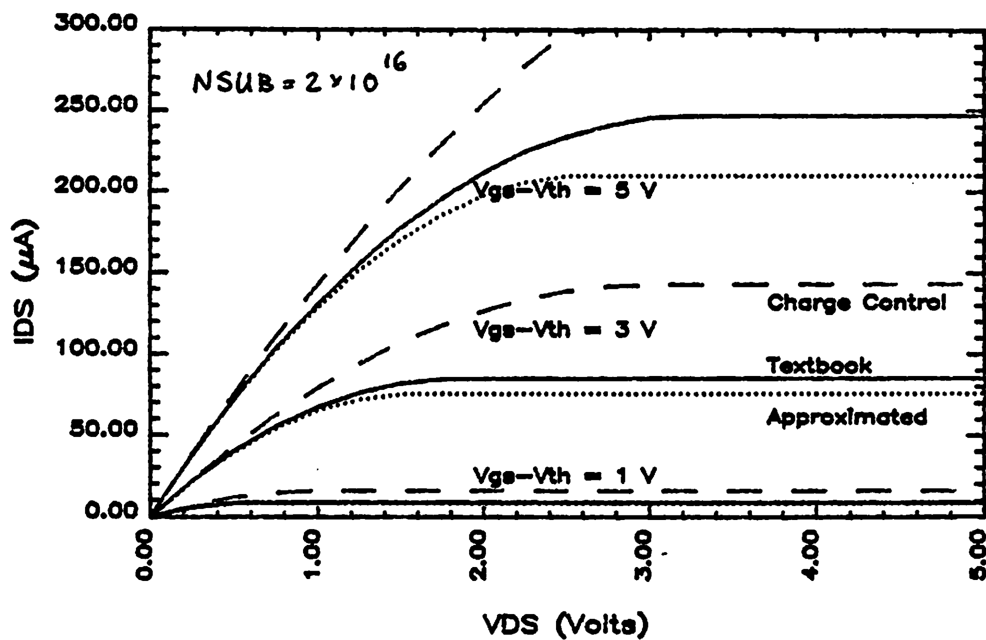


Figure 6.4 The Basic Current Equations with $N_{SUB} = 5.0\times10^{15}$ and $2.0\times10^{16} cm^{-3}$,

6.3. Surface-Mobility Modulation by Gate Voltage

Surface mobility is directly proportional to the channel conductance of a device. Besides the lattice scattering [57] and the impurity scattering [58] which determine the value of bulk mobility, the surface mobility is further degraded by the mechanism of surface scattering [59] and interband scattering [60]. Considerable effort has been devoted to theoretical and experimental studies of the surface mobility. Nonetheless, the use of an empirical expression is still the most practical approach for device modeling in CADs.

The empirical equation of U_S used in the MOS3 is:

$$U_S = \frac{U_O}{1 + \theta\left(V_{GS} - V_{TH}\right)} \tag{6.23}$$

For a comparison with the more elaborate formulation used in the MOS2 model in SPICE2 [61], the relationships between U_S and V_{GS} based upon these two expressions are plotted in Figure 6.5. These three plots show a very close match in the range of 10 V V_{DS}. Deviation is observed in both the low V_{DS}, 5 V, range, and the high V_{DS}, 20 V range. The plots demonstrate that the surface-mobility modulation effect can be matched by both empirical equations within a given operational range if the parameters are properly adjusted.

6.4. Velocity Saturation of Hot Electrons

Among various hot-electron effects [62], the saturation of hot-electron velocity has a direct impact on the characteristics of a short-channel device. It lowers the conduction current in the linear region and smooths the transition between the linear and saturation regions.

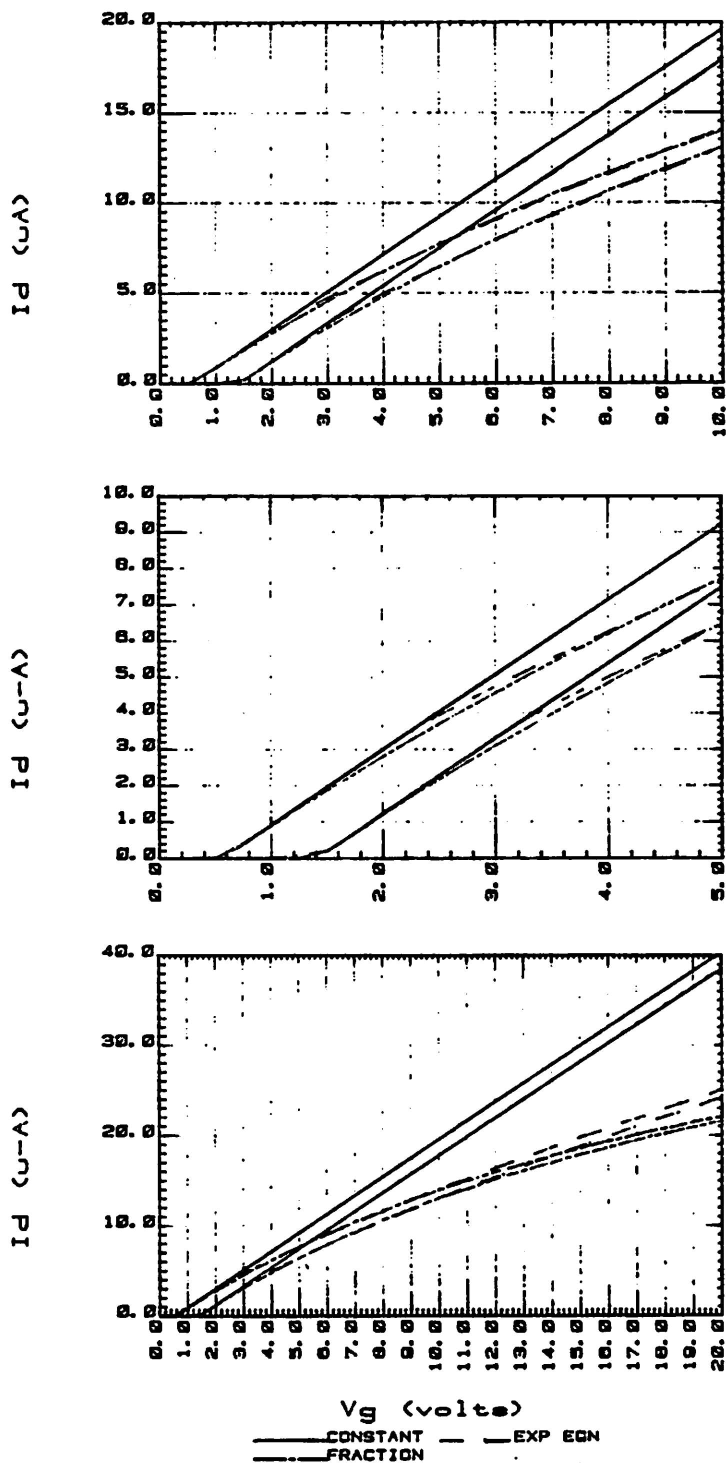


Figure 6.5 The Comparisons of Empirical Equations of Surface Mobility with Maximum $V_{DS} = 10V$, $5V$, and $20V$,

In a short-channel MOSFET, the lateral electric field in the channel is higher than that inside a long-channel device at the same operational bias. The effective mobility decreases and saturates when the electric field is stronger than the critical field. This relationship can be approximated by the following equation [63]:

$$U_{EFF}(x) = \frac{U_S}{1 + \frac{U_S}{V_{MAX}} \frac{dV}{dx}} \tag{6.24}$$

By substituting this expression in the basic current equation before carrying out the integration, the current equation becomes:

$$I_{DS} = \frac{I_{DS}^o}{1 + \frac{U_S}{V_{MAX}} \frac{V_{DS}}{L}} \tag{6.25}$$

The effective mobility can be expressed as:

$$U_{EFF} = \frac{U_S}{1 + \frac{U_S}{V_{MAX}} \frac{V_{DS}}{L}} \tag{6.26}$$

The effective mobility is plotted in Figure 6.6 as a function of V_{DS}. This effect has been interpreted as either a higher effective threshold voltage [64] or an effective feedback resistance [65].

6.5. Saturation Voltage

In a short-channel MOSFET the drain current saturates when the carrier velocity approaches its maximum [31], V_{MAX}, instead of approaching the channel pinch-off condition as it does in a long-channel devices. When the carrier velocity saturates, the current can be approximated as:

$$I_{DS}^o = Q_{INV}(drain) V_{MAX} \tag{6.27}$$

By substituting the expressions (6.19) and (6.20) in the above equation, it

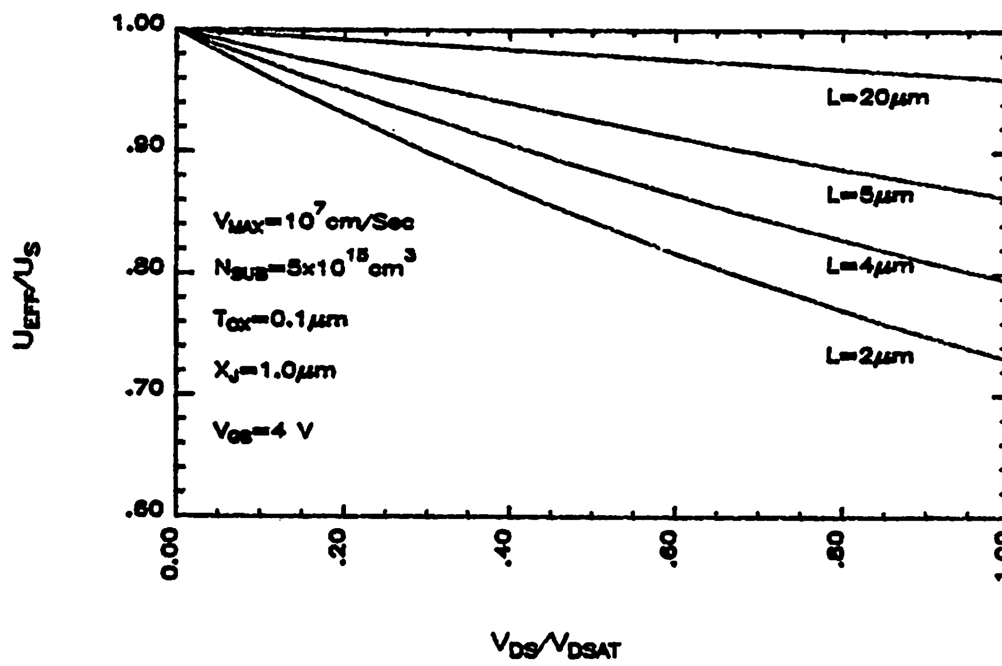


Figure 6.6 The Hot-Electron Effects on the Surface Mobility,

yields:

$$W C_{OX}\left[V_{GS} - V_{TH} - \left[1 + F_B\right]V_{DS,sat}\right]V_{MAX} = \frac{W}{L}\, U_S C_{OX}\left[V_{GS} - V_{TH} - \frac{1 + F_B}{2}V_{DS,sat}\right]V_{DS,sat} \tag{6.28}$$

$V_{DS,sat}$ is derived from the above equation:

$$V_{DS,sat} = \frac{V_{GS} - V_{TH}}{1 + F_B} + \frac{V_{MAX}L}{U_S} - \sqrt{\left(\frac{V_{GS} - V_{TH}}{1 + F_B}\right)^2 + \left(\frac{V_{MAX}L}{U_S}\right)^2} \tag{6.29}$$

$V_{DS,sat}$ is dependent upon V_{DS} through V_{TH}. At the limit of infinite V_{MAX}, this equation can be rearranged as follows:

$$V_{DS,sat} = \frac{V_{GS} - V_{TH}}{1 + F_B} + \frac{V_{MAX}L}{U_S}\left[1 - \sqrt{1 + \left(\frac{V_{GS} - V_{TH}}{1 + F_B}\,\frac{U_S}{V_{MAX}L}\right)^2}\right] \tag{6.30}$$

By only taking the first-order term in the Taylor series expansion, the term of square root can be simplified and the above equation becomes:

$$V_{DS,sat} = \frac{V_{GS} - V_{TH}}{1 + F_B} - \frac{U_S}{2V_{MAX}L}\left[\frac{V_{GS} - V_{TH}}{1 + F_B}\right]^2 \tag{6.31}$$

$V_{DS,sat}$ approaches to $\frac{V_{GS} - V_{TH}}{1 + F_B}$ which is the maxima of the current equation and corresponds to the channel pinch-off condition of a long-channel device.

In the derivation of $V_{DS,sat}$, Equation (6.20) is used instead of Equation (6.25). As pointed out in Murphy's work [64], if the exact expression is used, the velocity "pinch-down" condition is equivalent to the channel "pinch-off" condition. The resulting saturation voltage, $V_{DS,sat}{}^o$, is the maximum of the basic current equation in which the slope of I_{DS} is zero:

$$V_{DS,sat}{}^o = \frac{LV_{MAX}}{U_S}\left[\sqrt{1 + \frac{2U_S}{V_{MAX}L}\,\frac{V_{GS} - V_{TH}}{1 + F_B}} - 1\right] \tag{6.32}$$

$V_{DS,sat}{}^o$ is higher than $V_{DS,sat}$. From the view point of CAD applications, the zero-slope point is undesirable because the output conductance of a short-channel device is not zero. Equation (6.28) is equivalent to an approximation of the velocity-field relationship by two straight lines joined at the

saturation point, as shown in Figure 6.7. A comparison of saturation voltages based upon these two definitions is plotted in Figure 6.8 for the cases of $2\mu m$ and $5\mu m$ respectively. The discrepancy between $V_{DS,sat}$ and $V_{DS,sat}{}^{o}$ increases as V_{GS} increases.

6.6. Channel-Length Modulation

As V_{DS} gets larger than $V_{DS,sat}$, the point at which the carrier velocity begins to saturate moves toward the source, and the effective channel length is reduced. The formulation of the channel-length shortening factor, ΔL, is based upon Baum's theory [66] :

$$\Delta L = \sqrt{(\frac{E_P}{2B})^2 + \kappa(\frac{V_{DS} - V_{DS,sat}}{B})} - \frac{E_P}{2B} \tag{6.33}$$

$$I_{DS} = I_{DS,sat} \frac{L}{L - \Delta L} \tag{6.34}$$

where E_P is the lateral field at channel pinch-off point, and coefficient B is:

$$B = \frac{1}{X_D^2} \tag{6.35}$$

By making the slope of the I_{DS}-versus-V_{DS} characteristics continuous at $V_{DS} = V_{DS,sat}$, the expression E_P becomes:

$$E_P = \frac{I_{DS,sat}}{G_{DS,sat} L} \tag{6.36}$$

$I_{DS,sat}$ and $G_{DS,sat}$ are the drain current and the drain conductance at saturation voltage, respectively.

The point at which the velocity begins to saturate differs from the channel pinch-off point at which the free carriers begin to be depleted. $V_{DS,sat}$ is the voltage at the velocity-saturation point while E_P is the lateral field at the channel pinch-off point. Therefore, the voltage across the depleted surface should be less than $V_{DS} - V_{DS,sat}$. The empirical parameter κ is introduced to

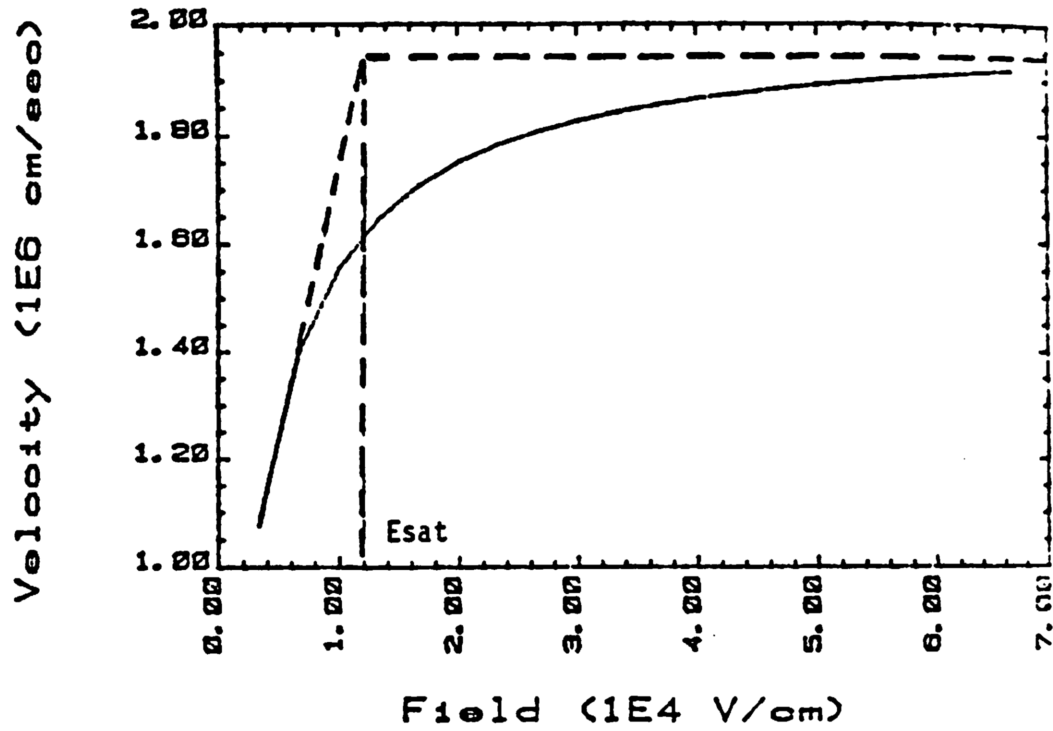
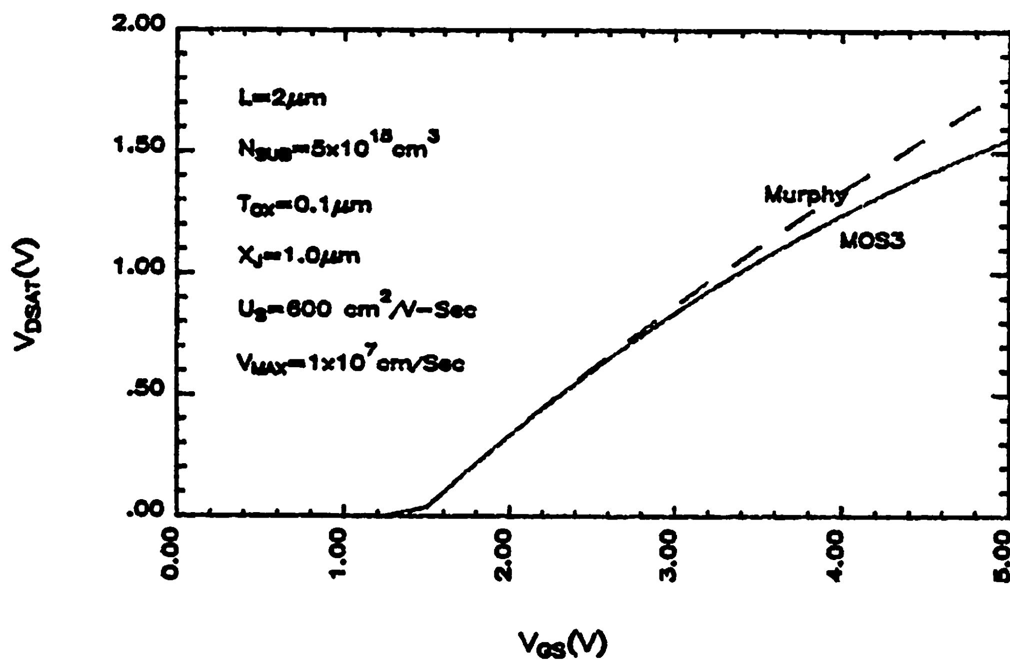


Figure 6.7 The Relationship between the Velocity and the Field Strength,



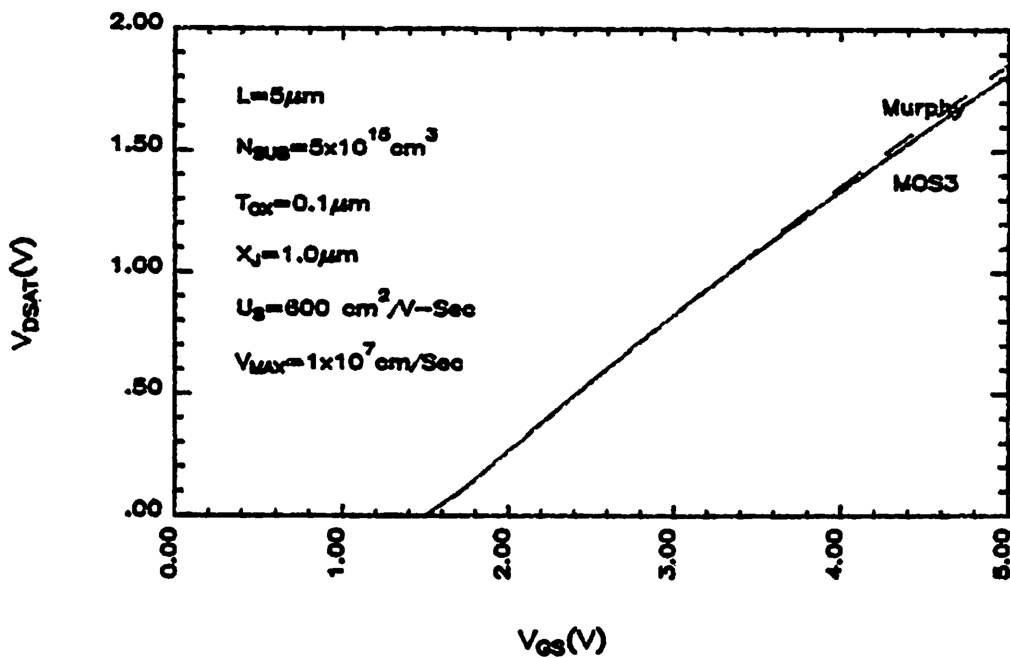


Figure 6.8 The $V_{DS,sat}$'s Based upon Different Definitions for the cases of $L = 2\mu m$ and $5\mu m$,

include this effect.

6.7. Capacitance Model with Charge Conservation

The companion capacitance model is based upon the charge-conservation concept [53] which is critical for the simulation of circuit operations depending upon charge transfer, for example, in switched capacitance circuits.

The total amount of charge residing on the gate can be formulated as:

$$Q_G = W \int_0^L Q_g(y)dy \tag{6.37}$$

$$= \frac{U_S W^2}{I_{DS}} \int_0^{V_{DS}} Q_g(V_y) Q_c(V_y) dV_y \tag{6.38}$$

where Q_g is the gate charge per unit area and Q_c the channel charge per unit area:

$$Q_g = C_{OX}\left[V_{GS} - \left[V_{FB} + \phi - F_D V_{DS}\right]\right] \tag{6.39}$$

$$Q_c = - C_{OX}\left[V_{GS} - V_{TH} - \left[1 + F_B\right]V_y\right] \tag{6.40}$$

The integration if carried out yields:

$$Q_G = W \times L \times C_{OX}\left[V_{GS} - \left[V_{FB} + \phi - F_D V_{DS}\right] - \frac{V_{DS}}{2} + \frac{1 + F_B}{12F_I}V_{DS}^2\right] \tag{6.41}$$

where

$$F_I = V_{GS} - V_{TH} - 1 + F_B/2V_{DS} \tag{6.42}$$

Similarly, the total bulk charge, Q_B, can be obtained:

$$Q_B = - W \times L \times C_{OX} Q_B^{o} \tag{6.43}$$

where

$$Q_B^{o} = \gamma F_S \sqrt{\phi + V_{SB}} + F_N\left(\phi + V_{SB}\right) + \frac{F_B}{2}V_{DS} - \frac{F_B(1 + F_B)}{12F_I}V_{DS}^2 \tag{6.44}$$

The charge-neutrality condition requires that the total channel charge be:

$$Q_C = -\left[Q_G + Q_B\right] \tag{6.45}$$

which is distributed between the source and the drain.

There are three charge quantities. Each of them has three associated derivatives which are the capacitive elements in the circuit model. Only six of these nine capacitance components are independent because Q_G, Q_B and Q_C are correlated with each other.

The resulting C-V characteristics, based upon the parameters listed in Table 6.1, is plotted in Figure 6.9, together with the corresponding C-V curves predicted by MOS2 [15] with the same parameters. Because of the linearization employed in MOS3, the results differ in the capacitive components which are related to the channel charge.

6.8. The Comparison Between MOS2 and MOS3

MOS3 is a semi-empirical model developed specifically for small-geometry devices. This section is intended to demonstrate the validity and performance of MOS3 through device characterizations and test-circuit simulations. The MOS2 model [15], which is based upon approximations for and analyses of devices with channel lengths greater than $2\mu m$, is used as the test vehicle for comparison. Both MOS2 and MOS3 are implemented in SPICE2G. Although they have many common parameters, different values of the mobility-related parameters must be used to produce approximately the same characteristics. For example, parameter VMAX (V_{MAX}) has no effect on the characteristics in the linear region simulated by MOS2, while it lowers the effective mobility in the medium V_{DS} range when simulated by MOS3. In order to get a close approximation, MOS2 requires a lower value

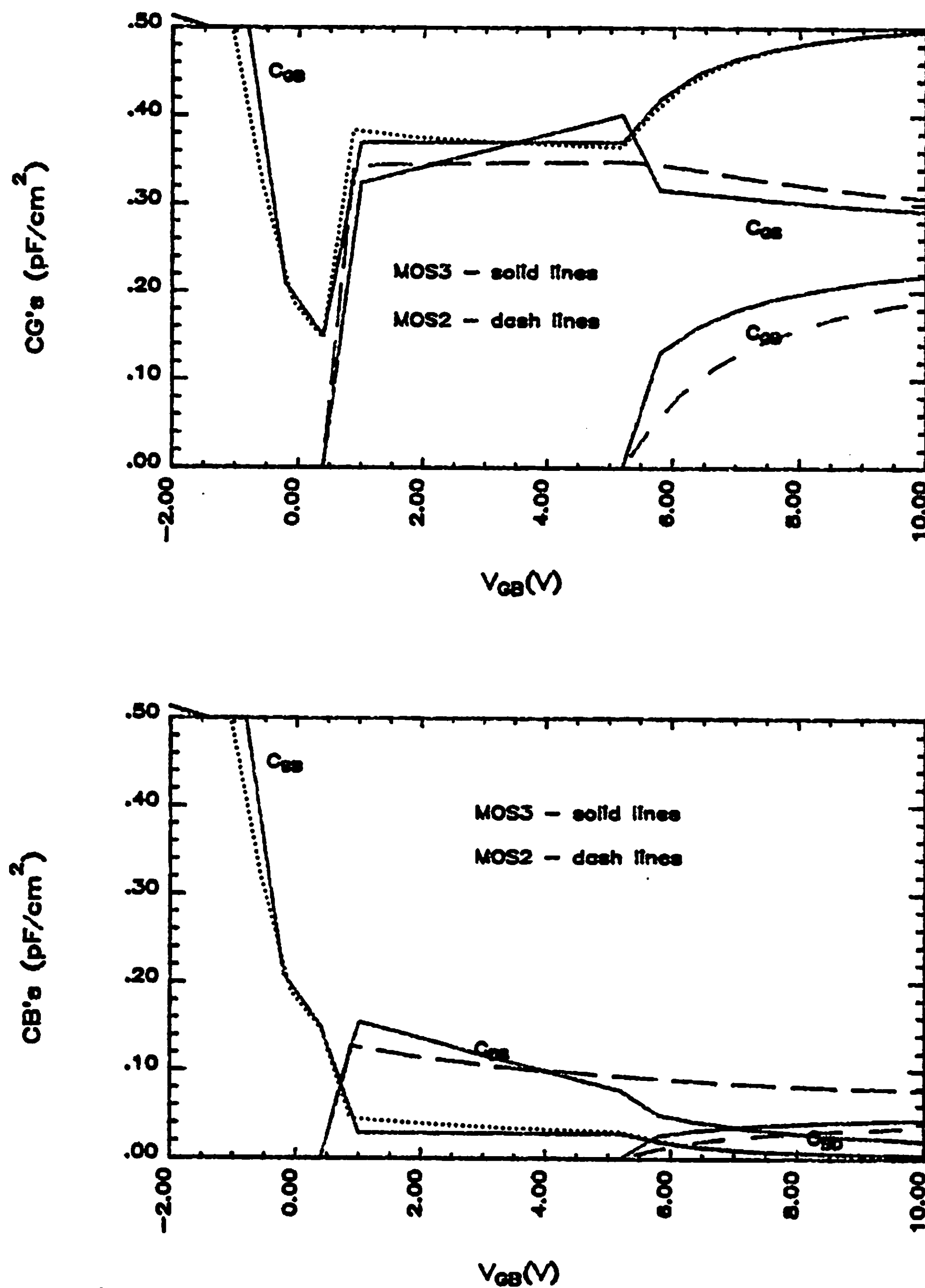


Figure 6.9 The Gate and Substrate Capacitances in MOS2 and MOS3,

of the parameter UO.

The differences between MOS2 and MOS3 are:

(a) The basic current equation used in MOS3 is an approximation of that of MOS2;

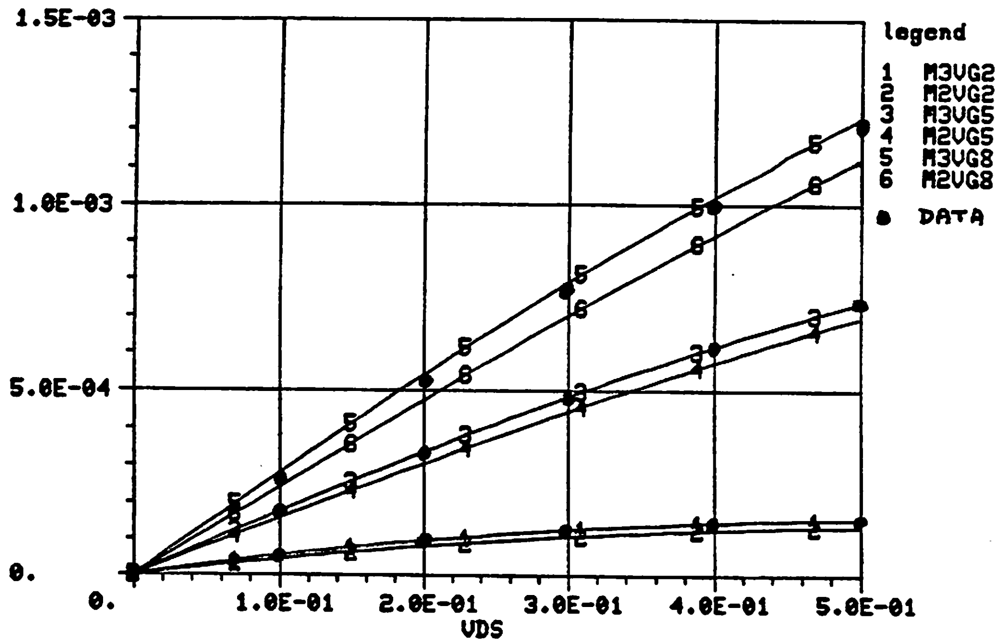
(b) Different empirical equations of surface-mobility modulation are used;

(c) The static-feedback effect is modeled by drain-induced barrier lowering in MOS3 and by charge sharing between the drain and the gate in MOS2;

(d) Parameter VMAX lowers both the effective mobility and the saturation voltage in MOS3 but affects only the saturation voltage in MOS2;

(e) The junction curvature effect is included in the threshold-voltage equation of MOS3 but the junctions are treated as plane junctions in MOS2.

Two devices are characterized by both MOS2 and MOS3 for comparison. One is long with a layout channel length of $20\mu m$; the other one is short with layout channel length of $2.3\mu m$, the effective channel length after side-diffusion correction is $1.6\mu m$. The two devices reside on the same chip and have the same width, $50\mu m$. Their simulated and measured characteristics are plotted in Figures 6.10 and 6.11 respectively, and the parameter values are listed in Table 6.1:



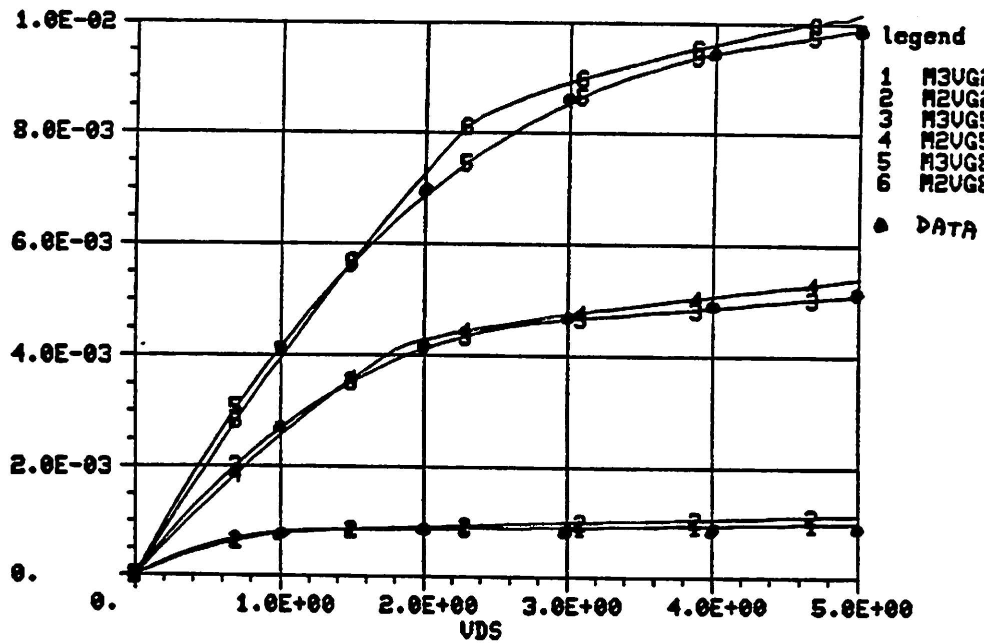
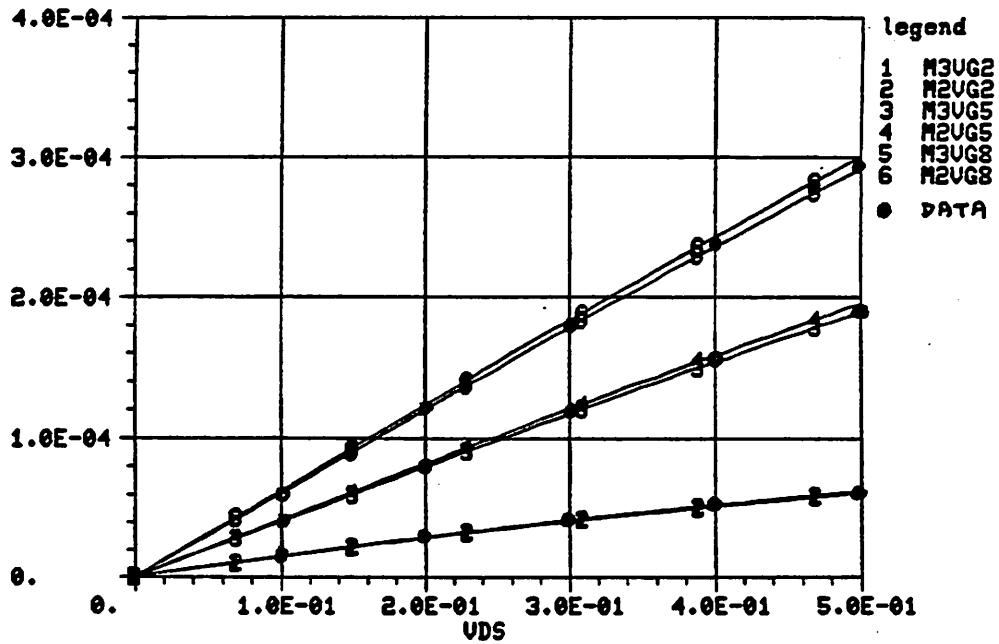


Figure 6.10The Experimental Measurements, the MOS2 and MOS3 Models in the Low and High V_{DS} Ranges of the Short-Channel Device, with $V_{BS} = 0V$,

50/20 DEVICE AT VBS=0



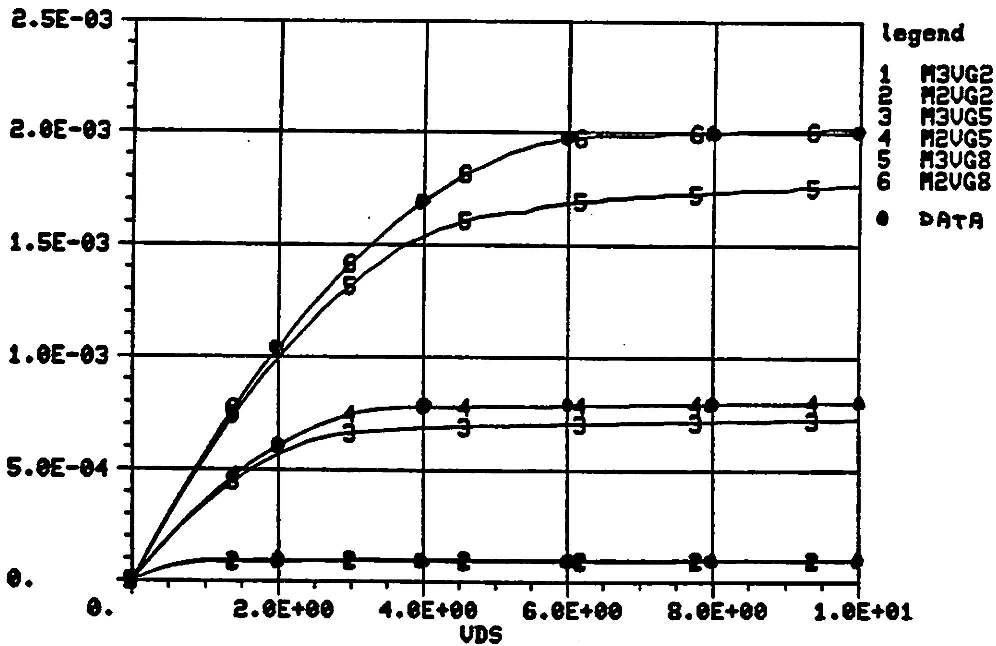


Figure 6.11The Experimental Measurements, the MOS2 and MOS3 Models in the Low and High V_{DS} Ranges of the Long-Channel Device, with $V_{BS} = 0V$,

Table 6.1				
Param.	Device No.1		Device No.2	
	MOS3	MOS2	MOS3	MOS2
W(um)	50	50	50	50
L(um)	2.3	2.3	20	20
XJ(um)	0.5	0.5	0.5	0.5
LD(um)	0.35	0.35	0.35	0.35
VTO(V)	0.452	0.452	0.452	0.452
TOX(um)	0.065	0.065	0.065 0.65	
NSUB(cm-3)	1.85E11	1.85E11	1.85E11	1.85E11
UO(cm2/V-s)	578	450	790	720
VMAX(M/s)	20E4	6E4	-	-
theta	0.06	-	0.045	-
eta	0.035	-	0	-
KAPPA	1.0	-	1.0	-
UEXP	-	0.24	-	0.20
UCRIT(V/M)	-	1.2E6	-	1.5E6
NEFF	-	7	-	1

Table 6.1 Device Model Parameters

All the process parameters except those related to mobility are the same. In the case of the short-channel device, MOS3 can fit both the high- and low-current ranges consistently while MOS2 can fit only the high-voltage range. MOS3 requires a higher UO and a higher VMAX, $\frac{V_{MAX}}{U_S}$, which is equivalent to a saturation field of $3.46\times10^4 V/cm$. This corresponds to the field in which the velocity begins to saturate. The values of VMAX and UO used in MOS2 yield a saturation field of $1.3\times10^4 V/cm$, which corresponds to the field at the corner of the velocity-saturation curve. This is the result of the different assumptions used in the models. The discrepancy between the predictions by the MOS3 model and the measurements of the long-channel device is

expected because the simulated characteristics of a long-channel device are dominated by the basic current equations. MOS2, whose basic current equation is based upon a more thorough analysis, is able to fit the long-channel device in both high and low current ranges consistently while MOS3 can fit only the low or high current range by using different mobility-related parameters.

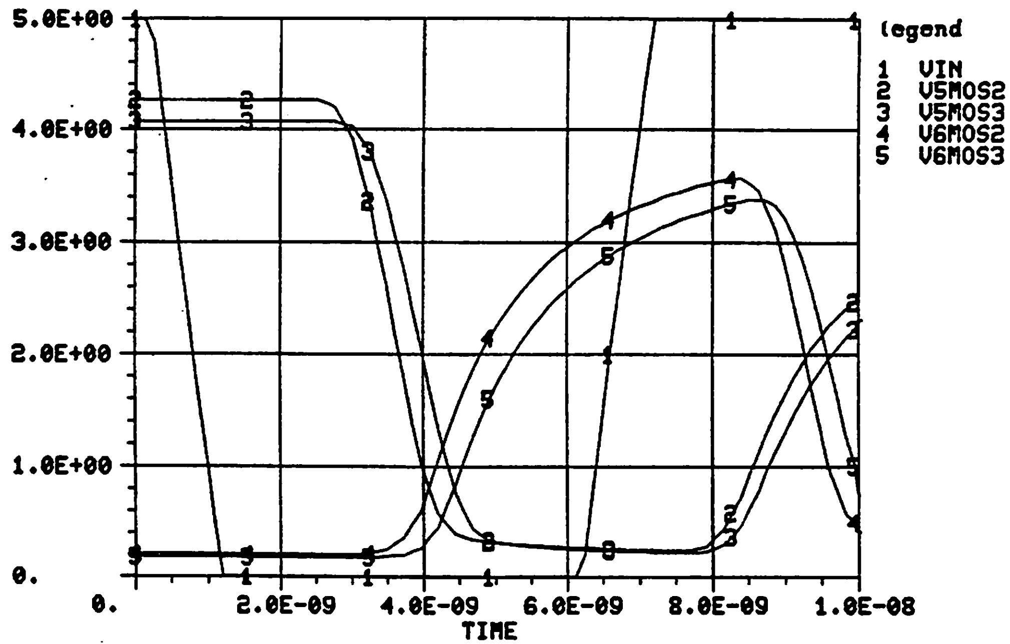
With devices of matched characteristics, several test circuits have been simulated by both models using SPICE2G.1. The inputs to SPICE2 are listed in Appendix D. The output waveforms are plotted in Figures 6.12(a) through (d). Even with the carefully chosen parameters, these two models do not give identical device characteristics, although the results are similar. The simulation statistics are compared in Table 6.2:

Table 6.2

Circuit Name	Analysis Type	X'tors MOS3	Iteration No. MOS2	MOS3	CPU (sec) MOS2	
Bootinv	Op Point	5	44	42	1.48	1.95
	Transient	-	235	280	9.54	15.77
Invchn	Op point	10	48	48	2.49	3.43
	Transient	-	208	306	14.94	26.69
Mosmem	Op point	12	164	34	9.25	3.21
	Transient	-	248	389	20.57	36.06
Ratlog	Op point	6	25	26	1.11	1.53
	Transient	-	778	648	36.47	40.13

Table 6.2 Simulation Statistics

The results show that for most of the circuits, the MOS3 model is up to 40% faster in computation than MOS2.



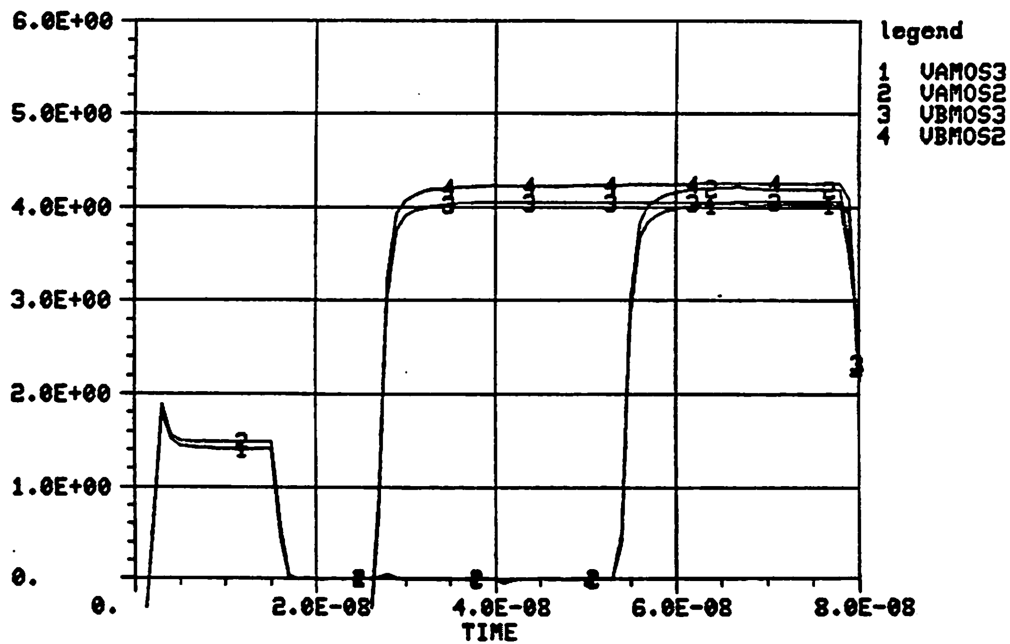
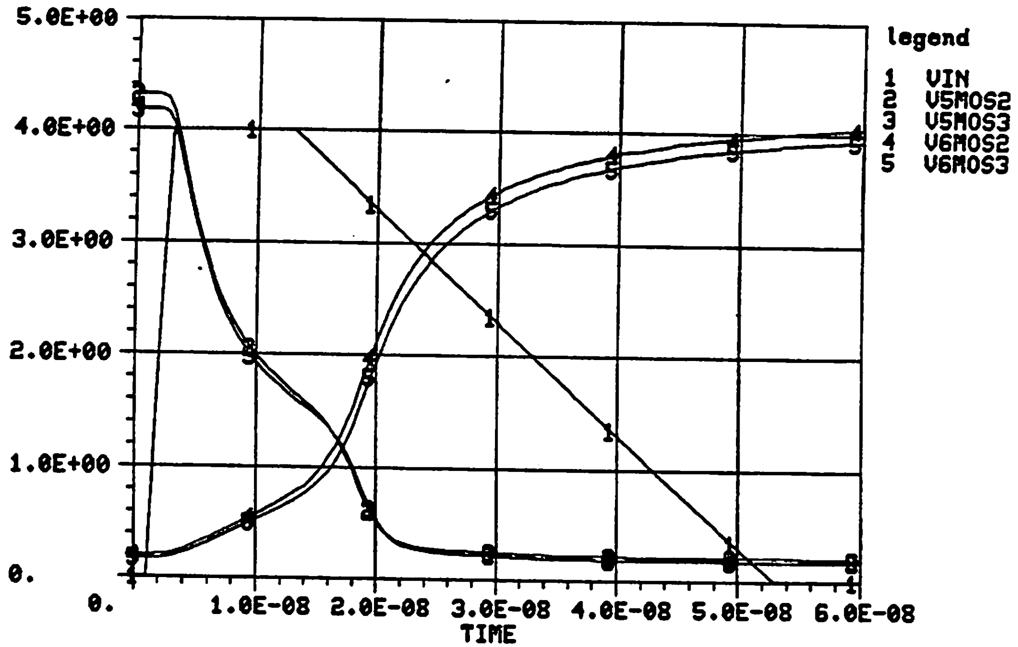


Figure 6.12The Simulation Results of Test Circuits using the MOS2 and MOS3 Models, (a) Five-Stage Inverter-Chain, (b) Ratioless Logic Circuit of Sift-Register,

(c) MOSMEM - 6-TRANSISTOR MEMORY CELL



(d) BOOTINV - BOOT STRAPPED DOUBLE INVERTER

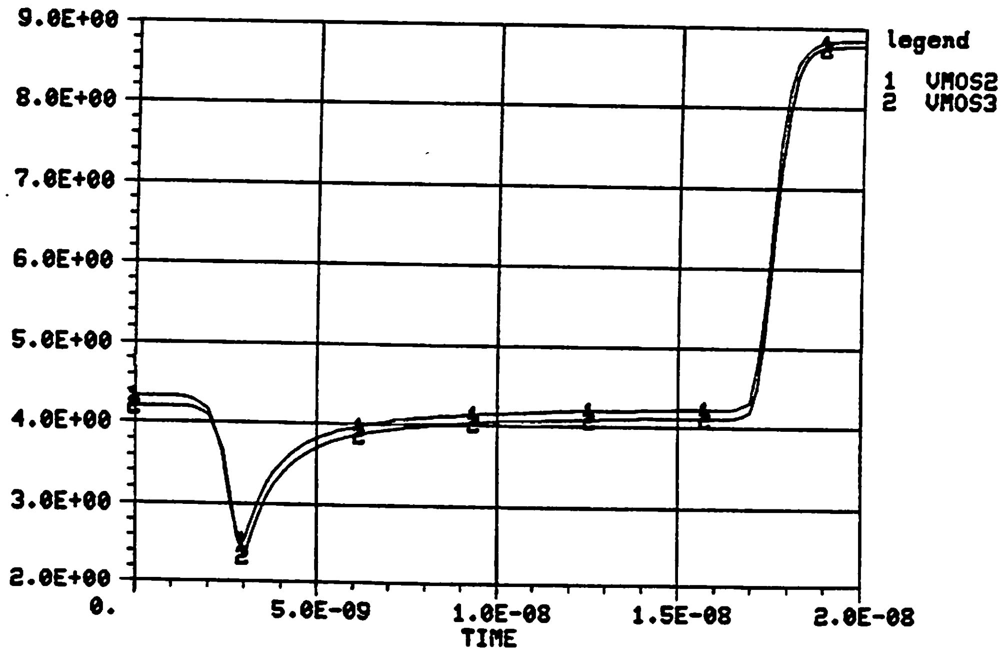


Figure 6.12 (c) Six-Transistor Memory Cell, (d) Boot-Strap Inverter,

CHAPTER 7

Summary

This analysis of MOSFETs emphasizes possible problems due to quantum-mechanical effects, the unification of strong- and weak-inversion regions and the modeling of small-geometry devices. The first two problem areas concern the modeling of MOSFETs in general. The modeling need for small-geometry devices is due to the recent advances in integrated-circuit processing which have led to today's VLSI chips.

The quantum-mechanical effects are attributed to both the degeneracy of the surface carrier population, which can be described only by Fermi-Dirac statistics, and the wave property of surface carriers in degeneracy, which is governed by the Schrödinger equation. Numerical evaluations of the drain current, the channel conductance, etc. based upon quantum-mechanical statistics, demonstrate that quantum-mechanical statistics alone do not result in a significant deviation in device characteristics within a practical operational voltage range. The onset voltage of degeneracy is at the high end of the voltage range of most practical applications. Though the differences in device characteristics induced by the wave property may be large, these differences can be absorbed in the empirical expression of surface mobility, whose variations are much greater in the practical operational range.

Existing MOSFET models which are valid for both the strong- and weak-inversion regions require time-consuming iterative solutions. The other models are valid in either strong- or weak-inversion region. The approach presented in Chapter 3 joins the weak- and strong-inversion regions by recognizing the existence of a transition region. It proves to be an

efficient approximation of the iterative solution.

Program TWIST was developed to simulate the characteristics of weak inversion and weak-injection punchthrough of short-channel MOS devices by solving the two-dimensional Poisson equation. The program is sufficiently fast in its analysis to allow reasonable interaction with a process or device designer as the simulation is performed. In a working hierarchy of CAD tools, structural and impurity parameters can be obtained from process simulators. TWIST can then be used to optimize all the aspects of barrier-controlled operations and as a pre-selector for structures to be simulated by a more elaborate two-dimensional simulation program to obtain high-current device characteristics.

The analysis of the punchthrough phenomena involves both theoretical analyses and two-dimensional device simulations. The formation and characteristics of the injection barrier are studied and the equation of the onset voltage of punchthrough is derived assuming a uniform substrate doping concentration. The experimental data supports the derived equation.

The MOS3 model has been developed and implemented in the circuit simulation program SPICE2 to address the features of small-geometry MOSFETs and to permit the effective simulations of integrated circuits containing small-geometry MOSFETs. The model equations are formulated to allow easy and automatic parameter extraction, a property which is as critical as the accuracy of the model itself. A comparison of the MOS2 and MOS3 models proves that the MOS3 model is accurate for small-geometry MOSFETs.

In the era of LSI and VLSI, the emphasis of modeling should be placed upon the small-geometry devices. The two-dimensional device simulation is an indispensable tool for the study of micron or submicron devices. In the course of expanding Program TWIST to include the solution of the current-

continuity equation, the attention must be put on finding and implementing both efficient algorithms of numerical solutions and adequate physical models of various high-current effects. The impurity profile in a small-geometry device critically affects device characteristics. The thermal redistribution in both one and two dimensions, must be considered in the generation of impurity profile. This profile dependence will be an important part of the future study of both two-dimensional device simulation and circuit-simulator oriented models. Although the onset of punchthrough is described in this thesis, a complete model of punchthrough conduction remains to be formulated. More research is needed in this area.

APPENDIX A

TWIST User's Guide

TWIST is a program for the "Two-dimensional Interactive Simulation of MOS Transistors" in weak inversion and/or weak injection. The device geometry and doping profile as well can be entered either through the console or a parameter file. The doping profile can be defined analytically by specifying process parameters or numerically by using the results from Program SUPREM. The resulting impurity concentration, carrier concentration, potential, and field distributions are either displayed as three-dimensional graphs or output as numerical tables. The interactive feature of TWIST allows maximum flexibility to the user.

Except for numerical parameters, which include the values of device dimension, impurity concentration, voltage, etc., soft (special purpose) keys on the keyboard are used to facilitate the question-and-answer session. Keys No.7 and No.8 are always designated as Yes and No, respectively. The definitions of other keys are displayed with the accompany questions. If an answer is entered from the keyboard, instead of the soft keys, the first character of the alphabetical answers must be in the upper case.

In the following, the procedure of using TWIST is explained step by step. All the information displayed by TWIST is shown in *italics* in the same sequence as prompted by TWIST. f7 and f8 are the abbreviations for keys No.7 and No.8 respectively.

1. Input Options

REQUEST->

Input from the console (Yes=f7/No=f8) ?

Yes-> TWIST will ask for the name of the parameter file, read the geometry and profile parameters from it, and branch to Section 5 if SUPREM results are used; otherwise it will proceed to Section 6. The format of the parameter file is detailed at the end of this guide.

No-> Parameters will be requested on the console.

2. Geometry Parameters

REQUEST->

Drawn channel length (μm) ?
Lateral span of source (μm) ?
Lateral span of drain (μm) ?
Oxide thickness: gate <> and field <> (μm) ?
Gate oxide location: from <> to <> (μm) ?
Span of oxide ramp (μm) ?
Drawn gate location: from <> to <> (μm) ?
Depth of the simulated structure (μm) ?

All the entries should be in units of μm. The source junction is defined as the origin of the coordinates.

3. Profile Options

REQUEST->

"Use SUPREM output (Yes=f7/No=f8) ?"

Yes-> The profile is generated by the interpolation of SUPREM results.

No-> Parameters of analytically generated profiles are entered from the console.

4. Profile Parameters

The profile can be tailored by up to three implantation steps, i.e. overall, selective and source/drain implants. The overall implantation covers the entire structure; the selective implantation can be directed into either a specified window or over the entire structure; the source/drain implantation goes only into the source/drain windows. The two-dimensional redistribution of drive-in is considered only for the selective and the source/drain implantations.

4.1. Substrate

REQUEST->

Substrate dopant ?
Substrate concentration (in unit of 1E15 cm-3) ?

4.2. Ion Implantation

REQUEST->

Any overall implant (Yes=f7/No=f8) ?
Any localized implant (Yes=f7/No=f8) ?
Any source/drain implant (Yes=f7/No=f8) ?

Yes-> Invoke the corresponding ion-implantation step.

No-> Skip the corresponding ion-implantation step.

The implant dopant can be entered either by the species (B/As/Ph/Sb), or by the type (-/+), where - stands for N-type and + for P-type. If a species is used, the associated drive-in process is characterized by the temperature and the time. Otherwise, the drive-in process is characterized by the diffusivity and the time.

REQUEST->

Implant parameters: Range(μm) <> Stndv(μm) <> Dose(cm-2) ?
Diffusion coefficient at drive-in temperature (cm2/sec) ? (optional)
Drive-in temperature ($^{\circ}C$) ? (optional)
Drive-in time (minutes) ?

A table of computed parameters is displayed at this point. For example,

...profile parameters:
diff const=4.75E-15(cm2/sec) peak conc=-3.85E+15(cm-3)
jct depth=2.19E-05(cm) average conc=-4.58E+15(cm-3)

where jct is the abbreviation of junction, diff of diffusion, and conc of concentration.

5. Profile Based on SUPREM Results

5.1. Substrate

REQUEST->

Substrate dopant ?
Uniform substrate (Yes=f7/No=f8) ?

Yes-> Substrate concentration will be requested.

Substrate concentration (in unit of 1E15 cm-3) ?

No-> File name and column index will be requested and the substrate profile is treated as the result of the overall implantation (see next section).

Data file name ?
Which column ?

The background concentration will be extracted and displayed:

Non-uniform substrate with background concentration = -7.500E+14cm-3

5.2. Ion-Implantation

REQUEST->

Implant dopant ?
Implant STNDV(μm) ?
Data file name ?
Which column ?

A table of the estimated profile parameters is displayed at this point.

```
...profile parameters:
  total dose=4.36E+15(cm-3)   standard D= 9.85E-06(cm)
  peak  conc=1.76E+20(cm-3)   impl range= 1.00E-06(cm)
  ave   conc=8.72E+19(cm-3)   jct  depth= 5.00E-05(cm)
```

where ave is the abbreviation of average, impl of implantation, D of deviation, jct of junction and conc of concentration.

6. Lateral Diffusion

After the doping profile is defined, the effect of lateral diffusion is displayed as:

```
Drawn source/drain junctions at ( 0.000um, 2.340um).....( 4, 42)
Corrected by side diffusions as (  .400um, 1.940um).....( 15, 31)
Lateral diffusion length of s/d:(  .400um,  .400um)
Effective channel length: 1.541um
```

7. Check Mesh and Profile, Save Parameters

After the setup, the mesh and the doping profile can be examined by answering Yes to the requests.

REQUEST->

Check impurity profile (Yes=f7/No=f8) ?
Check the mesh (Yes=f7/No=f8) ?

The other related information will be requested as explained in Section 9. The input parameters entered from the keyboard can be saved on the disc for repeated use by answering Yes to the request and specifying the file name to be used.

REQUEST->

Save input parameters (Yes=f7/No=f8) ?
Data file name ?

8. Potential Initialization and Self-Consistent Solution

REQUEST->

Applied voltages: VD, VG, VS, VB ?
Absolute resolution of 1-D iteration (mVs) ?
Absolute resolution of 2-D solution (mVs) ?
Relaxation factor (1<=x<2, ~1.7) ?
Maximum count of 2-D iterations ?
Convergence information per 2-D iteration (Yes=f7/No=f8) ?
Search for specific surface potential (Yes=f7/No=f8) ?

The suggested resolution of initialization is approximately from 0.2 to 0.01 mV. The relaxation factor should be equal to or greater than one, but less than and not equal to 2. Numbers close to 1.7 have been proved adequate. The optimal value varies depending upon the structure and the bias. If searching a specific surface potential is desired, the following question will be asked:

REQUEST->

Target surface potential value ?
Iterate which bias (D=Vd,G=Vg,S=Vs,B=Vb) ?
Searching tolerance (mVs) ?

A summary of the initialization is displayed. For example:

```
Initialize column  1: Converged at 26th iteration, maximum deviation =  0.00
Initialize column 19: Converged at 50th iteration, maximum deviation =  .190E-03
Initialize column 50: Converged at 50th iteration, maximum deviation =  .827E-04
Equal potential region between  -.05um( 2) and   .09um(11) as  -.10um( 1)
Equal potential region between  .91um(35) and  1.24um(49) as  1.30um(50)
Lateral depletion layer between   .30um(19) and   .91um(35)
Lateral depletion layer between   .09um(11) and   .61um(25)
```

The two-dimensional iteration either converges or is limited by the given count. The convergence message of each iteration loop can be turned on or off as desired. The message assumes the following format:

10th loop: max deviation =2.866E-02, at (35,12), by-pass 42.79%

A summary is displayed at the end of the two-dimensional iteration:

```
2-D iteration stops at loop 85:
   last max deviation =2.158E-04, at (44,18),
   by pass 55.80% ave per loop
**At surface**
   Potential minimum =   .199at X =  .920um, (23)
   Barrier   height  =   .683
   Current   density = 1.245E-08 Amp/cm2
   Barrier   width   =  .654um,
          from   .554um to  1.208um...(19,26)
   Source depletion width =  .133um
   Drain  depletion width =  .210um
Saddle potential =  .200, barrier height= -.682
          at (  .920um,  .130um)...(23, 6)
   current density = 2.014E-08 Amp/cm2
   Barrier width =  .404um,
          from   .718um to  1.122um...(21,25)
```

The result can be checked at this point by answering Yes to the following request.

REQUEST->

Check results (Yes=f7/No=f8) ?

The procedure is explained in the next section.

9. Output

The results may be displayed and examined at each check point, i.e. after the setup and 2-D iteration steps.

REQUEST->

Which one? 2-dimensional plots/f1, 3-dimensional plots/f2,
save in Fmgr file/f3, print the numbers/f4

After the choice is made, the domain of display will be requested. For a 2-D display, the user has to select an X or Y cross section and define the domain.

REQUEST->

Constant X or Y ?
Cross section index ?
From <> to <> (indices) ?

The functional values in the defined domain will be displayed to help the user estimate the minimum and the maximum. For the other choices of display, the messages are:

REQUEST->

From <> to <> (X-direction) ?
From <> to <> (Y-direction) ?

At this point, a 7x8 table is displayed to help the user determine the appropriate minimum and maximum. Then the following question is prompted:

REQUEST->

Estimated MIN/MAX function values (MIN=>MAX->skip) ?

If MIN is greater than or equal to MAX, the output will be skipped.

9.1. 3-D Graphics Display (f2)

REQUEST->

On the console (Yes=f7/No=f8) ?
Log scale (Yes=f7/No=f8) ?
How many points in X-direction ?
How many points in Y-direction ?
Tilt angle (degree) ?
Rotation angle (degree) ?

After the results have been plotted on the screen, graphs may be re-drawn on the plotter.

9.2. Numerical Display (f3,f4)

The results may be routed to the printer or saved on the disc.

9.3. More Outputs

Other data may be obtained at each check point. The user may have more than one output. The output alternatives are determined by the answer to the following request.

REQUEST->

Which one? doping concentration/f1, free carrier profile/f2,
field distribution/f3, potential profile/f4.

Except for the choice of displaying doping profile, the signs of the results will be changed if the answer to the following question is Yes.

REQUEST->

Referring to electron (Yes=f7/No=f8) ?

There are four field-display options.

REQUEST->

Which one? X-component(f1), Y-component(f2), X/Y-ratio(f3) or magnitude(f4).

10. Loops

The simulation can be repeated at a different bias without re-defining the geometry and the profile. The user can also analyze a new structure using a profile defined by the same process parameters or analyze the old structure using different process parameters.

REQUEST->

Another bias (Yes=f7/No=f8) ?
Another run (Yes=f7/No=f8) ?
Redefine the structure (Yes=f7/No=f8) ?
Redefine the profile (Yes=f7/No=f8) ?

11. Open File Error

If the specified disc file cannot be opened, the user can try again or exit.

12. Input File Format

12.1. Case of Analytically Generated Profile

line.1: title line
line.2: drawn channel length (μm)
line.3: lateral span of source (μm)
line.4: lateral span of drain (μm)
line.5: oxide thickness: thin? and thick ? (μm)
line.6: thin-oxide location: from ? to ? (μm)
line.7: lateral span of oxide ramp (μm)
line.8: drawn gate location: from ? to ? (μm)
line.9: depth of the simulated structure? (μm)
line.10: substrate dopant: B, As, Ph, Sb, +(n-type), -(p-type)
line.11: substrate doping concentration (*1E15 cm-3)
line.12: well implant dopant: B, As, Ph, Sb, +, -
line.13: well implant range(μm), stndev(μm) and dose (cm-2)
line.14: diffusion constant of well implant (cm2/sec)
line.15: drive in temperature for well implant ($^{\circ}C$)
line.16: drive in time for well implant (min)
line.17: localized implant location: from ? to ? (μm)
line.18: local implant dopant: B, As, Ph, Sb, +, -
line.19: local implant range(μm), stndev(μm) and dose (cm-2)
line.20: diffusion constant of selective implant (cm2/sec)
line.21: drive in temperature for selective implant ($^{\circ}C$)

line.22: drive in time for selective implant (min)

line.23: source/drain implant dopant: B, As, Ph, Sb, +, -

line.24: src/drn implant range(μm), stndev(μm) and dose (cm-2)

line.25: diffusion constant of src/drn implant (cm2/sec)

line.26: drive in temperature for src/drn implant (^oC)

line.27: drive in time for src/drn implant (min)")

12.2. Case of SUPREM generated profile

line.1: title line

line.2: drawn channel length (μm)

line.3: lateral span of source (μm)

line.4: lateral span of drain (μm)

line.5: oxide thickness: thin? and thick ? (μm)

line.6: thin oxide location: from ? to ? (μm)

line.7: lateral span of oxide ramp (μm)

line.8: drawn gate location: from ? to ? (μm)

line.9: depth of the simulated structure? (μm)

line.10: index of SUPREM input F

line.11: substrate dopant: B, As, Ph, Sb, +(n-type), -(p-type),

line.12: index of non-uniform substrate N

line.12.a: if line 12 is not N in 1st column:
substrate doping concentration (*1E15 cm-3)

line.12.b: well implant dopant: B, As, Ph, Sb, +, -

line.13: standard deviation(μm) of well implant

line.14: SUPREM save file name of well implant

line.15: column index of well implant

line.16: localized implant location: from ? to ? (μm)

line.17: local implant dopant: B, As, Ph, Sb, +, -

line.18: local implant standard deviation(μm)

line.19: local implant file name

line.20: local implant column index

line.21: source/drain implant dopant: B, As, Ph, Sb, +, -

line.22: src/drn implant standard deviation(μm)

line.23: src/drn implant file name

line.24: src/drn implant column index

APPENDIX B

Example Input to Program TWIST

1. Input to TWIST

```
******************************  TWIST  ********************************
1.84                 ..drawn channel length (μm)
0.184                ..lateral span of source (μm)
0.368                ..lateral span of drain  (μm)
6.50E-02   0.60      ..oxide thickness: thin? and thick ? (μm)
0.0      1.84        ..thin oxide location: from ? to ? (μm)
0   .0               ..lateral span of oxide ramp (μm)
0.0      1.84        ..drawn gate location: from ? to ? (μm)
4.5                  ..depth of the simulated structure? (μm)
F                    ..SUPREM input index
B                    ..subs dopant: B,As,Ph,Sb,+,-
U                    ..index of non-uniform substrate N
0.75                 ..subsrate concentration (*10^15 cm^-3)
$                    ..well implant dopant
0                    ..standard deviation(μm) of well implant
xxxxx1::s3           ..SUPREM save file name of well implant
1                    ..column index of well implant
9 -9                 ..localized implant location: from/to(μm)
$                    ..local implant dopant: B, As, Ph, Sb, +, -
0                    ..local implant standard deviation(μm)
0                    ..local implant file name
0                    ..local implant column index
```

```
As              ..src/drn implant dopant: B,As,Ph,Sb,+,-
0.02            ..src/drn implant standard deviation(μm)
suprm2::s3      ..src/drn implant file name
2               ..src/drn implant column index
```

2. Input to SUPREM for Generating Example Profile

```
title 2706 enhancement, check source/drain junction
global levl=2
subs  ornt=100, elem=b,      conc=7.5e14
grid  dysi=0.01, dpth=0.75,  ymax=1.0
print head=n,  idiv=n,       totl=n
plot  idiv=n,  totl=n
step  type=oxid, time=30,    temp=850,  trte=10,     modl=nit0
step  type=oxid, time=4,     temp=1150, modl=dry5
step  type=oxid, time=70,    temp=1150, trte=-4.286, modl=nit0
step  type=impl, elem=b,     dose=4e11, akev=100
print head=y,  idiv=n,       totl=n
plot  wind=0.8, cmin=14,     ndec=8,   totl=n,      idiv=y
step  type=impl, elem=as,    dose=1e16, akev=100
step  type=oxid, time=25,    temp=900,  modl=nit0
print head=y,  idiv=y,       totl=n
step  type=oxid, time=75,    temp=1000, modl=nit0
model name=dry5, lrte=2.5e5, lrea=2.0, prte=52.0, prea=1.23
save  lunm=20,  type=ascii
end
```

APPENDIX C

Example Console Session of Program TWIST

Input from the console (Yes=f7/No=f8)? "No"
Input data file name? "@input::xx"
*************************** Input Summary ***************************
drawn channel length 1.84um
lateral span of source .18um
lateral span of drain .37um
oxide thickness: thin: .06um, thick: .60um
thin oxide loc: from 0.00um to 1.84um
lateral oxide ramp 0.00um
drawn gate loc: from 0.00um to 1.84um
depth of the structure 4.50um
Use SUPREM generated profile.
Uniform substrate of dopant: B, concentration=-7.500E+14cm-3
..... profile parameters:
total dose = 4.36E+15(cm-3) standard D = 9.85E-06(cm)
peak conc = 1.76E+20(cm-3) impl range = 1.00E-06(cm)
ave conc = 8.72E+19(cm-3) jct depth = 5.00E-05(cm)
Drawn source/drain junctions at (0.00um, 1.84um).....(4, 42)
Corrected by side diffusions as (.42um, 1.42um).....(17, 29)
Lateral diffusion length of s/d:(.42um, .42um)
Effective channel length: 1.00um
Check impurity profile? (Yes=f7/No=f8) "No"
Check the mesh? (Yes=f7/No=f8) "No"
Save input parameters? (Yes=f7/No=f8) "No"

User's responses are in quotes.

```
Applied voltages: Vd, Vg, Vs, Vb ? "1 .3 0 0"
Absolute resolution of 1-D iteration (mVs) ? ".1"
Absolute resolution of 2-D solution (mVs) ? ".5"
Relaxation factor (1<=x<2, ~1.7) ? "1.7"
Maximum count of 2-D iterations ? "300"
Convergence information per 2-D iteration (Yes=f7/No=f8) ? "No"
Search for specific surface potential (Yes=f7/No=f8) ? "No"
Initialize column  1:Converged at 17th iteration, maximum deviation =  0.00
Initialize column 18:Converged at 12th iteration, maximum deviation =  0.00
Initialize column 49:Converged at 28th iteration, maximum deviation =  0.00
Equal  potential region between  -.12um( 2) and  .37um(16) as  -.18um( 1)
Equal  potential region between  1.47um(30) and  2.14um(48) as  2.21um(49)
Lateral depletion layer between  .48um(18) and  1.47um(30)
Lateral depletion layer between  .37um(16) and  1.36um(28)
************************   End of Initialization   ************************
**2-D iteration stops at loop  81:
   last max deviation =9.803E-04, at (24,17), by pass 62.25% ave per loop
**At surface**
   Potential minimum =   .551, at X =  .632, (20)
   Barrier   height  =  .331
   Current   density = 2.077E-08 Amp/cm2
   Barrier   width   =  .392um, from   .422 to  .814...(17,22)
   Barrier   depth   =  .859um, from  0.000 to  .859...( 1,22)
   Current / width   =  8.27    Amp/um, with Uso=700.0 cm2/sec-V
   Source depletion width = 0.000um
   Drain  depletion width =  .605um
Check results (Yes=f7/No=f8) ? "No"
More iterations (Yes=f7/No=f8) ? "No"
******************   End of Two Dimensional Solution   ******************
```

Another bias point (Yes=f7/No=f8) ? "Yes"

Applied voltages: Vd, Vg, Vs, Vb ? "2 .3 0 0"

Re-initialize the potential (Yes=f7/No=f8) ? "Yes"

With same iteration parameters (Yes=f7/No=f6) ? "Yes"

Search for specific surface potential (Yes=f7/No=f8) ? "No"

Initialize column 1:Converged at 17th iteration, maximum deviation = 0.00

Initialize column 18:Converged at 12th iteration, maximum deviation = 0.00

Initialize column 49:Converged at 35th iteration, maximum deviation = 0.00

Equal potential region between -.12um(2) and .37um(16) as -.18um(1)

Equal potential region between 1.47um(30) and 2.14um(48) as 2.21um(49)

Lateral depletion layer between .48um(18) and 1.47um(30)

Lateral depletion layer between .37um(16) and 1.36um(28)

************************ End of Initialization ************************

**2-D iteration stops at loop 111:

last max deviation =1.049E-03, at (29,15), by pass 63.23% ave per loop

At surface

Potential minimum = .589, at x = .484, (18)

Barrier height = .293

Current density = 4.167E-08 Amp/cm2

Barrier width = .210um, from .422 to .632...(17,20)

Barrier depth = 1.000um, from 0.000 to 1.000...(1,24)

Current / width = 94.2 Amp/um, with Uso=700.0 cm2/sec-V

Source depletion width = 0.000um

Drain depletion width = .787um

Saddle potential = .589, barrier height= -.293 at (.484, .023)...(18, 2)

Injection current /width = 87.4 Amp/um, with Uso=700.0 cm2/sec-V

Check results (Yes=f7/No=f8) ? "Yes"

Which one ? (2-dimensional plots/f1), (3-dimensional plots/f2),

(Save in Fmgr file /f3), (Print the numbers /f4). "Save on disc"

Referring to electron? (Yes=f7/No=f8) "No"

X-mesh->(n,x): "1 50"

Y-mesh->(n,y): "1 50"

-	1	2	3	4	5	6	7
YX(um)	-.18	-.12	-5.50E-02	0.0	1.60E-02	3.37E-02	5.34E-02
1 0.00	.88	.88	.87	.87	.86	.86	.85
2 .02	.88	.88	.87	.86	.86	.86	.85
3 .05	.88	.88	.87	.86	.86	.86	.85
4 .07	.88	.88	.87	.86	.86	.85	.85
5 .10	.88	.88	.87	.86	.86	.85	.85
6 .13	.88	.87	.87	.86	.86	.85	.85
7 .16	.87	.87	.87	.86	.85	.85	.84
8 .19	.87	.87	.86	.85	.85	.84	.84

Estimated MIN/MAX function values? (MIN=>MAX->skip) "0 1"

File name? "@poten::xx"

Another output? (Yes=f7/No=f8) "Yes"

Which one ? (Doping concentration/f1), (Carrier distribution/f2),
(Field distribution /f3), (Potential profile /f4). "Carrier"

Which one ? (2-dimensional plots/f1), (3-dimensional plots/f2),
(Save in FMGR file /f3), (Print the numbers /f4). "Save on disc"

Referring to electron? (Yes=f7/No=f8) "No"

X-mesh->(n,x): "1 50"

Y-mesh->(n,y): "1 50"

-	1	2	3	4	5	6	7
YX(um)	-.18	-.12	-5.50E-02	0.0	1.60E-02	3.37E-02	5.34E-02
1 0.00	-1.7E+20	-1.5E+20	-1.2E+20	-8.9E+19	-7.5E+19	-6.5E+19	-6.2E+19
2 .02	-1.7E+20	-1.5E+20	-1.2E+20	-8.9E+19	-7.4E+19	-6.4E+19	-5.1E+19
3 .05	-1.6E+20	-1.5E+20	-1.2E+20	-8.7E+19	-7.3E+19	-6.3E+19	-5.0E+19
4 .07	-1.6E+20	-1.4E+20	-1.1E+20	-8.3E+19	-7.0E+19	-6.0E+19	-4.8E+19
5 .10	-1.5E+20	-1.3E+20	-1.0E+20	-7.8E+19	-6.6E+19	-5.7E+19	-4.5E+19
6 .13	-1.3E+20	-1.2E+20	-1.0E+20	-7.2E+19	-6.1E+19	-5.2E+19	-4.2E+19
7 .16	-1.2E+20	-1.1E+20	-9.0E+19	-6.5E+19	-5.4E+19	-4.7E+19	-3.8E+19
8 .19	-1.0E+20	-9.7E+19	-7.8E+19	-5.6E+19	-4.7E+19	-4.1E+19	-3.3E+19

Estimated MIN/MAX function values? (MIN=>MAX->skip) "0 1"

File name? "@carri::xx"

Another output? (Yes=f7/No=f8) "No"

More iterations (Yes=f7/No=f8) ? "No"

****************** End of Two Dimensional Solution ******************

Another bias point (Yes=f7/No=f8) ? "No"

Another run (Yes=f7/No=f8) ? "No"

APPENDIX D

Test-Circuit Inputs to SPICE2.G

Invchn - Five-Stage Saturated Inverter Chain by Short MOS (MOS3)

```
.tran 0.12n 12n
.options defad=1e-9 defas=1e-9
.opt acct
.op
m1 7 7 2 8 nmos w=5u  l=2.3u
m2 2 1 0 8 nmos w=50u l=2.3u
m3 7 7 3 8 nmos w=5u  l=2.3u
m4 3 2 0 8 nmos w=50u l=2.3u
m5 7 7 4 8 nmos w=5u  l=2.3u
m6 4 3 0 8 nmos w=50u l=2.3u
m7 7 7 5 8 nmos w=5u  l=2.3u
m8 5 4 0 8 nmos w=50u l=2.3u
m9 7 7 6 8 nmos w=5u  l=2.3u
ma 6 5 0 8 nmos w=50u l=2.3u
vin 1 0 pulse(5 0 0.2n 1n 1n 5n 12n)
vdd 7 0 dc 5
vbb 8 0 dc 0
.model nmos nmos(vto=0.452  nsub=3.6e15 tox=0.065u ld=0.35u xj=0.5u
+     uo=580   theta=0.06  vmax=20e4  level=3  kappa=1.0  eta=0.035)
.plot  tran v(2) v(3) v(4) v(5) v(6) v(1) (-1,5)
.print tran v(2) v(3) v(4) v(5) v(6) v(1)
.end
```

Ratlog - Ratioless Dynamic Logic Circuit by Short MOS(MOS3)

```
.opt acct defl=2.3u defw=50u defas=1n defad=1n
.tran 1n 110n
m1  9  11  2  10  nmos
m2  9  12  4  10  nmos
m3  2   1  0  10  nmos
m4  4   3  0  10  nmos
m5  3  12  2  10  nmos
m6  1  11  5  10  nmos
c1  1   0  0.05pf
c2  2   0  0.05pf
c3  3   0  0.05pf
c4  4   0  0.05pf
c5  5   0  0.05pf
vin    5  0   pulse(0  4   1n   2n   2n   20n   500n)
vp1   11  0   pulse(0  5   1n   2n   2n   12n   52n)
vp2   12  0   pulse(0  5  26n   2n   2n   12n   52n)
vdd    9  0   dc  5
vbb   10  0   dc  -2.5
.model nmos nmos(vto=0.452  nsub=3.6e15 tox=0.065u ld=0.35u xj=0.5u
+      uo=580    theta=0.06  vmax=20e4  level=3  kappa=1.0  eta=0.035)
.plot  tran    v(4)  v(1)  v(2)  v(3)  v(4)  v(11)  v(12)  (0,6)
.print tran    v(4)  v(1)  v(2)  v(3)  v(4)  v(11)  v(12)  (0,6)
.end
```

Mosmem - MOS Memory Cell by Short MOS (MOS3)

```
.opt acct  defas=2n  defad=2n  defl=2.3u
.tran 0.5ns 60ns
.op
vdd 9 0 dc 5
vs  7 0 pulse(2 0 30ns  2ns  2ns 30ns   200ns)
vw  1 0 pulse(0 2  1ns  2ns 40ns 10ns   200ns)
vwb 2 0 pulse(2 0  1ns  2ns  2ns 100ns  200ns)
m1 3 1 0 0  nmos w=50u
m2 4 2 0 0  nmos w=50u
m3 9 9 3 0  nmos w=5u
m4 9 9 4 0  nmos w=5u
m5 5 7 3 0  nmos w=5u
m6 6 7 4 0  nmos w=5u
m7 5 6 0 0  nmos w=50u
m8 6 5 0 0  nmos w=50u
m9 9 9 5 0  nmos w=5u
m10 9 9 6 0 nmos w=5u
m11 8 4 0 0 nmos w=50u
m12 9 9 8 0 nmos w=5u
.model nmos nmos(vto=0.452  nsub=3.6e15 tox=0.065u ld=0.35u xj=0.5u
+     uo=580   theta=0.06  vmax=20e4  level=3  kappa=1.0  eta=0.035)
.print tran v(6) v(5) v(7) v(1) v(2) v(8)
.plot  tran v(6) v(5) v(7) v(1) v(2) v(8) (0,5)
.end
```

Bootinv - Bootstrapped Double Inverter Circuit by Short MOS (MOS3)

```
.opt acct
.tran 0.2ns 20ns
.op
m1 1 1 3 6 nmos w=10u l=2.3u ad=0.02p as=0.02p
m2 3 2 0 6 nmos w=50u l=2.3u ad=2p    as=0.02p
m3 1 1 4 6 nmos w=10u l=2.3u ad=0.2p  as=0.2p
m4 1 4 5 6 nmos w=10u l=2.3u ad=0.02p as=0.02p
m5 5 3 0 6 nmos w=50u l=2.3u ad=2p    as=0.02p
cl5 5 0 0.1pf
cl2 3 0 0.1pf
cb4 4 5 0.1pf
vdd 1 0 dc 5
vbb 6 0 dc 0
vin 2 0 pulse(4 0 1ns 2ns 2ns 13ns 20ns)
.model nmos nmos(vto=0.452  nsub=3.6e15 tox=0.065u ld=0.35u xj=0.5u
+      uo=580   theta=0.06  vmax=20e4  level=3  kappa=1.0  eta=0.035)
.print tran v(5) v(3) v(4) v(2)
.plot  tran v(5) v(3) v(2) (0,5)
.plot  tran v(4)
.end
```

Invchn - Five-Stage Saturated Inverter Chain by Short MOS (MOS2)

```
.tran 0.12n 12n
.options defad=1e-9 defas=1e-9
.opt acct
.op
m1 7 7 2 8 nmos w=5u  l=2.3u
m2 2 1 0 8 nmos w=50u l=2.3u
m3 7 7 3 8 nmos w=5u  l=2.3u
m4 3 2 0 8 nmos w=50u l=2.3u
m5 7 7 4 8 nmos w=5u  l=2.3u
m6 4 3 0 8 nmos w=50u l=2.3u
m7 7 7 5 8 nmos w=5u  l=2.3u
m8 5 4 0 8 nmos w=50u l=2.3u
m9 7 7 6 8 nmos w=5u  l=2.3u
ma 6 5 0 8 nmos w=50u l=2.3u
vin 1 0 pulse(5 0 0.2n 1n 1n 5n 12n)
vdd 7 0 dc 5
vbb 8 0 dc 0
.model nmos nmos(vto=0.452  nsub=3.6e15 tox=0.065u ld=0.35u xj=0.5u
+     uo=450    ucrit=12e4 uexp=0.240  utra=0.25  level=2
+     vmax=6e4  neff=7)
.plot  tran v(2) v(3) v(4) v(5) v(6) v(1) (-1,5)
.print tran v(2) v(3) v(4) v(5) v(6) v(1)
.end
```

Ratlog - Ratioless Dynamic Logic Circuit by Short MOS(MOS2)

```
.opt acct defl=2.3u defw=50u defas=1n defad=1n
.tran 1n 110n
m1  9  11  2  10  nmos
m2  9  12  4  10  nmos
m3  2   1  0  10  nmos
m4  4   3  0  10  nmos
m5  3  12  2  10  nmos
m6  1  11  5  10  nmos
c1  1   0  0.05pf
c2  2   0  0.05pf
c3  3   0  0.05pf
c4  4   0  0.05pf
c5  5   0  0.05pf
vin    5  0   pulse(0  4   1n   2n   2n   20n   500n)
vp1   11  0   pulse(0  5   1n   2n   2n   12n   52n)
vp2   12  0   pulse(0  5  26n   2n   2n   12n   52n)
vdd    9  0   dc  5
vbb   10  0   dc  -2.5
.model nmos nmos(vto=0.452  nsub=3.6e15 tox=0.065u ld=0.35u xj=0.5u
+     uo=450    ucrit=12e4 uexp=0.240  utra=0.25  level=2
+     vmax=6e4  neff=7)
.plot  tran    v(4)  v(1)  v(2)  v(3)  v(4)  v(11)  v(12)  (0,6)
.print tran    v(4)  v(1)  v(2)  v(3)  v(4)  v(11)  v(12)  (0,6)
.end
```

Mosmem - MOS Memory Cell (MOS2)

```
.opt acct  defas=2n  defad=2n  defl=2.3u
.tran 0.5ns 60ns
.op
vdd 9 0 dc 5
vs  7 0 pulse(2 0 30ns  2ns  2ns 30ns   200ns)
vw  1 0 pulse(0 2  1ns  2ns 40ns 10ns   200ns)
vwb 2 0 pulse(2 0  1ns  2ns  2ns 100ns  200ns)
m1 3 1 0 0  nmos w=50u
m2 4 2 0 0  nmos w=50u
m3 9 9 3 0  nmos w=5u
m4 9 9 4 0  nmos w=5u
m5 5 7 3 0  nmos w=5u
m6 6 7 4 0  nmos w=5u
m7 5 6 0 0  nmos w=50u
m8 6 5 0 0  nmos w=50u
m9 9 9 5 0  nmos w=5u
m10 9 9 6 0 nmos w=5u
m11 8 4 0 0 nmos w=50u
m12 9 9 8 0 nmos w=5u
.model nmos nmos(vto=0.452  nsub=3.6e15 tox=0.065u ld=0.35u xj=0.5u
+      uo=450    ucrit=12e4 uexp=0.240  utra=0.25  level=2
+      vmax=6e4  neff=7)
.print tran v(6) v(5) v(7) v(1) v(2) v(8)
.plot  tran v(6) v(5) v(7) v(1) v(2) v(8) (0,5)
.end
```

Bootinv - Bootstrapped Double Inverter Circuit by Short MOS (MOS2)

```
.opt acct
.tran 0.2ns 20ns
.op
m1 1 1 3 6 nmos w=10u l=2.3u ad=0.02p as=0.02p
m2 3 2 0 6 nmos w=50u l=2.3u ad=2p    as=0.02p
m3 1 1 4 6 nmos w=10u l=2.3u ad=0.2p  as=0.2p
m4 1 4 5 6 nmos w=10u l=2.3u ad=0.02p as=0.02p
m5 5 3 0 6 nmos w=50u l=2.3u ad=2p    as=0.02p
cl5 5 0 0.1pf
cl2 3 0 0.1pf
cb4 4 5 0.1pf
vdd 1 0 dc 5
vbb 6 0 dc 0
vin 2 0 pulse(4 0 1ns 2ns 2ns 13ns 20ns)
.model nmos nmos(vto=0.452  nsub=3.6e15 tox=0.065u ld=0.35u xj=0.5u
+      uo=450    ucrit=12e4 uexp=0.240  utra=0.25  level=2
+      vmax=6e4  neff=7)
.print tran v(5) v(3) v(4) v(2)
.plot  tran v(5) v(3) v(2) (0,5)
.plot  tran v(4)
.end
```

APPENDIX E

Listing of Program TWIST

File: TWDEFN

```
*DEFINE HP1000
*DEFINE INTERACTIVE
*DEFINE HP2648A
*DEFINE GRAPHICS
RAT4
S/<EXP>/E/G
```


File: TCOMMN

```
COMMON /CNTRL/ KONSOL,KLOOP,KSUPRM,KREAD,INPFIL(10),KEYBRD,^
               NEWNSH,NEWDOP,KDBUG,LURD1,LURD2,LUWR
COMMON /GEOMT/ XCHANL,XSOURC,XDRAIN,XGATE0,XGATE1,TOX0,TOX1,^
               XOX0,XOX1,XOXR,XIMPL0,XIMPL1,^
               XMAX,YMAX
COMMON /IMPLT/ TYPE,CSUB,DOPE(3),RANGE(3),STNDV(3),DOSE(3),^
               TEMP(3),DCOEF(3),DRVIN(3),XJCT(3),CPEAK(3),CSTEP(3)
COMMON /INDEX/ NSOURC,NDRAIN,NGATE0,NGATE1,NOX0,NOX1,NIMPL0,NIMPL1,^
               NXMAX,NYMAX,NOXIDE,^
               NSRC0,NDRN0
COMMON /PARAM/ WSURF(3),CSURF(3),^
               VFB,PHIB,GAMMB,ALPHB,PHIJ,CMINR0,^
               STYPE(3),PHIF(3),VBCRT(3),GAMM(3),ALPHA(3),PHIMP(3)
COMMON /POTEN/ VDB,VGB,VSB,PHIFP,PHIFN,ATOL1,ATOL2,RELAX1,RELAX2,^
               KIMSG0,KIMSG1,KIMSG2,LOCPHF(2,48),COFPHF(2,48),^
               KMAX2,KPHIS,KBIAS,PHSREF,ATOLS,VB
COMMON /CONST/ Q,EPSI,EPSIO2,VT300K,CNI
```

File: TEMA

```
COMMON /XYZ/ Z(50,50),CONC(50,48),CARRIE(50,48),POTSI(50,48),^
             XPOS(50),YPOS(48),DELX(49),DELY(47),QONE(50,48),^
             CNORTH(50,48),CSOUTH(50,48),CEAST(50,48),CWEST(50,48),^
             CNOX(50,2),CSOX(50,2),CEOX(50,2),CWOX(50,2),^
             QONE1(47),CN1(47),CS1(47),QONES1(50),CNS1(50),CSS1(50),^
             POTOX(50,2),DELOX(50,2),CNOX1(50,2),CSOX1(50,2)
```


File: TCOMMO

```
*CALL TCOMMN
      COMMON /TOUTV/ X1,X2,Y1,Y2,Z1,Z2,TILT,ROTAT,NX1,NX2,NY1,NY2,^
                     NX,NY,LU,KLOG
      COMMON /GLABL/ LABLX(40),LABLY(40),LABLZ(40),KFONTX(2,10),KFONTY(2,10),^
                     KFONTZ(2,10)
```

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
*IF HP1000&!BATCH
      PROGRAM TWIST
*ENDIF
*IF HP1000&BATCH
      PROGRAM TWISB
*ENDIF
*IF !HP1000
      PROGRAM TWIST (INPUT=201,OUTPUT=201,SUPSAV=201,KONSOL=201,KEYBRD=201,^
                     TAPE5=INPUT,TAPE6=OUTPUT,TAPE7=SUPSAV,TAPE8=KONSOL,^
                     TAPE9=KEYBRD)
*ENDIF
#
#     Two-dimensional Interactive Simulation of MOS Transistors
#
#                                        Sally Liu
#                                        Bernd Hoefflinger
#                                        Donald O. Pederson     March 1980
#
*CALL TEMA
*CALL TCOMMO
#
*IF !HP1000
      DATA      Q/1.602(EXP)-19/,  EPSI/1.036(EXP)-12/,EPSIO2/3.45(EXP)-13/,^
         VT300K/0.0259(EXP)0/ ,    CNI/1.450(EXP)+10/
*ENDIF
      DATA KYES/2HY /
#
#.....ASSIGN LU UNIT NUMBERS
*IF HP1000&!BATCH
      KONSOL=ITALU(JUNK)
      KEYBRD=KONSOL
*ENDIF
*IF HP1000&BATCH
      CALL SYBIO
      KONSOL=6
      KEYBRD=5
*ENDIF
*IF !HP1000
      KONSOL=9
      KEYBRD=8
      LURD1=5
      LURD2=7
      LUWR=6
*ENDIF
#
#.....PREPARE THE TERMINAL: HOME CURSOR
#                           CLEAR SCREEN
#                           DISPLAY LOGO
#                           DEFINE SOFT KEYS: F7=YES AND F8=NO
#                           LOCK DISPLAY MEMORY
*IF HP1000&!BATCH
      CALL LINK (1HD,KANSWR,-1)
*ENDIF
*IF !HP1000
      CALL OUTP1 (KANSWR,-1)
*ENDIF
#
#.....INITIALIZE GEOMTRY AND DOPING PROFILE
      KGO=1
      KLOOP=0
      WHILE (KGO.EQ.1) [
          KLOOP=KLOOP+1
*IF HP1000&!BATCH
          CALL LINK (1HA)
          CALL LINK (1HB)
*ENDIF
*IF HP1000&BATCH
          CALL LINK (1H1)
          CALL LINK (1H2)
*ENDIF
*IF !HP1000
          CALL GETPA
          CALL BETPA
*ENDIF
#
#.....CHECK PROFILE
```



```
*IF HP1000&!BATCH
          CALL LINK (1HD,KANSWR,-2)
*ENDIF
*IF HP1000&BATCH
          CALL LINK (1H4,KANSWR,-2)
*ENDIF
*IF !HP1000
          CALL OUTP1 (KANSWR,-2)
*ENDIF
          IF (KANSWR.EQ.KYES) CALL OUTPS (1)
#
#.....CHECK MESH
*IF HP1000&!BATCH
          CALL LINK (1HD,KANSWR,-3)
*ENDIF
*IF HP1000&BATCH
          CALL LINK (1H4,KANSWR,-3)
*ENDIF
*IF !HP1000
          CALL OUTP1 (KANSWR,-3)
*ENDIF
          IF (KANSWR.EQ.KYES) CALL OUTPS (-1)
#
#.....SAVE INPUT PARAMETERS
*IF HP1000&!BATCH
          CALL LINK (1HD,KANSWR,-4)
*ENDIF
*IF HP1000&BATCH
          CALL LINK (1H4,KANSWR,-4)
*ENDIF
*IF !HP1000
          CALL OUTP1 (KANSWR,-4)
*ENDIF
          IF (KANSWR.EQ.KYES) CALL OUTPS (0)
#
#.....SOLVE FOR POTENTIAL
          KSOLV=0
          REPEAT [
*IF HP1000&!BATCH
              CALL LINK (1HC,KSOLV)
*ENDIF
*IF HP1000&BATCH
              CALL LINK (1H3,KSOLV)
*ENDIF
*IF !HP1000
              CALL SOLVE (KSOLV)
*ENDIF
              IF (KSOLV.NE.0) CALL OUTPS (4)
          ] UNTIL (KSOLV.EQ.0)
#
#.....CHECK IF ANOTHER RUN
*IF HP1000&!BATCH
          CALL LINK (1HD,KANSWR,-5)
*ENDIF
*IF HP1000&BATCH
          CALL LINK (1H4,KANSWR,-5)
*ENDIF
*IF !HP1000
          CALL OUTP1 (KANSWR,-5)
*ENDIF
          IF (KANSWR.EQ.KYES)  KGO=1
          ELSE                 KGO=0
      ]
#
#.....UN-LOCK THE MEMORY
*IF HP1000&!BATCH
      CALL LINK (1HD,KANSWR,-6)
*ENDIF
*IF HP1000&BATCH
      CALL LINK (1H4,KANSWR,-6)
*ENDIF
*IF !HP1000
      CALL OUTP1 (KANSWR,-6)
*ENDIF
*IF HP1000&!BATCH
      CALL EXEC (6)
*ELSE
      STOP
*ENDIF
      END
```

File: &TWIST

```
*IF HP1000
      BLOCK DATA TBLKD, Common Blocks
*
*     INITIALIZE COMMON BLOCKS
*
*CALL TCOMMO
*
      DATA      Q/1.602(EXP)-19/,  EPSI/1.036(EXP)-12/,EPSI02/3.45(EXP)-13/,^
         VT300K/0.0259(EXP)0/ ,    CNI/1.450(EXP)+10/
      END
*ELSE
      INTEGER FUNCTION IFBRK (DUMMY)
      IFBRK=0
      RETURN
      END
*ENDIF
```


File: &ANGLE

```
*CALL TWDEFN
      SUBROUTINE ANGLE (THETA,KQUAD) -
*
*   1) CHANGE ANGLE THETA FROM DEGREE TO RADIENT
*   2) LIMIT IT BETWEEN 0 AND 2-PI
*   3) SET QUADRANT-INDEX KQUAD
*
*.....DEFINE CONSTANTS
      PI=4 0(EXP)0*ATAN2(1.0(EXP)0,1.0(EXP)0)
      TWOPI=PI+PI
      RAD=PI/180.0(EXP)0
      QUAD12=0.5(EXP)0*PI
      QUAD23=PI
      QUAD34=PI+QUAD12
      QUAD41=0.0(EXP)0
*
*.....TRANFORM THETA FROM DEGREE INTO RADIENT, TAKE THE ABSOLUTE VALUE
      THETA=THETA*RAD
      ABSX=ABS(THETA)
*
*.....LIMIT THETA BETWEEN 0 AND 2-PI
      WHILE (ABSX.GT.TWOPI)  ABSX=ABSX-TWOPI
      IF    (THETA.GE.0.0(EXP)0) THETA=ABSX
      ELSE                       THETA=TWOPI-ABSX
*
*.....GET THE QUADRANT INDEX
      IF             (THETA.LT.QUAD12) KQUAD=1
      ELSE IF        (THETA.LT.QUAD23) KQUAD=2
           ELSE IF (THETA.LT.QUAD34) KQUAD=3
                ELSE                 KQUAD=4
*
*.....DONE
      RETURN
      END
```

File: &ASCAL

```
*CALL TWDEFN
      SUBROUTINE ASCAL (IGCB,XONY,XMIN,XMAX,YMIN,YMAX)
*
*     ALLOCATE AND SCALE THE CENTRAL PORTION OF THE PLOTTING SURFACE
*
      DIMENSION IGCB(192)
*
*.....DEFINE MARGINS, CASE OF VERTICAL ORIENTATION
      IF (XONY.LE.1.0(EXP)0) [
          XFRAME=100.0(EXP)0
          YFRAME=XFRAME/XONY
          XLEFT=XFRAME/3.6(EXP)0
          XRITE=XFRAME/6.0(EXP)0
          YLOWR=YFRAME*0.2(EXP)0
          YUPPR=YLOWR
      ]
*
*.........................CASE OF HORIZONTAL ORIENTATION
      ELSE [
          YFRAME=100.0(EXP)0
          XFRAME=YFRAME*XONY
          XLEFT=XFRAME*0.2(EXP)0
          XRITE=XLEFT
          YLOWR=YFRAME/3.6(EXP)0
          YUPPR=YFRAME/6.0(EXP)0
      ]
*
*.....DEFINE THE BOUNDARY
      XLIMT1=XLEFT
      XLIMT2=XFRAME-XLEFT
      YLIMT1=YLOWR
      YLIMT2=YFRAME-YUPPR
      XRATIO=(XMAX-XMIN)/(XLIMT2-XLIMT1)
      YRATIO=(YMAX-YMIN)/(YLIMT2-YLIMT1)
*
*.....DEFINE THE PLOTTING AREA
      CALL VIEWP(IGCB,0.0(EXP)0,XFRAME,0.0(EXP)0,YFRAME)
      CALL WINDW(IGCB,XMIN-XRATIO*XLEFT,XMAX+XRATIO*XRITE,^
                YMIN-YRATIO*YLOWR,YMAX+YRATIO*YUPPR)
*
*....DONE
     RETURN
     END
```


File: &ASTEP

```
*CALL TWDEFN
      SUBROUTINE ASTEP (KIMPL)
*
*     PROCESS PARAMTERS OF STEP PROFILE APPROXIMATION
*
*CALL TCOMMN
*
      STNDV2=STNDV(KIMPL)*STNDV(KIMPL)
      FOURDT=(STNDV2+STNDV2)+4.0(EXP)0*DCOEF(KIMPL)*DRVIN(KIMPL)
      IF (FOURDT.NE.0.0(EXP)0) [
          PI=4.0(EXP)0*ATAN2(1.0(EXP)0,1.0(EXP)0)
          CPEAK(KIMPL)=DOSE(KIMPL)/SQRT(PI*FOURDT)
          XJCT(KIMPL)=RANGE(KIMPL)+SQRT(FOURDT*ALOG(ABS(CPEAK(KIMPL)/CSUB)))
          CSTEP(KIMPL)=DOSE(KIMPL)/XJCT(KIMPL)
      ]
      ELSE [
          CPEAK(KIMPL)=DOSE(KIMPL)
          XJCT(KIMPL)=RANGE(KIMPL)
          CSTEP(KIMPL)=DOSE(KIMPL)
      ]
*
*.....DISPLAY IMPLANT PARAMETERS
      WRITE (KONSOL,1000) DCOEF(KIMPL),CPEAK(KIMPL),XJCT(KIMPL),CSTEP(KIMPL)
1000  FORMAT(/,"..... profile parameters: ",^
             /,10X,"diff const =",1PE10.3,"(cm2/sec)",^
                5X,"peak conc  =",1PE10.3,"(cm-3)",^
             /,10X,"jct depth  =",1PE10.3,"(cm)",^
                8X,"average conc =",1PE10.3,"(cm-3)"/)
*
*.....DONE
      RETURN
      END
```

File: &AXLAB

```
*CALL TWDEFN
      SUBROUTINE AXLAB (KGCB,KDCB,XONY,X1,X2,Y1,Y2,LABLX,LABLY,KFONTX,KFONTY)
#
#      LABEL AXES ON 2-D PLOTS
#
      DIMENSION KGCB(1),KDCB(1),LABLX(1),LABLY(1),KFONTX(1),KFONTY(1)
#
      DATA NDGTO/5/, CHARN/20.0(EXP)0/,^
           ASPEC/0.7(EXP)0/, SLANT/0.0(EXP)0/, SUP/0.75(EXP)0/
#
      CALL TSCAL (KGCB,XONY,CHARN,CHITE,XLENG,YLENG)  # SCALE CHARACTER SIZE
      CHITY=CHITE*(Y2-Y1)/YLENG                       # X'FORM INTO USER
      CHITX=CHITY*(X2-X1)/(Y2-Y1)/XONY                #   DEFINED
      CHXFTR=CHITX*SUP                                #   UNITS
      CHYFTR=CHITY*SUP
      CALL LORG (KGCB,1)                              # SET TEXT ORIGIN
      CALL LINE (KGCB,0)                              # SET SOLID LINE
      PION2=2.0(EXP)0*ATAN2(1.0(EXP)0,1.0(EXP)0)
      CALL LABLN (KGCB,KDCB,LABLY,KFONTY,CHITE,TEXTY,NFY)
      IF (NFY.NE.0) [                                 # Y-LABEL
          CALL LDIR(KGCB,PION2)                       # VERTICAL DIR.
          TEXTY=TEXTY*(Y2-Y1)/(X2-X1)                 # SCALE TEXT LENGTH
          X0=X1-(NDGTO+0.5(EXP)0)*CHITX
          Y0=(Y1+Y2-TEXTY)*0.5(EXP)0
          CALL MOVE (KGCB,X0,Y0)                      # MOVE TO FIRST POINT
          XOFF=-CHXFTR                                # DEFINE SUPER-SCRIPT
          YOFF=0.0(EXP)0
          CALL LABWR (KGCB,KDCB,LABLY,KFONTY,CHITE,NFY,XOFF,YOFF)
      ]
      CALL LABLN (KGCB,KDCB,LABLX,KFONTX,CHITE,TEXTX,NFX)
      IF (NFX.NE.0) [                                 # X-LABEL
          CALL LDIR (KGCB,0.0(EXP)0)
          Y0=Y1-NDGTO*CHITY*ASPEC-CHITY-CHITY         # POSITION X-LABEL
          X0=(X1+X2-TEXTX)*0.5(EXP)0
          CALL MOVE (KGCB,X0,Y0)                      # MOVE TO STARTING P'NT
          XOFF=0.0(EXP)0
          YOFF=CHYFTR
          CALL LABWR (KGCB,KDCB,LABLX,KFONTX,CHITE,NFX,XOFF,YOFF)
      ]
#
#.....DONE
      RETURN
      END
```


File: &AXLIN

```
*CALL TWDEFN
      SUBROUTINE AXLIN (KGCB,KDCB,XONY,X1,X2,Y1,Y2)
#
#      DRAW A FRAME DEFINED BY (X1,X2,Y1,Y2) WITH LINEAR TIC MARK
#           ALWAYS DIVIDED INTO 5 GRIDS EACH WITH 5 TIC MARKS (TOTALLY 25)
#
      DIMENSION KGCB(1),KDCB(1),LBUF(4),XTIC(4),YTIC(4),XDEL(4),YDEL(4)
#
      EQUIVALENCE (NTIC,NXTIC,NYTIC),(NTICO,NXTICO,NYTICO)
#
      DATA CHARN/25.0(EXP)0/, NSEC/0/,  NCR/2HGG/
      DATA TRATIO/0.02(EXP)0/,NTICO/25/,NTIC/5/ ,KGRID/0/
#
#.....SCALE THE CHARACTERS
      CALL TSCAL (KGCB,XONY,CHARN,CHITE,XLENG,YLENG)
      XSPAN=X2-X1;                          YSPAN=Y2-Y1
      CHITY=CHITE*YSPAN/YLENG;              CHITX=CHITE*XSPAN/(XONY*XLENG)
#
#.....TIC SIZE
      TICX=XSPAN*TRATIO;                    TICY=YSPAN*TRATIO
      XTIC(1)=TICX;                         YTIC(1)=0.0(EXP)0
      XTIC(2)=0.0(EXP)0;                        YTIC(2)=TICY
      XTIC(3)=-TICX;                        YTIC(3)=0.0(EXP)0
      XTIC(4)=0.0(EXP)0;                        YTIC(4)=-TICY
#
#.....THE SPACING
      XTSPAC=XSPAN/NXTICO;                  YTSPAC=YSPAN/NYTICO
      XDEL(1)=0.0(EXP)0;                        YDEL(1)=-YTSPAC
      XDEL(2)=XTSPAC;                       YDEL(2)=0.0(EXP)0
      XDEL(3)=0.0(EXP)0;                        YDEL(3)=YTSPAC
      XDEL(4)=-XTSPAC;                      YDEL(4)=0.0(EXP)0
#
#.....LABEL ATTRIBUTION
      CALL LORG (KGCB,7)
      CALL LINE (KGCB,0)
      CALL GFONT(KGCB,6NFONT2 ,NSEC,NCR,KDCB)
      NS=1
      NC=8
1010  FORMAT (F7.2,1X)
#
#.....LEFT, LOWER, RIGHT, UPPER
      XP=X1
      YP=Y2
      PION2=2.0(EXP)0*ATAN2(1.0(EXP)0,1.0(EXP)0)
      DO KD=1,4 [
          XD=XDEL(KD);                      YD=YDEL(KD) # DEFINE SPACE
          XT=XTIC(KD);                      YT=YTIC(KD) # DEFINE TIC SIZE
          XTO=0.5(EXP)0*XT;                     YTO=0.5(EXP)0*YT
          IF ((KD.EQ.1).OR.(KD.EQ.3)) [
              IF (KD.EQ.1) [ CALL LDIR (KGCB,0.0(EXP)0); PRT=YP ]
          ]
          ELSE [
              IF (KD.EQ.2) [ CALL LDIR (KGCB,PION2);      PRT=XP ]
          ]
          CALL MOVE (KGCB,XP,YP)
          IF ((KD.EQ.1).OR.(KD.EQ.2)) [                 # MARK 1ST LABEL
              CALL CODE; WRITE (LBUF,1010) PRT
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
              CALL MOVE  (KGCB,XP,YP)
          ]
          DO K=1,NTICO [                                # DRAW THE AXIS
              XP=XP+XD;                     YP=YP+YD
              CALL DRAW (KGCB,XP,YP)
              KT=MOD(K,NTIC)
              IF (KT.EQ.0) [ TX=XT;     TY=YT    ]
              ELSE         [ TX=XTO;    TY=YTO   ]
              CALL DRAW (KGCB,XP+TX,YP+TY)
              CALL MOVE (KGCB,XP,YP)
              IF ((KT.EQ.0).AND.((KD.EQ.1).OR.(KD.EQ.2))) [
                  IF (KD.EQ.1) PRT=YP
                  ELSE         PRT=XP
                  CALL CODE;  WRITE (LBUF,1010) PRT
                  CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
                  CALL MOVE  (KGCB,XP,YP)
              ]
          ]
      ]
#
#.....CLOSE FONT FILE
      CALL GFONT (KGCB,0,0,0,KDCB)
```

File: &AXLIN

```
#
#.....DONE
      RETURN
      END
```


File: &AXLOG

```
*CALL TWDEFN
      SUBROUTINE AXLOG (KGCB,KDCB,XONY,X1,X2,Y1,Y2)
#
#     DRAW A FRAME DEFINED BY (X1,X2,Y1,Y2) WITH SEMI-LOG TIC MARK
#          EACH DECADE IN Y DIVIDED INTO 'NYTIC'((=20) TIC MARKS
#          ASSUME  BOTH  Y1,Y2 )0
#
      DIMENSION KGCB(1),KDCB(1),LBUF(4),XTIC(4),YTIC(4),XDEL(4),YDEL(4),^
                DYLG(20),NYOFF(4),YOFF(2),NAMTEN(3)
#
      DATA NSEC/0/, NCR/2HGG/, NAMTEN/2H10,2H  ,2H  /
      DATA CHARN/25.0(EXP)0/,  SUP/0.75(EXP)0/,^
           ASPEC/0.7(EXP)0/,   SLANT/0.0(EXP)0/
#
#.....PRESET DATA
      DATA YTSPAC/2.0(EXP)0/,  TRATIO/0.02(EXP)0/
      DATA NXTIC/5/, NXGRD/25/, KGRID/0/
#
#.....SCALE CHARACTERS
      CALL TSCAL (KGCB,XONY,CHARN,CHITE,XLENG,YLENG)
      XSPAN=X2-X1;                          YSPAN=Y2-Y1
      CHITY=CHITE*YSPAN/YLENG;              CHITX=CHITE*XSPAN/(XONY*XLENG)
      CHYFTR=CHITY*SUP
#
#.....TIC SIZE
      TICX=XSPAN*TRATIO;                    TICY=YSPAN*TRATIO
      XTIC(1)=TICX;                         YTIC(1)=0.0(EXP)0
      XTIC(2)=0.0(EXP)0;                        YTIC(2)=TICY
      XTIC(3)=-TICX;                        YTIC(3)=0.0(EXP)0
      XTIC(4)=0.0(EXP)0;                        YTIC(4)=-TICY
#
#.....LINEAR SPACING
      XTSPAC=XSPAN/NXGRD
      XDEL(1)=0.0(EXP)0;                        YDEL(1)=-1.0(EXP)0
      XDEL(2)=XTSPAC;                       YDEL(2)=0.0(EXP)0
      XDEL(3)=0.0(EXP)0;                        YDEL(3)=1.0(EXP)0
      XDEL(4)=-XTSPAC;                      YDEL(4)=0.0(EXP)0
#
#.....LOG SPACING
      Y11=FLOAT(IFIX(Y1))                                      # TAKE THE INTEGER
      Y11A=Y11
      Y22=FLOAT(IFIX(Y2))
      Y22A=Y22
      IF (Y11.NE.Y1) [ IF (Y1.LT.0.0(EXP)0) Y11=Y11-1 ]
      ELSE                                  Y11=Y11-1
      IF (Y22.NE.Y2) [ IF (Y2.LT.0.0(EXP)0) Y22=Y22-1 ]
      ELSE                                  Y22=Y22-1
      NYGRD=Y22-Y11                                            # NUMBER OF DECADES
      YTSPAC=AMIN1(ABS(YTSPAC),10.(EXP)0)                         # TIC SPACING
#
#.....COMMENT OUT UN-USED CODE
#     IF (YTSPAC.EQ.0.0(EXP)0) YTSPAC=2.0(EXP)0
#     IF (YTSPAC.EQ.10.0(EXP)0) [                                 # IF NO TICS
#         NYGO=1
#         NYOFF(1)=1;     NYOFF(2)=1
#         NYOFF(3)=1;     NYOFF(4)=1
#         DYLG(1)=ALOGT(YTSPAC)
#     ]
#     ELSE [                                                   # WITH TICS
#         NYGO=MINO(IFIX(10.0(EXP)0/YTSPAC),20)                   # TIC COUNT
#         YTS0=10.0(EXP)0/NYGO                                    # ROUND OFF

          NYGO=5
          YTS0=YTSPAC
          YTS=YTS0
          DO N=1,NYGO [                                        # TIC OFF SET IN LOG10
              DYLG(N)=ALOGT(YTS)
              YTS=YTS+YTS0
          ]
          YOFF(2)=Y2-Y22                                       # AXIS LIMIT OFF SET
          YOFF(1)=Y1-Y11
          DO N=1,2 [                                           # TIC NEXT TO THE END
              IF (YOFF(N).EQ.0.0(EXP)0) NYOFF(N)=0
              ELSE [
                  J=1
                  JEND=0
                  WHILE ((J.LE.NYGO).AND.(JEND.EQ.0)) [
                      IF (YOFF(N).LE.DYLG(J)) JEND=J
                      ELSE                    J=J+1
                  ]
```

112

```
         IF (JEND.EQ.0)                  JEND=1
         IF ((N.EQ.2).AND.(YOFF(N).NE.DYLG(JEND))) [
             JEND=JEND-1
             IF (JEND.EQ.0) [
                 JEND=NYGO
                 Y22=Y22-1
                 YOFF(2)=YOFF(2)+1
             ]
         ]
         IF ((N.EQ.1).AND.(JEND.EQ.NYGO).AND.(Y11.EQ.Y11A)) [
             JEND=1                                     # IF Y1 IS INTEGER
             Y11=Y11+1
         ]
         NYOFF(N)=JEND
     ]
   ]
   NYOFF(3)=NYOFF(1)                                    # 1ST TIC INDICES
   NYOFF(4)=NYOFF(2)
   NYOFF(1)=NYOFF(2)
   NYOFF(2)=NYOFF(3)
#     ]
#
#.....LABEL ATTRIBUTION
      CALL LORG (KGCB,7)
      CALL LINE (KGCB,0)
      CALL GFONT(KGCB,6HFONT2 ,NSEC,NCR,KDCB)
      NS=1;       NC=8                                  # LINEAR INDICES
      NSB=1;      NCB=5                                 # BASE 10 INDICES
      NSE=1;      NCE=4                                 # EXPONENT INDICES
1050  FORMAT (I3,1X)
1010  FORMAT (F7.2,1X)
#
#.....LEFT, LOWER, RIGHT, UPPER
      XP=X1;                          YP=Y2
      PION2=2.0(EXP)0*ATAN2(1.0(EXP)0,1.0(EXP)0)
      DO KD=1,4 [
          XD=XDEL(KD);                YD=YDEL(KD)       # GRID SPACE
          XT=XTIC(KD);                YT=YTIC(KD)       # TIC SIZE
          XTO=0.5(EXP)0*XT;               YTO=0.5(EXP)0*YT
          IF ((KD.EQ.1).OR.(KD.EQ.3)) [                 # VERTICAL AXES
              NPT=NYGRD
              IF (KD.EQ.1) YPO=Y22                      # 1ST DECADE
              ELSE         YPO=Y11
              KK1=NYOFF(KD)                             # 1ST TIC
              IF ((KD.EQ.3).AND.(Y2.EQ.Y22A)) [ KK1=1; YPO=YPO+1 ]
              IF (NYGRD.GE.1) IF (KD.EQ.1) KK2=1        # LAST TIC IN DECADE
                              ELSE         KK2=NYGO
              ELSE                         KK2=NYOFF(KD+1)
          ]
          ELSE NPT=NXGRD                                # HORIZONTAL AXES
          CALL MOVE (KGCB,XP,YP)
          IF ((KD.EQ.1).OR.(KD.EQ.2)) [                 # MARK 1ST LABEL
              IF (KD.EQ.1) CALL LDIR (KGCB,0.0(EXP)0)
              ELSE         CALL LDIR (KGCB,PION2)
              IF ((KD.EQ.1).AND.(Y2.EQ.Y22A)) [
                  CALL CSIZE (KGCB,CHITE,ASPEC,SLANT,0)
                  CALL GTEXT (KGCB,NAMTEN,NSB,NCB,KDCB)
                  CALL MOVE  (KGCB,XP,YP+CHYFTR)
                  CALL CSIZE (KGCB,CHITE*SUP,ASPEC,SLANT,0)
                  IPRT=YP;    CALL CODE; WRITE (LBUF,1050) IPRT
                  CALL GTEXT (KGCB,LBUF,NSE,NCE,KDCB)
                  CALL MOVE  (KGCB,XP,YP)
              ]
              IF (KD.EQ.2) [
                  CALL CSIZE (KGCB,CHITE,ASPEC,SLANT,0)
                  CALL CODE; WRITE (LBUF,1010) XP
                  CALL GTEXT(KGCB,LBUF,NS,NC,KDCB)
                  CALL MOVE  (KGCB,XP,YP)
              ]
              CALL MOVE (KGCB,XP,YP)
          ]
          DO K=1,NPT [
              IF ((KD.EQ.2).OR.(KD.EQ.4)) [             # LINEAR AXES
                  XP=XP+XD
                  CALL DRAW (KGCB,XP,YP)
                  KT=MOD(K,NXTIC)
                  IF (KT.EQ 0) [ IF (KGRID.EQ.0)      TY=YT
                                 ELSE IF (KD.EQ.2)    TY=YSPAN
                                      ELSE            TY=0.0(EXP)0
                               ]
```



```
                  ELSE                                TY=YTO
                  CALL DRAW (KGCB,XP,YP+TY)
                  CALL MOVE (KGCB,XP,YP)
                  IF ((KT.EQ.0).AND.(KD.EQ.2)) [
                      CALL CODE;  WRITE (LBUF,1010) XP
                      CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
                      CALL MOVE  (KGCB,XP,YP)
                  ]
              ]
              ELSE [                                    # LOG AXES
                  IF ((K.EQ.NPT).AND.(YOFF(KD/2+1).NE.1.0(EXP)0))
                      KK2=NYOFF(KD+1)
                  IS=ISIGN(1,KK2-KK1)                   # DIRECTION
                  KK=KK1                                # DRAW THE AXIS
                  IF (KK1.NE.KK2)
                      WHILE ((KK.EQ.KK2).OR.(ISIGN(1,KK2-KK).EQ.IS)) [
                          YP=YPO+DYLG(KK)
                          CALL DRAW (KGCB,XP,YP)
                          IF (KK.NE.NYGO)         TX=XTO  # TIC OFF SET
                          ELSE [ IF (KGRID.EQ.0)  TX=XT
                                 ELSE IF (KD.EQ.1) TX=XSPAN
                                      ELSE        TX=0.0(EXP)0
                               ]
                          CALL DRAW (KGCB,XP+TX,YP)       # DRAW THE TIC
                          CALL MOVE (KGCB,XP,YP)
                          IF ((KK.EQ.NYGO).AND.(KD.EQ.1)) [# LABEL THE AXIS
                              CALL CSIZE (KGCB,CHITE,ASPEC,SLANT,0)
                              CALL GTEXT (KGCB,NAMTEN,NSB,NCB,KDCB)
                              CALL MOVE  (KGCB,XP,YP+CHYFTR)
                              CALL CSIZE (KGCB,CHITE*SUP,ASPEC,SLANT,0)
                              IPRT=YP; CALL CODE; WRITE (LBUF,1050) IPRT
                              CALL GTEXT (KGCB,LBUF,NSE,NCE,KDCB)
                              CALL MOVE  (KGCB,XP,YP)
                          ]
                          KK=KK+IS                      # UPDATE TIC INDEX
                      ]
                  ELSE [                                # IF ONLY ONE TIC
                      YP=YPO+DYLG(KK1)
                      CALL DRAW (KGCB,XP,YP)
                      CALL DRAW (KGCB,XP+XTO,YP)
                      CALL MOVE (KGCB,XP,YP)
#
#.....COMMENT OUT UN-USED CODE
#                     IF (KD.EQ.1) [
#                         CALL CSIZE (KGCB,CHITE,ASPEC,SLANT,0)
#                         PRT=10.0(EXP)0**(DYLG(KK1))
#                         CALL CODE; WRITE (LBUF,1010) PRT
#                         CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
#                         CALL MOVE (KGCB,XP,YP)
#                     ]
#
                  ]
                  IF (K.NE.NPT) YPO=YPO+YD              # UPDATE DECADE
                  ELSE IF (YOFF(KD/2+1).NE.DYLG(KK2)) [ # LAST SEGMENT
                      IF (KD.EQ.1) YP=Y1
                      ELSE         YP=Y2
                      CALL DRAW (KGCB,XP,YP)
                      IF ((KD.EQ.1).AND.(Y1.EQ.Y11A)) [
                          CALL CSIZE (KGCB,CHITE,ASPEC,SLANT,0)
                          CALL GTEXT (KGCB,NAMTEN,NSB,NCB,KDCB)
                          CALL MOVE  (KGCB,XP,YP+CHYFTR)
                          CALL CSIZE (KGCB,CHITE*SUP,ASPEC,SLANT,0)
                          IPRT=YP;    CALL CODE; WRITE (LBUF,1050) IPRT
                          CALL GTEXT (KGCB,LBUF,NSE,NCE,KDCB)
                          CALL MOVE  (KGCB,XP,YP)
                      ]
                  ]
                  IF (K.EQ.1) IF (KD.EQ.1) KK1=NYGO     # UPDATE INDEX
                              ELSE         KK1=1
              ]
          ]
      ]
      IF (NYGRD.EQ.0) [
          IF (Y1.NE.Y11A) [
              CALL MOVE (KGCB,X1,Y1)
              PRT=10.0(EXP)0**Y1
              CALL CSIZE (KGCB,CHITE,ASPEC,SLANT,0)
              CALL CODE; WRITE (LBUF,1010) PRT
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
          ]
```

File: &AXLOG

```
        IF (Y2.NE.Y22A) [
            CALL MOVE (KGCB,X1,Y2)
            PRT=10.0(EXP)0**Y2
            CALL CSIZE (KGCB,CHITE,ASPEC,SLANT,0)
            CALL CODE; WRITE (LBUF,1010) PRT
            CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
        ]
      ]
#
#.....CLOSE FONT FILE
      CALL GFONT (KGCB,0,0,0,KDCB)
#
#.....DONE
      RETURN
      END
```


File: &CARRI

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE CARRI
#
#     CALCULATE CHARGE DENSITY PER UNIT AREA
#        Z(N,1) -> XPOS(N) IN UM
#        Z(N,2) -> CARRIERS PER UNIT AREA
#
*CALL TEMA
*CALL TCOMMN
#
#.....LOOP FROM SOURCE TO DRAIN
      TSUM=0.0(EXP)0
      DO N=NSOURC,NDRAIN [
#
#.... LOCATE UPPER BOUNDARY
            M=1
            TYPC=SIGN(1.0(EXP)0,(CARRIE(N,M)))
            WHILE ((M.LT.NYMAX).AND.(TYPC.NE.TYPE)) [
                  M=M+1
                  TYPC=SIGN(1.0(EXP)0,(CARRIE(N,M)))
            ]
#
#.....INTEGRATE CARRIERS
            CALL INTGR (N,M,MB1,MB2,QSUM)
            Z(N,1)=XPOS(N)*1.0(EXP)4
            Z(N,2)=ABS(QSUM)
            TSUM=TSUM+QSUM
      ]
#
#.....PRINT OUTPUT
      IF (TSUM.NE.0.0(EXP)0) WRITE (KONSOL,1000) (Z(K,2),K=NSOURC,NDRAIN)
1000  FORMAT("**Carrier per cm2 from junction to junction : "/,(3X,7(1PG10.3)))
#
#.....DONE
      RETURN
      END
```

File: &CHECK

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE CHECK
$
$     CHARACTERIZE DEVICE
$
*CALL TEMA
*CALL TCOMMN
$
$.....CHECK SURFACE REGION
      CALL CROSS (1)
$
$.....CHECK FOR PUNCH THROUGH
      CALL SADDL
$
$.....DONE
      RETURN
      END
```


File: &CHKDP

```
*CALL TWDEFN
      SUBROUTINE CHKDP (KIMPL)
$
$     SCALE AND LIMIT IMPLANT PARAMETERS
$
*CALL TCOMMN
$
      DIMENSION DIFF(4),EAOVRK(4)
$
$.....SCALING FACTORS
      DATA UM/1.0(EXP)-4/,  SECND/60.0(EXP)0/,  ZEROC/273.0(EXP)0/
$
$.....CONSTANTS
      DATA   DIFF/2.551(EXP)1, 5.230(EXP)0, 1.445(EXP)3, 1.803(EXP)3/,^
          EAOVRK/4.430(EXP)4, 4.525(EXP)4, 5.100(EXP)4, 5.410(EXP)4/
$
$.....SCALE LOCAL IMPLANT LOCATION
      IF (KIMPL.EQ.2) [
          XIMPL0=XIMPL0*UM
          XIMPL1=XIMPL1*UM
          IF (XIMPL0.GT.XIMPL1) [ T=XIMPL0; XIMPL0=XIMPL1; XIMPL1=T ]
      ]
$
$.....DETERMINE DOPANT TYPE
      KDOPE=DOPE(KIMPL)
      IF ((KDOPE.EQ.1).OR.(KDOPE.EQ.6)) DTYPE=-1.0E0
      ELSE                              DTYPE=1.0E0
$
$.....SCALING
      RANGE(KIMPL)=RANGE(KIMPL)*UM
      STNDV(KIMPL)=AMAX1(STNDV(KIMPL)*UM,0.0(EXP)0)
      DOSE(KIMPL)=SIGN(DOSE(KIMPL),DTYPE)
      DRVIN(KIMPL)=AMAX1(DRVIN(KIMPL)*SECND,0.0(EXP)0)
$
$.....DETERMINE THE DRIVE-IN DIFFUSIVITY
      IF ((KDOPE.GE.1).AND.(KDOPE.LE.4))
           DCOEF(KIMPL)=DIFF(KDOPE)*EXP(-EAOVRK(KDOPE)/(TEMP(KIMPL)+ZEROC))
      ELSE DCOEF(KIMPL)=AMAX1(DCOEF(KIMPL),0.0(EXP)0)
$
$.....STEP APPROXIMATION
      CALL ASTEP (KIMPL)
$
$.....DONE
      RETURN
      END
```

File: &CHKGM

```
*CALL TWDEFN
      SUBROUTINE CHKGM
#
#     SCALE AND LIMIT GEOMETRY PARAMETERS
#
*CALL TCOMMN
#
#.....SCALING FACTOR
      DATA UM/1(EXP)-4/
#
#.....SCALING AND LIMITING
      TOXO=AMAX1(TOXO*UM,0.0(EXP)0)
      TOX1=AMAX1(TOX1*UM,0.0(EXP)0)
      XSOURC=AMAX1(XSOURC*UM,0.0(EXP)0)
      XDRAIN=AMAX1(XDRAIN*UM,0.0(EXP)0)
      XOXR=AMAX1(XOXR*UM,0.0(EXP)0)
      XCHANL=AMAX1(XCHANL*UM,0.0(EXP)0)
      YMAX=AMAX1(YMAX*UM,0.0(EXP)0)
      XMAX=XCHANL+XSOURC+XDRAIN
      XGATE0=XGATE0*UM
      XGATE1=XGATE1*UM
      XOX0=XOX0*UM
      XOX1=XOX1*UM
#
#.....INTERCHANGE IF NECESSARY
      IF (XGATE0.GT.XGATE1) [ T=XGATE0; XGATE0=XGATE1; XGATE1=T ]
      IF (XOX0.GT.XOX1)     [ T=XOX0;   XOX0=XOX1;     XOX1=T   ]
#
#.....DONE
      RETURN
      END
```


File: &CROSS

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE CROSS (M)
#
#     CHARACTERIZE CROSS-SECTION AT Y-LOCATION M
#
*CALL TEMA
*CALL TCOMMN
#
      DATA UM/1.0(EXP)4/, USO/700.0(EXP)0/
#
      TWOVT=VT300K+VT300K
      YBARR=YPOS(M)*UM
#
#.....SEARCH FOR POTENTIAL MINIMUM WITHIN CROSS SECTION
      CALL PHMIN (M,PHIN,LMIN)
      XBARR=XPOS(LMIN)*UM
      PBARR=POTSI(1,M)-PHIN
#
#.....DETERMINE THE BASE REGION
      CALL WBASE (LMIN,M,NB1,NB2)
      WBARR=XPOS(NB2)-XPOS(NB1)
#
#.....SEARCH FOR SOURCE DEPLETION REGION LIMIT
      PHIN=POTSI(1,M)
      POTX=POTSI(2,M)
      NS2=1
      WHILE ((NS2.LT.NSOURC).AND.((PHIN-POTX).LE.TWOVT)) [
           NS2=NS2+1
           POTX=POTSI(NS2+1,M)
      ]
#
#.....SEARCH FOR DRAIN DEPLETION REGION LIMIT
      PHIN=POTSI(NXMAX,M)
      POTX=POTSI(NXMAX-1,M)
      ND2=NXMAX
      WHILE ((ND2.GT.NDRAIN).AND.((PHIN-POTX).LE.TWOVT)) [
           ND2=ND2-1
           POTX=POTSI(ND2-1,M)
      ]
#
#.....ESTIMATE THE CURRENT IF ANY
      CALL INTGR (LMIN,M,MB1,MB2,SUM)
      HBARR=YPOS(MB2)-YPOS(MB1)
      CINJO=Q*VT300K*USO/WBARR
      CINJ=ABS(CINJO*SUM)
#
#.....SCALE AND WRITE THE RESULTS
      CINJ=CINJ*UM
      CINJO=CINJO*CNI*CNI/ABS((CONC(LMIN,M)))
      HBY1=YPOS(MB1)*UM
      HBY2=YPOS(MB2)*UM
      HBARR=HBY2-HBY1
      WBX1=XPOS(NB1)*UM
      WBX2=XPOS(NB2)*UM
      WBARR=WBARR*UM
      IF (M.EQ.1) WRITE (KONSOL,1030)
      ELSE        WRITE (KONSOL,1035) YBARR,M
      WRITE (KONSOL,1000) PHIN,XBARR,LMIN,PBARR,CINJO,WBARR,WBX1,WBX2,NB1,NB2
      IF (CINJ.NE.0.0(EXP)0) WRITE (KONSOL,1020) HBARR,HBY1,HBY2,MB1,MB2,CINJ
      IF (M.EQ.1) [
           WS=WBX1-XPOS(NSOURC)*UM
           WD=XPOS(NDRAIN)*UM-WBX2
           WRITE (KONSOL,1050) WS,WD
1050       FORMAT(3X,"Source depletion width =",F6.3,"um"^
                 /,3X,"Drain  depletion width =",F6.3,"um")
      ]
#
#.....FORMAT STATEMENTS
1030  FORMAT("**At surface**")
1035  FORMAT("**Y-cross section at y =",F6.3,"um, (",I2,")")
1000  FORMAT(3X,"Potential minimum =",F7.3,"at X =",F6.3,"um, (",I2,")"^
            /,3X,"Barrier   height  =",F7.3^
            /,3X,"Current   density =",1PG10.3," Amp/cm2"^
            /,3X,"Barrier   width   =",0PF6.3,"um, from ",F6.3,"um to ",F6.3,^
                                   "um...(",I2,",",I2,")")
1020  FORMAT(3X,"Barrier   depth   =",0PF6.3,"um, from ",F6.3,"um to ",F6.3,^
                                   "um...(",I2,",",I2,")"^
```

File: &CROSS

```
      /,3X,"Current / width    =",1PG10.3,^
                                 " Amp/um, with USO=700 0 cm2/sec-V")
#
#.....DONE
      RETURN
      END
```


File: &DOPLC

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE DOPLC
#
#     LOCALIZED IMPLANT
#
*CALL TEMA
*CALL TCOMMN
#
      EQUIVALENCE (RNG, RANGE(2)),(SIGMA,STNDV(2)),(DIFF,DCOEF(2)),^
                  (TIME,DRVIN(2)),(CPK,  CPEAK(2)),(XJ,XJCT(2))
#
#
#.....STATEMENT FUNCTION
      ARGLM(X,Y)=SIGN(AMIN1(ABS(X),ABS(Y)),X)
#
#.....CONSTANTS
      DT=DIFF*TIME
      ONSQDT=0.5<EXP>0/SQRT(DT)
      STNDV2=SIGMA*SIGMA
      FOURDT=(STNDV2+STNDV2)+(DT+DT+DT+DT)
      B=RNG*SQRT((DT+DT)/FOURDT)/SIGMA
      IF (DT.NE.0.0<EXP>0) C=RNG/(FOURDT*B)
      DXJ=AMAX1((XJ-RNG),0.0<EXP>0)+SIGMA+SIGMA+SIGMA
      N1=LOCX(XIMPL0-DXJ)
      N2=LOCX(XIMPL1+DXJ)
      IF ((N1.LT.N2).AND.(N1.LT.NXMAX).AND.(N2.GT.1)) [
          N1=MAX0(N1,1)
          N2=MIN0(N2,NXMAX)
      ]
      A1=XIMPL0*ONSQDT
      A2=XIMPL1*ONSQDT
#
#.....IDEAL GAUSSIAN
      DO M=1,NYMAX [
          Y=YPOS(M)
          ARG=(Y-RNG)*(Y-RNG)/FOURDT
          CIMPL0=CPK*EXP(-ARGLM(ARG,85.0<EXP>0))
#
#.....DRIVE IN, INCLUDE SURFACE REFLECTION
          IF (DT.NE.0.0<EXP>0) [
              CY=C*Y
              CIMPL0=0.25<EXP>0*CIMPL0*(2.0<EXP>0+ERRFN(B+CY)+ERRFN(B-CY))
          ]
#
#.....LATERAL DIFFUSION
          IF ((N1.GE.1).AND.(N2.LE.NXMAX)) DO N=N1,N2 [
              IF (DT.NE.0.0<EXP>0) [
                  XA=XPOS(N)*ONSQDT
                  CIMPL=CIMPL0*(ERRFN(XA-A1)-ERRFN(XA-A2))
              ]
              ELSE  CIMPL=CIMPL0
              CONC(N,M)=CONC(N,M)+CIMPL
          ]
      ]
#
#.....DONE
      RETURN
      END
```

File: &DOPNG

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE DOPNG
#
#     GENERATE DOPING PROFILES
#
*CALL TEMA
*CALL TCOMMN
#
#.....LOAD SUBSTRATE CONCENTRATION
      DO N=1,NXMAX; DO M=1,NYMAX; CONC(N,M)=CSUB
#
#.....IMPLANTATION
      IF (DOSE(1).NE.0.0(EXP)0) CALL DOPWL
      IF (DOSE(2).NE.0.0(EXP)0) CALL DOPLC
      IF (DOSE(3).NE.0.0(EXP)0) CALL DOPSD
#
#.....CALCULATE IMPLANT RELATED PARAMETERS
      CALL PARMS
#
#.....DONE
      RETURN
      END
```


File: &DOPSI

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE DOPSI
#
#     GENERATE DOPING PROFILE USING SUPREM SAVE FILE
#
*CALL TEMA
*CALL TCOMMN
#
      DIMENSION NSTRNG(20)
      DIMENSION KDOPE(6),MESSG0(5,2),MESSG1(4,3)
#
      DATA LNMSG0/5/,^
           MESSG0/2HSu,2Hbs,2Htr,2Hat,2He ,^
                  2HIm,2Hpl,2Han,2Ht ,2H  /
      DATA LNMSG1/4/,^
           MESSG1/2Hov,2Her,2Hal,2Hl ,^
                  2Hlo,2Hca,2Hl',2Hd ,^
                  2Hsr,2Hc/,2Hdr,2Hn /
      DATA KDOPE /2HB ,2HA ,2HP ,2HS ,2H+ ,2H- /
      DATA KESC/033B/, KYES/2HY /, UM/1.0(EXP)-4/
#
#.....DEFINE SOFT KEYS
*IF HP2648A
      CALL SKEYP
*ENDIF
#
#.....GET SUBSTRATE TYPE
      WRITE (KONSOL,1000) (MESSG0(K,1),K=1,LNMSG0)
1000  FORMAT(5A2," dopant (B=f1,As=f2,Ph=f3,Sb=f4,",^
                          "+(N-type)=f5,-(P-type)=f6) ? _")
      READ  (KEYBRD,2000) KANSWR
*IF BATCH
      WRITE (KONSOL,2000) KANSWR
*ENDIF
      K=1; WHILE ((KANSWR.NE.KDOPE(K)).AND.(K.LE.6)) K=K+1
      IF (K.GT.6) GO TO 801
      IF ((K.EQ.1).OR.(K.EQ.6)) TYPE=-1.0(EXP)0
      ELSE                      TYPE= 1.0(EXP)0
      IF (TYPE.EQ.-1.0(EXP)0) KB=6
      ELSE                    KB=5
#
#.....CHECK IF UNIFORM SUBSTRATE
      WRITE (KONSOL,1010)
1010  FORMAT("Uniform substrate (Yes=f7/No=f8) ? _")
      READ  (KEYBRD,2000) KANSWR
*IF BATCH
      WRITE (KONSOL,2000) KANSWR
*ENDIF
2000  FORMAT(A1)
#
#.....IF UNIFORM, READ CSUB, LOAD CONCENTRATION ARRAY AND RESET FLAG
      IF (KANSWR.EQ.KYES) [
          WRITE (KONSOL,2010)
2010      FORMAT("Substrate concentration (in unit of 1E15 cm-3) ? _")
          READ  (KEYBRD,*) CSUB
*IF BATCH
          WRITE (KONSOL,2011) CSUB
2011      FORMAT(1PG10.3)
*ENDIF
          IF (CSUB.EQ.0.0(EXP)0) GO TO 802
          CSUB=SIGN(CSUB*1.0(EXP)15,TYPE)
          DO N=1,NXMAX; DO M=1,NYMAX; CONC(N,M)=CSUB
          KSUB=0
      ]
#
#.....IF NOT UNIFORM, SET THE FLAG
      ELSE [ KSUB=1; CSUB=0.0(EXP)0 ]
#
#.....ION IMPLANTATIONS
      DO KIMPL=1,3 [
          IF (KSUB.LE.0) [
              WRITE (KONSOL,2020) (MESSG1(K,KIMPL),K=1,LNMSG1)
2020          FORMAT("Any ",4A2," implant (Yes=f7/No=f8) ? _")
              READ  (KEYBRD,2000) KANSWR
*IF BATCH
              WRITE (KONSOL,2000) KANSWR
*ENDIF
```

```
*
*.....GET WINDOW OF LOCALIZED IMPLANT
              IF (KANSUR.EQ.KYES) [
                IF (KIMPL.EQ 2) [
                    WRITE (KONSOL,2025)
2025                FORMAT("Localized implant: from <> to <> (um) ? _")
                    READ  (KEYBRD,*) XIMPLO,XIMPL1
*IF BATCH
                    WRITE (KONSOL,2026) XIMPLO,XIMPL1
2026                FORMAT(1P2G15.3)
*ENDIF
                    XIMPLO=XIMPLO*UM
                    XIMPL1=XIMPL1*UM
                    IF (XIMPLO.GT.XIMPL1) [ T=XIMPLO;XIMPLO=XIMPL1;XIMPL1=T ]
                ]
*
*.....GET DOPANT TYPE
                WRITE (KONSOL,2030)
2030            FORMAT("Implant dopant (B=f1,As=f2,Ph=f3,Sb=f4,",^
                                  "+(N-type)=f5,-(P-type)=f6) ? _")
                READ  (KEYBRD,2000) KANSU1
*IF BATCH
                WRITE (KONSOL,2000) KANSU1
*ENDIF
                IF ((KANSU1.EQ.KDOPE(1)).OR.(KANSU1.EQ.KDOPE(6)))
                    DOPE(KIMPL)=6
                ELSE DOPE(KIMPL)=5
*
*.....GET STANDARD DEVIATION
                WRITE (KONSOL,2040)
2040            FORMAT("Implant STNDV(um) ? _")
                READ  (KEYBRD,*) SD
*IF BATCH
                WRITE (KONSOL,2041) SD
2041            FORMAT(1PG10.3)
*ENDIF
                STNDV(KIMPL)=SD*UM
            ]
          ]
          ELSE DOPE(1)=KB
*
*.....GET DATA FILE NAME
          IF ((KSUB.EQ.1).OR.(KANSUR.EQ.KYES)) [
              WRITE (KONSOL,3010)
3010          FORMAT("Data file name ? _")
              READ  (KEYBRD,3020) NSTRNG
3020          FORMAT(20A2)
*IF BATCH
              WRITE (KONSOL,3020) NSTRNG
*ENDIF
*
*.....GET COLUMN INDEX
              WRITE (KONSOL,3030)
3030          FORMAT("Which column ? _")
              READ  (KEYBRD,*) ICOLMN
*IF BATCH
              WRITE (KONSOL,3031) ICOLMN
3031          FORMAT(I2)
*ENDIF
*
*.....GENERATE PROFILE USING SUPREM SAVE FILE
              CALL REDSP (KIMPL,ICOLMN,NSTRNG)
          ]
          ELSE [ DOSE(KIMPL)=0.0(EXP)0; XJCT(KIMPL)=0.0(EXP)0 ]
*
*.....RESET SUBSTRATE FLAG
          IF (KSUB.EQ.1) KSUB=-1
      ]
*
*.....PREPROCESS PARAMETERS, RESET SUPREM FLAG
      CALL PARMS
      KSUPRM=0
*
*.....DONE
      RETURN
*
*.....ERROR
801   CONTINUE
      WRITE (KONSOL,8001)
8001  FORMAT("** Un-recognizable substrate dopant! Program terminated! **")
```



```
      GO TO 800
802   CONTINUE
      WRITE (KONSOL,8002)
8002  FORMAT("** Zero substrate doping concentration! Program terminated! **")
800   CONTINUE
*IF HP1000&!BATCH
      WRITE (KONSOL,8003) KESC
8003  FORMAT(R1,"n")
      CALL EXEC (6)
*ELSE
      STOP
*ENDIF
      END
```

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE DOPS2 (KIMPL,KSMAX,FDTX,SUPY,SUPC)
#
#     GENERATE IMPURITY PROFILE BASED ON SUPREM OUTPUT
#
*CALL TEMA
*CALL TCOMMN
#
      DIMENSION  SUPY(1),SUPC(1)
#
#.....CONSTANTS
      IF (KIMPL.GE.2) [
         SIGMA=SQRT(FDTX)
         ONSQDT=1.0(EXP)0/SIGMA
         DXJ=5.0(EXP)0*SIGMA
         IF (KIMPL.EQ.2) [
            N1=LOCX(XIMPLO-DXJ)
            N2=LOCX(XIMPL1+DXJ)
            IF ((N1.LT.N2).AND.(N1.LT.NXMAX).AND.(N2.GT.1)) [
               N1=MAX0(N1,1)
               N2=MIN0(N2,NXMAX)
               NINT=(N1+N2)/2
            ]
            A1=XIMPLO*ONSQDT
            A2=XIMPL1*ONSQDT
         ]
         ELSE [
            N1=LOCX(DXJ)
            N2=LOCX(XCHANL-DXJ)
            N1=MIN0(N1,NXMAX)
            N2=MAX0(N2,1)
            NINT=1
         ]
      ]
      ELSE NINT=NXMAX/2
#
#.....LINEAR INTERPOLATION
      SUM=0.0(EXP)0
      KS=1
      KLIM=0
      RNG=RANGE(KIMPL)
      KPEAK=LOCY(RNG)
      DO M=1,NYMAX [
         Y=YPOS(M)
         YS=SUPY(KS)
         YS1=SUPY(KS+1)
         REPEAT [
            IF ( ((Y.GE.YS).AND.(Y.LE.YS1)).OR.(KS.GE.KSMAX) ) [
               KGO=0
               IF (KLIM.EQ.0) [
                  KLIM=1
                  NINT=M-1
                  COLD=CIMPLO
               ]
            ]
            ELSE [
               KS=KS+1
               YS=YS1
               YS1=SUPY(KS+1)
               KGO=1
               IF (KS.GT.KPEAK)
                  SUM=SUM+0.5(EXP)0*(SUPC(KS)+SUPC(KS-1))*(YS1-YS)
            ]
         ] UNTIL (KGO.EQ.0)
         CIMPLO=(SUPC(KS+1)-SUPC(KS))*(Y-YS)/(YS1-YS)+SUPC(KS)
#
#.....OVERALL IMPLANT
         IF (KIMPL.EQ.1) DO N=1,NXMAX; CONC(N,M)=CONC(N,M)+CIMPLO
#
#.....LOCALIZED IMPLANT
         IF (KIMPL.EQ.2) [
            IF ((N1.GE.1).AND.(N2.LE.NXMAX)) [
               DO N=N1,N2 [
                  IF (FDTX.NE.0.0(EXP)0) [
                     XA=XPOS(N)*ONSQDT
                     CIMPL=CIMPLO*(ERRFN(XA-A1)-ERRFN(XA-A2))
                  ]
```



```
                  ELSE CIMPL=CIMPLO
                  CONC(N,M)=CONC(N,M)+CIMPL
               ]
            ]
         ]
#
#.....SIDE DIFFUSION AT SOURCE JUNCTION
         IF (KIMPL.EQ.3) [
            DO N=1,N1 [
               IF (FDTX.NE.0.0(EXP)0) [
                  ARG=XPOS(N)*ONSQDT
                  CIMPL=0.5(EXP)0*CIMPLO*(1.0(EXP)0-ERRFN(ARG))
               ]
               ELSE CIMPL=CIMPLO
               CONC(N,M)=CONC(N,M)+CIMPL
            ]
#
#.....SIDE DIFFUSION AT DRAIN JUNCTION
            DO N=N2,NXMAX [
               IF (FDTX.NE.0.0(EXP)0) [
                  ARG=-(XPOS(N)-XCHANL)*ONSQDT
                  CIMPL=0.5(EXP)0*CIMPLO*(1.0(EXP)0-ERRFN(ARG))
               ]
               ELSE CIMPL=CIMPLO
               CONC(N,M)=CONC(N,M)+CIMPL
            ]
         ]
      ]
#
#.....DONE
      IF (KLIM.EQ.-1)
         SUM=SUM+0.5(EXP)0*(CONC(NINT,NINT)+CIMPLO)*(YMAX-YPOS(NINT))
      SUM=SUM+SUM
      RETURN
      END
```

File: &DOPSD

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE DOPSD
#
#     SOURCE AND DRAIN IMPLANTATION
#
*CALL TEMA
*CALL TCOMMN
#
      EQUIVALENCE (RNG, RANGE(3)),(SIGMA,STNDV(3)),(DIFF,DCOEF(3)),^
                  (TIME,DRVIN(3)),(CPK,  CPEAK(3)),(XJ,  XJCT(3))
#
#.....STATEMENT FUNCTION
      ARGLM(X,Y)=SIGN(AMIN1(ABS(X),ABS(Y)),X)
#
#.....CONSTANTS
      DT=DIFF*TIME
      ONSQDT=0.5(EXP)0/SQRT(DT)
      STNDV2=SIGMA*SIGMA
      FOURDT=(STNDV2+STNDV2)+(DT+DT+DT+DT)
      B=RNG*SQRT((DT+DT)/FOURDT)/SIGMA
      IF (DT.NE.0.0(EXP)0) C=RNG/(FOURDT*B)
      DXJ=AMAX1((XJ-RNG),0.0(EXP)0)+SIGMA+SIGMA+SIGMA
      NS=LOCX(DXJ)
      ND=LOCX(XCHANL-DXJ)
#
#.....IDEAL GUASSIAN
      DO M=1,NYMAX [
         Y=YPOS(M)
         ARG=(Y-RNG)*(Y-RNG)/FOURDT
         CIMPLO=CPK*EXP(-ARGLM(ARG,85.0(EXP)0))
         IF (DT.NE.0.0(EXP)0) [
            CY=C*Y
            CIMPLO=0.25(EXP)0*CIMPLO*(2.0(EXP)0+ERRFN(B+CY)+ERRFN(B-CY))
         ]
#
#.....SIDE DIFFUSION AT SOURCE JUNCTION
         DO N=1,NS [
            IF (DT.NE.0.0(EXP)0) [
               ARG=XPOS(N)*ONSQDT
               CIMPL=CIMPLO*(1.0(EXP)0-ERRFN(ARG))
            ]
            ELSE CIMPL=CIMPLO
            CONC(N,M)=CONC(N,M)+CIMPL
         ]
#
#.....SIDE DIFFUSION AT DRAIN JUNCTION
         DO N=ND,NXMAX [
            IF (DT.NE.0.0(EXP)0) [
               ARG=(XPOS(N)-XCHANL)*ONSQDT
               CIMPL=CIMPLO*(1.0(EXP)0+ERRFN(ARG))
            ]
            ELSE CIMPL=CIMPLO
            CONC(N,M)=CONC(N,M)+CIMPL
         ]
      ]
#
#.....DONE
      RETURN
      END
```


File: &DOPWL

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE DOPWL
#
#     OVERALL IMPLANT
#
*CALL TEMA
*CALL TCOMMN
#
      EQUIVALENCE (RNG, RANGE(1)),(SIGMA,STNDV(1)),(DIFF,DCOEF(1)),^
                  (TIME,DRVIN(1)),(CPK,  CPEAK(1))
#
#.....STATEMENT FUNCTION
      ARGLM(X,Y)=SIGN(AMIN1(ABS(X),ABS(Y)),X)
#
#.....CONSTANTS
      DT=DIFF*TIME
      STNDV2=SIGMA*SIGMA
      FOURDT=(STNDV2+STNDV2)+(DT+DT+DT+DT)
      B=RNG*SQRT((DT+DT)/FOURDT)/SIGMA
      IF (DT.NE.0.0(EXP)0) C=RNG/(FOURDT*B)
#
#.....IDEAL GAUSSIAN
      DO M=1,NYMAX [
         Y=YPOS(M)
         ARG=(Y-RNG)*(Y-RNG)/FOURDT
         CIMPL=CPK*EXP(-ARGLM(ARG,85.0(EXP)0))
#
#.....DRIVE-IN, INCLUDE THE REFLECTION AT THE SURFACE
         IF (DT.NE.0.0(EXP)0) [
            CY=C*Y
            CIMPL=CIMPL*(1.0(EXP)0+0.5(EXP)0*(ERRFN(B+CY)+ERRFN(B-CY)))
         ]
#
#.....LOAD CONCENTRATION ARRAY
         DO N=1,NXMAX] CONC(N,M)=CONC(N,M)+CIMPL
      ]
#
#.....DONE
      RETURN
      END
```

File: &ERRFN

```
*CALL TWDEFN
      FUNCTION ERRFN(X)
*
*     CLOSE FORM APPROXIMATION OF ERROR FUNCTION
*
*.....COEFFICIENTS OF THE EQUIVALENT POLYNOMIAL
      DATA COEFF1/0.0705230784<EXP>0/,COEFF2/0.0422820123<EXP>0/,^
           COEFF3/0.0092705272<EXP>0/,COEFF4/0.0001520143<EXP>0/,^
           COEFF5/0.0002765672<EXP>0/,COEFF6/0.0000430638<EXP>0/
*
*.....CALCULATE THE POWERS OF THE ABSOLUTE X
      X1=ABS(X)
      X2=X1*X1
      X3=X2*X1
      X4=X3*X1
      X5=X4*X1
      X6=X5*X1
*
*.....TERMS USED IN THE CLOSE FORM APPROXIMATION
      ARG1=1.0<EXP>0+COEFF1*X1+COEFF2*X2+COEFF3*X3+^
                     COEFF4*X4+COEFF5*X5+COEFF6*X6
      ARG2=ARG1*ARG1
      ARG4=ARG2*ARG2
      ARG8=ARG4*ARG4
      ARGH=ARG8*ARG8
*
*.....THE CLOSE FORM APPROXIMATION
      ERRFN=1.0<EXP>0-1.0<EXP>0/ARGH
      IF (X.LT.0.0<EXP>0) ERRFN=-ERRFN
*
*.....DONE
      RETURN
      END
```


File: &GETPA

```
*CALL TWDEFN
      SUBROUTINE GETPA
*
*     READ IN PARAMETERS
*
*CALL TCONMN
*
      DIMENSION MESSAG(9,2)
*
      DATA KYES/2HY /,LENMSG/9/,^
           MESSAG/2Hde,2Hvi,2Hce,2H s,2Htr,2Huc,2Htu,2Hre,2H? ,^
                  2Hdo,2Hpi,2Hng,2H p,2Hro,2Hfi,2Hle,2H? ,2H  /
*
*.....CHECK INPUT OPTION
      WRITE (KONSOL,1000)
1000  FORMAT("Input from the console (Yes=f7/No=f8) ? _")
      READ  (KEYBRD,2000) KANSWR
2000  FORMAT(A1)
*IF BATCH
      WRITE (KONSOL,2000) KANSWR
*ENDIF
*
*.....INPUT FROM CONSOLE
      NEWNSH=1
      KREAD=1
      IF (KANSWR.EQ.KYES) [
          KREAD=0
*
*.....READ IN STRUCTURE PARAMETERS
          IF (KLOOP.NE.1) [
              WRITE (KONSOL,3000) (MESSAG(K,1),K=1,LENMSG)
3000          FORMAT("Redefine the ",9A2," (Yes=f7/No=f8) ?_")
              READ  (KEYBRD,2000) KANSWR
*IF BATCH
              WRITE (KONSOL,2000) KANSWR
*ENDIF
          ]
          IF (KANSWR.EQ.KYES) [
              IF (KLOOP.EQ.1) CALL REDGM
              ELSE            CALL RDFGM
          ]
          ELSE NEWNSH=0
*
*.....READ IN PROFILE PARAMETERS
          KANSWR=KYES
          IF (KLOOP.NE.1) [
              WRITE (KONSOL,3000) (MESSAG(K,2),K=1,LENMSG)
              READ  (KEYBRD,2000) KANSWR
*IF BATCH
              WRITE (KONSOL,2000) KANSWR
*ENDIF
          ]
          IF (KANSWR.EQ.KYES) NEWDOP=1
          ELSE                NEWDOP=0
          IF (NEWDOP.EQ.1) [
              WRITE (KONSOL,3010)
              READ  (KONSOL,2000) KANSWR
3010          FORMAT("Use SUPREM generated profile (Yes=f7/No=f8) ? _")
              IF (KANSWR.EQ.KYES) KSUPRM=1
              ELSE                KSUPRM=0
          ]
          IF ((NEWDOP.EQ.1).AND.(KSUPRM.EQ.0)) [
              IF (KLOOP.EQ.1) CALL REDDP
              ELSE            CALL RDFDP
          ]
      ]
*
*.....INPUT FROM FILE
      ELSE CALL REDFI
*
*.....DONE
      RETURN
      END
```

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE INDEP (LOC,KTYPE,TOX,PHIGB)
#
#     INITIALIZE 1-D POTENTIAL ARRAY IN DEPLETION CHANNEL REGION
#
*CALL TEMA
*CALL TCOMMN
#
#.....CALCULATE SOME USEFUL QUANTITIES
      COX=EPSIO2/TOX
      PHJCT=AMAX1(PHIB+PHIF(KTYPE)+VSB,0.0(EXP)0)
      GAMMA=GAMM(KTYPE)/COX
      ALFA=ALPHA(KTYPE)
      PHIC=PHIMP(KTYPE)
      IF (KTYPE.LE.2) [
           WIMP=XJCT(KTYPE)
           CIMP=CSTEP(KTYPE)
           VIMP=Q*CIMP/COX
      ]
      ELSE [
           WIMP=WSURF(1)
           CIMP=CSURF(1)
           VIMP=Q*CIMP/COX
      ]
      CD=CIMP-CSUB
      CDONCA=CD/CSUB
      WN=ALPHB*SQRT(PHJCT/(CDONCA*(1.0(EXP)0+CDONCA)))
      QN=Q*CD*AMIN1(WN,WIMP)
      WP=ALPHB*SQRT(PHJCT/(1.0E0+1.0(EXP)0/CDONCA))
      QP=Q*CSUB*WP
#
#.....DETERMINE FLAT-BAND CONDITION
      IF (WN.LE.WIMP) VGBFB=VFB+PHJCT
      ELSE            VGBFB=VFB+(QP-QN)/COX-VIMP+ALPHB*(WIMP+WP)*(WIMP+WP)
      DELCG=COX*(PHIGB-VGBFB)/Q
#
#.....SURFACE ACCUMULATION
      IF (PHIGB.GE.VGBFB) DELPHS=VT300K*ALOG(DELCG/CD-1.0(EXP)0)
#
#.....SURFACE DEPLETION
      ELSE [
           IF (WN.LE.WIMP) [
                KP=0
                WG=-DELCG/CD
                IF (WG.LT.WIMP) DELPHS=(-ALFA+ALPHB)*WG*WG
                ELSE            KP=1
           ]
           ELSE KP=1
           IF (KP.EQ.1) [
                WG=-DELCG/CSUB
                IF (WG.LE.WP) [
                     DELPHS=0.0(EXP)0
                     WP=WP-WG
                ]
#
#.....ACCUMULATION IN SUBSTRATE
                ELSE [
                     DELPHS=VT300K*ALOG(-DELCG/CSUB-1.0(EXP)0)
                     WP=0.0(EXP)0
                ]
           ]
      ]
#
#.....POTENTIALS DUE TO DEPLETED SUBSTRATE
      YN=AMAX1(WIMP-WN,0.0(EXP)0)
      YP=WIMP+WP
      MN=MAX0(LOCY(YN),1)
      MP=LOCY(YP)
      IF (YN.NE.YPOS(MN)) MN=MN+1
      IF (WP.NE.0.0(EXP)0) [
           DO M=MN,MP [
                Y=YPOS(M)
                POTSI(LOC,M)=ALPHB*(YP-Y)*(YP-Y)
                CARRIE(LOC,M)=0.0(EXP)0
           ]
           PHIP=ALPHB*WP*WP
      ]
```



```
#
#.....POTENTIAL DUE TO ACCUMULATED SUBSTRATE
      ELSE [
           DO M=MN,MP [
                POTSI(LOC,M)=DELPHS
                CARRIE(LOC,M)=0.0(EXP)0
           ]
           PHIP=DELPHS
      ]
#
#.....POTENTIAL DUE TO DEPLETED N-REGION
      MI=LOCY(WIMP)
      DO M=MN,MI [
           Y=YPOS(M)
           POTSI(LOC,M)=POTSI(LOC,M)-ALFA*(WIMP-Y)*(WIMP-Y)
      ]
      PHIN=PHIP+ALFA*WN*WN
#
#.....BURIED CHANNEL
      MN=MN-1
      IF (MN.GE.1) [
           DO M=1,MN [
                POTSI(LOC,M)=PHIN
                CARRIE(LOC,M)=-CONC(LOC,M)
           ]
      ]
#
#.....DEPLETED SURFACE
      IF (DELPHS.LT.0.0(EXP)0) [
           MG=LOCY(WG)
           ALFS=-ALFA+ALPHB
           IF (MG.GE.1) [
                DO M=1,MG [
                     Y=YPOS(M)
                     POTSI(LOC,M)=POTSI(LOC,M)+ALFS*(Y-WG)*(Y-WG)
                     CARRIE(LOC,M)=0.0(EXP)0
                ]
           ]
      ]
#
#.....ACCUMULATED SURFACE
      IF (DELPHS.GT.0.0(EXP)0) [
           POTSI(LOC,1)=POTSI(LOC,1)+DELPHS
           CARRIE(LOC,M)=-CONC(LOC,M)-Q*DELCG
      ]
#
#.....NEUTRAL SUBSTRATE
      MX=MP+1
      IF (MX.LE.NYMAX) [
           DO M=MX,NYMAX [
                POTSI(LOC,M)=0.0(EXP)0
                CARRIE(LOC,M)=-CONC(LOC,M)
           ]
      ]
#
#.....DONE
      RETURN
      END
```

```
*CALL TUDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE INENH (LOC,KTYPE,TOX,PHIGB)
#
#     INITIALIZE 1-D POTENTIAL ARRAY IN ENHANCEMENT CHANNEL REGION
#
*CALL TEMA
*CALL TCOMMN
#
#.....CALCULATE SOME USEFUL QUANTITIES
      COX=EPSIO2/TOX
      IF (KTYPE.EQ.0) {                         # case of uniform substrate
          PHIBI=AMAX1(2.5(EXP)0*PHIB+VSB,0.0(EXP)0)    # bit-in potential at vth
          GAMMA=GAMMB/COX                       # effective body-effect coeff
          ALFA=ALPHB                            # effective depl-rgn width coeff
          VBC=0.0(EXP)0                             # knee-voltage of vsb
          PHIC=0.0(EXP)0                            # built-in potential at vbc
          WIMP=0.0(EXP)0                            # depth of implant region
          CIMP=0.0(EXP)0                            # concentration of implant
          VIMP=0.0(EXP)0                            # shift in vth by implant
      }
      ELSE {                                    # case of non-uniform substrate
          PHIBI=AMAX1(PHIB+PHIF(KTYPE)*1.5(EXP)0+VSB,0.0(EXP)0)
          GAMMA=GAMM(KTYPE)/COX
          ALFA=ALPHA(KTYPE)
          VBC  =VBCRT(KTYPE)
          PHIC=PHIMP(KTYPE)
          IF (KTYPE.LE.2) {
              WIMP=XJCT(KTYPE)
              CIMP=-CSTEP(KTYPE)
              VIMP=-Q*CIMP/COX
          }
          ELSE {
              WIMP=WSURF(1)
              CIMP=-CSURF(1)
              VIMP=-Q*CIMP/COX
          }
      }
      IF (VSB.LE.VBC)VTH=VFB+PHIBI+GAMMA*SQRT(PHIBI)
      ELSE           VTH=VFB+PHIBI+VIMP+GAMMB*SQRT(AMAX1(PHIBI-PHIC,0.0(EXP)0))
#
#.....WEAK INVERSION CASE
      IF (PHIGB.LE.VTH) {
          IF (VSB.LE.VBC) {
              G=GAMMA
              VGBEFF=PHIGB-VFB
              DPHI=0.0(EXP)0
              VTEFF=VT300K
              IF (KTYPE.NE.0) {
                  DPHI=PHIF(KTYPE)-PHIB
                  VTEFF=VT300K*CSUB/(CIMP+CSUB)
              }
          }
          ELSE {
              G=GAMMB
              VGBEFF=PHIGB-VFB+PHIC
              DPHI=0.0(EXP)0
              VTEFF=VT300K+PHIC
          }
          G2OM2=0.5(EXP)0*G*G
          SQARG=AMAX1(VGBEFF+0.5(EXP)0*G2OM2-DPHI-VTEFF,0.0(EXP)0)
          PHIS=VGBEFF+G2OM2-G*SQRT(SQARG)
      }
#
#.....STRONG INVERSION CASE
      ELSE {
          ARG=(PHIGB-VFB-PHIBI)*(PHIGB-VFB-PHIBI)/(GAMMA*GAMMA*VT300K)
          PHIS=PHIBI+VT300K*ALOG(AMAX1(ARG,1.0(EXP)-10))
      }
#
#.....ASSIGN POTENTIAL ARRAY
      POTSI(LOC,1)=PHIS
      CARRIE(LOC,1)=0.0(EXP)0
      WDPL=SQRT(PHIS/(ALPHB+ALPHA(KTYPE)))
      IF (WDPL.GT.WIMP) WDPL=SQRT(AMAX1(PHIS-PHIC,0.0(EXP)0)/ALPHB)
      DO M=2,NYMAX {
          Y=YPOS(M)
          IF (Y.LT.WDPL) {
```



```
              POTSI(LOC,M)=ALPHB*(WDPL-Y)*(WDPL-Y)
              CARRIE(LOC,M)=0.0(EXP)0
              IF (Y.LT.WIMP) POTSI(LOC,M)=POTSI(LOC,M)+ALFA*(WIMP-Y)*(WIMP-Y)
          }
          ELSE {
              POTSI(LOC,M)=0.0(EXP)0
              CARRIE(LOC,M)=-CONC(LOC,M)
          }
      }
#
#.....DONE
      RETURN
      END
```

File: &INEQU

```
*CALL TUDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE INEQU (N1,N2,NR,KMSG)
#
#     LOAD POTENTIAL AND CARRIER ARRAYS FROM N1 TO N2 AS NR
#
*CALL TEMA
*CALL TCOMMN
#
      DATA UM/1.0(EXP)4/
#
      IF (N1.NE.N2) DO N=N1,N2; DO M=1,NYMAX [
           POTSI (N,M)=POTSI (NR,M)
           CARRIE(N,M)=CARRIE(NR,M)
      ]
      ELSE DO M=1,NYMAX [
           POTSI(N1,M)=POTSI(NR,M)
           CARRIE(N1,M)=CARRIE(NR,M)
      ]
      X1=XPOS(N1)*UM
      X2=XPOS(N2)*UM
      XR=XPOS(NR)*UM
      IF (KMSG.NE.0) WRITE (KONSOL,1000) X1,N1,X2,N2,XR,NR
1000  FORMAT("Equal  potential region between ",^
             F6.2,"um(",I2,") and ",F6.2,"um(",I2,") as ",F6.2,"um(",I2,")")
#
#.....DONE
      RETURN
      END
```


File: &INITL

```
*CALL TUDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE INITL (KMSG)
#
#     INITIALIZE POTENTIAL DISTRIBUTION
#
*CALL TEMA
*CALL TCOMMN
#
      DATA UM/1.0(EXP)4/, KYES/2HY /
#
      KPASS=1
#
#.....CHECK IF ENVOKE LOCAL ITERATION
      KLITR=51
      IF (RELAX1.EQ.1) [
           WRITE (KONSOL,1030)
           READ  (KEYBRD,2000) KANSUR
*IF BATCH
           WRITE (KONSOL,2000) KANSUR
*ENDIF
           IF (KANSUR.EQ.KYES) [
                WRITE (KONSOL,1040)
                READ  (KEYBRD,*) KLITR
*IF BATCH
                WRITE (KONSOL,1041) KLITR
1041            FORMAT(I4)
*ENDIF
                WRITE (KONSOL,1050)
                READ  (KEYBRD,2000) KANSUR
*IF BATCH
                WRITE (KONSOL,2000) KANSUR
*ENDIF
                IF (KANSUR.EQ.KYES) KIMSGO=1
                ELSE                KIMSGO=0
           ]
      ]
1030  FORMAT("Envoke local iteration (Yes=f7/No=f6) ? _")
1040  FORMAT("Start from <> iteration <> ?? _")
1050  FORMAT("Convergence information per grid (Yes=f7/No=f8) ? _")
2000  FORMAT(A1)
#
#.....DEFINE SURFACE DEPLETION REGIONS
      CALL WDEPL (WS0,WS1,WD0,WD1)
      IF ((NSOURC.GE.1).AND.(ABS(DOSE(3)).GT.1.0(EXP)0)) [
           XS0=XPOS(NSOURC)-WS0
           NS0=LOCX(XS0)
           XS1=XPOS(NSOURC)+WS1
           NS1=LOCX(XS1)
           IF (XS1.NE.XPOS(NS1)) NS1=NS1+1
           NS0=MAX0(NS0,2)
           NS1=MIN0(NS1,NDRAIN-1)
      ]
      ELSE [
           XS1=XPOS(NSOURC)
           XS0=XS1-WS1
           NS1=NSOURC
           NS0=LOCX(XS0)
           NS0=MAX0(NS0,2)
      ]
      IF ((NDRAIN.LE.NXMAX).AND.(ABS(DOSE(3)).GT.1.0(EXP)0)) [
           XD0=XPOS(NDRAIN)+WD0
           ND0=LOCX(XD0)
           XD1=XPOS(NDRAIN)-WD1
           ND1=LOCX(XD1)
           IF (XD0.NE.XPOS(ND0)) ND0=ND0+1
           ND0=MIN0(ND0,NXMAX-1)
           ND1=MAX0(ND1,NSOURC+1)
      ]
      ELSE [
           XD1=XPOS(NDRAIN)
           XD0=XD1+WD1
           ND1=NDRAIN
           ND0=LOCX(XD0)
           ND0=MIN0(ND0,NXMAX-1)
      ]
      XC=0.5(EXP)0*(XPOS(NS1)+XPOS(ND1))
      NC=LOCX(XC)
```

File: &INITL

```
      NC=MAX0(NC,NS1)
#
#.....INITIALIZE BOUNDARY PLANE AT SOURCE END
      CALL POSHI (1,KPASS,KLITR,KMSG)
#
#.....INITIALIZE MIDDLE CROSS-SECTION OF CHANNEL REGION IF EXISTS;
#           OTHERWISE, INNER BOUNDARY OF DRAIN-CONTROLLED DEPLETION REGION
      IF (ND1.GT.NS1) CALL POSHI (NC, KPASS,KLITR,KMSG)
      ELSE            CALL POSHI (ND1,KPASS,KLITR,KMSG)
#
#.....INITIALIZE BOUNDARY PLANE AT DRAIN END
      CALL POSHI (NXMAX,KPASS,KLITR,KMSG)
#
#.....LOAD POTENTIAL AND CARRIER ARRAYS IN SOURCE, DRAIN AND CHANNEL REGIONS
      CALL INEQU (2,NS0,1,KMSG)
      CALL INEQU (ND0,NXMAX-1,NXMAX,KMSG)
      IF (ND1.GT.NS1) [
          IF (NS1.NE.NC) CALL INEQU (NS1,NC-1,NC,KMSG)
          IF (ND1.NE.NC) CALL INEQU (NC+1,ND1,NC,KMSG)
      ]
#
#.....SOURC- AND DRAIN-CONTROLLED DEPLTION REGIONS
      CALL INLAT (ND1,ND0)
      XD0=XPOS(ND0)*UM
      XD1=XPOS(ND1)*UM
      IF (KMSG.GT.0) WRITE (KONSOL,3000) XD1,ND1,XD0,ND0
      IF (ND1.GT.NS1) CALL INLAT (NS0,NS1)
      ELSE [
          CALL INLAT (NS0,NC)
          NS1=NC
          ND1=NC
      ]
      XS0=XPOS(NS0)*UM
      XS1=XPOS(NS1)*UM
      IF (KMSG.GT.0) WRITE (KONSOL,3000) XS0,NS0,XS1,NS1
3000  FORMAT ("Lateral depletion layer between ",F6.2,"um(",I2,") and "^
                                              ,F6.2,"um(",I2,")")
#
#.....INITIALIZE OXIDE
      CALL INOXD
#
#.....DONE
      RETURN
      END
```


File: &INLAT

```
*CALL TUDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE INLAT (N1,N2)
#
#     INITIALIZE LATERAL DEPLETION REGIONS
#
*CALL TEMA
*CALL TCOMMN
#
#.....DETERMINE POLARITY
      POT1=POTSI(N1,1)
      POT2=POTSI(N2,1)
      IF (POT1.NE.POT2) [
          WDS=ABS((XPOS(N2))-(XPOS(N1)))
          ALPHAX=WDS*WDS/ABS(POT1-POT2)
#
#.....ASSIGN THE POTENTIAL
          DO M=1,NYMAX [
              POT1=POTSI(N1,M)
              POT2=POTSI(N2,M)
              X1=XPOS(N1)
              X2=XPOS(N2)
              XTOTL=ABS(X2-X1)
              IF (POT1.GT.POT2) [
                  NL=N1; NL1=N1+1;  NR=N2;    NR1=N2-1;    ND=1
                  XL=X1; POTL=POT1; POTR=POT2
              ]
              ELSE [
                  NL=N2; NL1=N2-1;  NR=N1;    NR1=N1+1; ND=-1
                  XL=X2; POTL=POT2; POTR=POT1
              ]
              WDY=SQRT(ALPHAX*(POTL-POTR))
              IF (ND.GT.0) FOR (N=NL1; N<=NR1; N=N+ND) [
                  X=ABS((XPOS(N))-XL)
                  IF (X.LT.WDY) [
                      POTSI(N,M)=(WDY-X)*(WDY-X)/ALPHAX+POTR
                      CARRIE(N,M)=0.0<EXP>0
                  ]
                  ELSE [
                      POTSI(N,M)=POTR
                      CARRIE(N,M)=CARRIE(NR,M)
                  ]
              ]
              ELSE FOR (N=NL1; N>=NR1; N=N+ND) [
                  X=ABS((XPOS(N))-XL)
                  IF (X.LT.WDY) [
                      POTSI(N,M)=(WDY-X)*(WDY-X)/ALPHAX+POTR
                      CARRIE(N,M)=0.0<EXP>0
                  ]
                  ELSE [
                      POTSI(N,M)=POTR
                      CARRIE(N,M)=CARRIE(NR,M)
                  ]
              ]
          ]
      ]
#
#.....EQUALPOTENTIAL
      ELSE DO N=N1,N2; DO M=1,NYMAX [
          POTSI(N,M)=POTSI(N1,M)
          CARRIE(N,M)=CARRIE(N1,M)
      ]
#
#.....DONE
      RETURN
      END
```

File: &INOXD

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE INOXD
#
#     INITIALIZE OXIDE POTENTIAL ARRAYS
#
*CALL TEMA
*CALL TCOMMN
#
#.....UNDER GATE ELECTRODE
      IF (NGATE0.NE.NGATE1) DO N=NGATE0,NGATE1 [
          PHIS=POTSI(N,1)
          DELOX1=DELOX(N,1)
          DELOXT=DELOX1+DELOX(N,2)
          POTOX(N,2)=VGB
          POTOX(N,1)=(VGB-PHIS)*DELOX1/DELOXT+PHIS
      ]
#
#.....OUTSIDE GATE ELECTRODE
      NG0=NGATE0-1
      NG1=NGATE1-1
      NXMAX1=NXMAX-1
      DO KK=1,2 [
          IF (KK.EQ.1) [ NN1=2;   NN2=NG0    ]
          ELSE         [ NN1=NG1; NN2=NXMAX1 ]
          DO N=NN1,NN2 [
              PHIS=POTSI(N,1)
              DELOX1=DELOX(N,1)
              DELOXT=DELOX1+DELOX(N,2)
              POTOX1=CSOX1(N,1)*(PHIS+PHIS)
              POTOX(N,1)=POTOX1
              POTOX(N,2)=CSOX1(N,2)*POTOX1
          ]
      ]
#
#.....DONE
      RETURN
      END
```


File: &INSAD

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE INSAD (LOC,PHIBI)
#
#     INITIALIZE POTENTIAL ARRAY IN SOURCE AND/OR DRAIN
#
*CALL TEMA
*CALL TCOMMN
#
#.....ASSIGN POTENTIAL ARRAY BASED ON ABRUPT JUNCTION ASSUMPTION
      WDEPL=SQRT(AMAX1(PHIBI,0.0<EXP>0)/ALPHB)
      DO M=1,NYMAX [
          CONCX=CONC(LOC,M)
          TPX=SIGN(1.0<EXP>0,CONCX)
          IF (TPX.NE.TYPE) [
              POTSI(LOC,M)=PHIBI
              CARRIE(LOC,M)=-CONCX
          ]
          ELSE [
              Y=YPOS(M)
              IF (Y.LT.WDEPL) [
                  POTSI(LOC,M)=(WDEPL-Y)*(WDEPL-Y)*ALPHB
                  CARRIE(LOC,M)=0.0<EXP>0
              ]
              ELSE [
                  POTSI(LOC,M)=0.0<EXP>0
                  CARRIE(LOC,M)=-CONCX
              ]
          ]
      ]
#
#.....DONE
      RETURN
      END
```

File: &INTGR

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
#
      SUBROUTINE INTGR (N,M,MB1,MB2,SUM)
#
      INTEGRATE CONDUCTION CARRIERS
#
*CALL TEMA
*CALL TCOMMN
#
#.....LOCATE UPPER BOUNDARY
      MB1=M
      TYPC=SIGN(1.0<EXP>0,(CARRIE(N,M)))
      WHILE ((MB1.GT.1).AND.(TYPC.EQ.TYPE)) [
           MB1=MB1-1
           TYPC=SIGN(1.0<EXP>0,(CARRIE(N,MB1)))
      ]
      IF (MB1.NE.M) MB1=MB1+1
#
#.....INTEGRATE UP TO THE ZERO CARRIER POINT
      MB2=MB1
      CARRO=CARRIE(N,MB2)
      SUM=0.0<EXP>0
      TYPC=SIGN(1.0<EXP>0,(CARRIE(N,M)))
      WHILE ((MB2.LT.NYMAX).AND.(TYPC.EQ.TYPE)) [
           MB2=MB2+1
           OLDCAR=CARRO
           CARRO=CARRIE(N,MB2)
           TYPC=SIGN(1.0<EXP>0,CARRO)
           SUM=SUM+0.5<EXP>0*DELY(MB2-1)*(OLDCAR+CARRO)
      ]
#
#.....DONE
      RETURN
      END
```


File: &ITER8

```
*CALL TWDEFN
      SUBROUTINE ITER8 (PHI,PHIA,QOVRE,CDOPE,ABSTL,NITR8,KITR)
#
#     NEWTON-RALPHSON ITERATION AT A MESH POINT
#
*CALL TCOMMN
#
#.....STATEMENT FUNCTION
      ARGLM(X,Y)=SIGN(AMIN1(ABS(X),ABS(Y)),X)
#
      NITR8=1
      KGO=1
      REPEAT [
           ARGP=ARGLM(-(PHI-PHIFP)/VT300K,85.0<EXP>0)
           ARGN=ARGLM( (PHI-PHIFN)/VT300K,85.0<EXP>0)
           CONCP=CSUB*EXP(ARGP)
           CONCN=CMINRO*EXP(ARGN)
           FVAL=PHI-PHIA-QOVRE*(CDOPE+CONCP-CONCN)
           DVAL=1.0<EXP>0+QOVRE*(CONCP+CONCN)/VT300K
           DELPHI=FVAL/DVAL
           IF (ABS(DELPHI).GT.ABSTL) [
                PHI=PHI-DELPHI
                NITR8=NITR8+1
                IF (NITR8.GE.KITR) [
                     NITR8=NITR8-1
                     KGO=0
                ]
           ]
           ELSE KGO=0
      ] UNTIL (KGO.EQ.0)
#
#.....DONE
      CDOPE=CONCP-CONCN
      RETURN
      END
```

File: &LABLN

```
*CALL TWDEFN
      SUBROUTINE LABLN (KGCB,KDCB,LABL,KFONT,CHITE,TXTLEN,NFONT)
@
@     GET LENGTH AND FONT COUNT OF LABEL ARRAY "LABL" OF SPEC "KFONT"
@
      DIMENSION KGCB(1),KDCB(1),LABL(1),KFONT(1),NAMFNT(3),NUMBR(6)
@
      DATA NXFNT/2/,   NYFNT/10/,  NSEC  /0/,  NCR/2HGG/,^
           NAMFNT/2HFO,2HNT,2H  /, NUMBR /2H1 ,2H2 ,2H3 ,2H4 ,2H5 ,2H6 /,^
           ASPEC/0.7(EXP)0/, SLANT/0.0(EXP)0/, SUP/0.75(EXP)0/
@
      NS=1                                                   @ RESET POSITION PNT'R
      KEND=0                                                 @ RESET END INDEX
      NFONT=1                                                @ RESET FONT COUNT
      KFOLD=0                                                @ RESET OLD FONT INDEX
      TXTLEN=0.0(EXP)0                                           @ RESET TEXT LENGTH
      CALL LDIR (KGCB,0.0(EXP)0)
      REPEAT [
          NP=(NFONT-1)*NXFNT+1
          NC=KFONT(NP)                                       @ GET CHAR'S COUNT
          IF (NC.EQ.0) KEND=1                                @ SET END INDEX IF ZERO
          ELSE [
              KF=KFONT(NP+1)                                 @ GET FONT INDEX
              IKF=IABS(KF)
              IF (IKF.NE.KFOLD) [                            @ CHANGE FONT
                  NAMFNT(3)=NUMBR(IKF)
                  CALL GFONT (KGCB,NAMFNT,NSEC,NCR,KDCB)
              ]
              IF (KF.LT.0)                                   @ CHANGE SIZE
                  CALL CSIZE (KGCB,CHITE*SUP,ASPEC,SLANT,0)
              CALL GLEN  (KGCB,LABL,NS,NC,XT,YT,KDCB)        @ GET TEXT LENGTH
              IF (KF.LT.0)                                   @ RESTORE SIZE
                  CALL CSIZE (KGCB,CHITE,ASPEC,SLANT,0)
              TXTLEN=TXTLEN+XT                                @ UPDATE LABEL LENGTH
              NS=NS+NC                                        @ UPDATE POINTER
              NFONT=NFONT+1                                   @ UPDATE FONT INDEX
              KFOLD=IKF                                       @ UPDATE FONT INDEX
          ]
      ] UNTIL ((NFONT.GT.NYFNT).OR.(KEND.EQ.1))
      NFONT=NFONT-1
      IF (NFONT.NE.0) CALL GFONT (KGCB,0,0,0,KDCB)            @ CLOSE FONT FILE
      RETURN
      END
```

File: &LABWR

```
*CALL TWDEFN
      SUBROUTINE LABWR (KGCB,KDCB,LABL,KFONT,CHITE,NFONT,XOFF,YOFF)
@
@     WRITE OUT LABEL ARRAY LABL WITH SPEC IN "KFONT" UP TO "NFONT" FONTS
@
      DIMENSION KGCB(1),KDCB(1),LABL(1),KFONT(1),NAMFNT(3),NUMBR(6)
@
      DATA NXFNT/2/,   NYFNT/10/,  NSEC  /0/,      NCR/2HGG/,^
           NAMFNT/2HFO,2HNT,2H  /, NUMBR/2H1 ,2H2 ,2H3 ,2H4 ,2H5 ,2H6 /,^
           ASPEC/0.7(EXP)0/,   SLANT/0.0(EXP)0/,   SUP/0.75(EXP)0/
@
      XOFO=XOFF/3.0(EXP)0
      YOFO=YOFF/3.0(EXP)0
      NFOLD=0                                          @ RESET OLD FONT INDEX
      NS=1                                             @ RESET POINTER
      DO K=1,NFONT [
          KK=(K-1)*NXFNT+1
          NC=KFONT(KK)                                 @ GET CHARACTER COUNT
          KF=KFONT(KK+1)                               @ GET FONT FLAG
          IKF=IABS(MOD(KF,10))                              @ GET FONT INDEX
          IF (IKF.NE.NFOLD) [                          @ CHECK IF CHANGE FONT
              NAMFNT(3)=NUMBR(IKF)                     @ GET FONT FILE NAME
              CALL GFONT (KGCB,NAMFNT,NSEC,NCR,KDCB) @ GET NEW FONT
          ]
          IF (KF.LT.0) [                               @ CHECK IF CHANGE SIZE
              CALL CSIZE (KGCB,CHITE*SUP,ASPEC,SLANT,0)  @ CHANGE SIZE
              CALL WHERE (KGCB,XP,YP)                  @ GET CURRENT LOC
              IF (IABS(KF).LT.10) CALL MOVE (KGCB,XP+XOFF,YP+YOFF) @ OFFSET
              ELSE                CALL MOVE (KGCB,XP-XOFO,YP-YOFO) @ OFFSET
          ]
          CALL GTEXT (KGCB,LABL,NS,NC,KDCB)            @ OUTPUT TEXT
          IF (KF.LT.0) [                               @ CHECK IF RESTORE SIZE
              CALL CSIZE (KGCB,CHITE,ASPEC,SLANT,0)  @ RESTORE SIZE
              CALL WHERE (KGCB,XP,YP)                  @ GET CURRENT LOC
              IF (IABS(KF).LT.10) CALL MOVE (KGCB,XP-XOFF,YP-YOFF)  @ RESTORE
              ELSE                CALL MOVE (KGCB,XP+XOFO,YP+YOFO)  @ RESTORE
          ]
          NS=NS+NC                                     @ UPDATE POINTER
          NFOLD=IKF                                    @ UPDATE OLD FONT INDEX
      ]
      CALL GFONT (KGCB,0,0,0,KDCB)                     @ CLOSE FONT FILE
      RETURN
      END
```

File: &LOCXY

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      INTEGER FUNCTION LOCX (X)
C
C     GET X-LOCATION INDEX
C
*CALL TEMA
*CALL TCOMMN
C
      LOCX=1
      XP=XPOS(LOCX)
      WHILE ((X.GT.XP).AND.(LOCX.LT.NXMAX)) [
          LOCX=LOCX+1
          XP=XPOS(LOCX)
      ]
      IF ((LOCX.NE.1).AND.(X.LT.XP)) LOCX=LOCX-1
C
C.....DONE
      RETURN
      END
```


File: &LOCXY

```
*IF HP1000
EMA (XYZ,0)
*ENDIF
      INTEGER FUNCTION LOCY (Y)
C
C     GET Y-LOCATION INDEX
C
*CALL TEMA
*CALL TCOMMN
C
      LOCY=1
      YP=YPOS(LOCY)
      WHILE ((Y.GT.YP).AND.(LOCY.LT.NYMAX)) [
          LOCY=LOCY+1
          YP=YPOS(LOCY)
      ]
      IF ((LOCY.NE.1).AND.(Y.LT.YP)) LOCY=LOCY-1
C
C.....DONE
      RETURN
      END
```

File: &LUNIT

```
*CALL TWDEFN
      INTEGER FUNCTION  LUNIT (X0,X1)
#
#     GET THE POWER OF 10 OF (X0,X1) IN ENGINEERING UNIT
#
      DIMENSION PARS(2), KORDR(2), LBUF(2)
#
      PARS(1)=ABS(X0)                 # LOAD INPUT INTO ARRAY
      PARS(2)=ABS(X1)
      DO K=1,2 [                      # LOOP THROUGH INPUTS
          KORDR(K)=0                  # RESET ORDER
          IF (PARS(K).NE.0.0(EXP)0) [
              ORDR=ALOGT(PARS(K))     # TAKE LOGT
              CALL CODE; WRITE (LBUF,1000) ORDR      # KEEP INETEGER PART ONLY
1000          FORMAT (F4.0)
              CALL CODE; READ (LBUF,2000) KORDR(K)   # TRANSFORM BACK
2000          FORMAT (I4)
          ]
          KORDR(K)=3*(KORDR(K)/3)                    # IN UNIT OF THREE
      ]
      LUNIT=MAX0(KORDR(1),KORDR(2))                  # TAKE THE MAXIMUM
      RETURN
      END
```


File: &MESSG

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE MESSG (KANSWR,KFLAG)
#
#     DISPLAY MESSGES ON CONSOLE
#
*CALL TEMA
*CALL TCONMN
#
      DIMENSION MSGRAD(3),MSGOXD(3),MSGPOS(8),MSGNXT(8),MSGSIZ(8),MSGTOP(8)
#
      EQUIVALENCE (NANY,KYES)
      DATA NSTR/2H**/, NANX/2HX /,NANY/2HY /,^
           MSGRAD/2HGr,2Had,2Hed/,MSGOXD/2HOx,2Hid,2He /,^
           MSGPOS/2HPo,2Hsi,2Hti,2Hon,2H a,2Hrr,2Hay,2H  /,^
           MSGSIZ/2HMe,2Hsh,2H s,2Hiz,2He ,2Har,2Hra,2Hy:/,^
           MSGNXT/2HNe,2Hxt,2H t,2Ho ,2Hsi,2Hli,2Hco,2Hn /,^
           MSGTOP/2HOn,2H t,2Hop,2H o,2Hf ,2Hox,2Hid,2He:/
*IF HP2640A
      DATA KESC/033B/
*ENDIF
*IF !BATCH
      DATA KBELL/007B/
*ENDIF
#
#.....GET MESSAGE INDEX
      KMSSG=IABS(KFLAG)
#
#.....PREPARE THE TERMINAL: HOME CURSOR
#                           CLEAR SCREEN
#                           DISPLAY LOGO
#                           DEFINE SOFT KEYS: F7=YES AND F8=NO
#                           LOCK DISPLAY MEMORY
*IF HP2640A
      IF (KMSSG.EQ.1) WRITE (KONSOL,1000) (KESC,K=1,19)
1000  FORMAT (R1,"h",R1,"J",^                         # home cursor, clear display
         11X,R1,"&dD     TWIST      ",R1,"&d@"^       # underline program name
         19X,"Version A5, November 27, 1980"^         # version and date
         /,/,R1,"&dJ",11X,^                           # half-bright inverse vedio
             R1,"&dHTw",R1,"&dJo-dimensional ",^      # underline key characters
             R1,"&dHI", R1,"&dJnteractive ",^
             R1,"&dHS", R1,"&dJimulation of MOS ",^
             R1,"&dHT", R1,"&dJransistors",^
             11X,R1,"&d@"^                            # end of logo
         /,/,R1,"1",^                                 # lock display memory
             R1,"&f7k2a3LYes",^                       # define f7=Yes
             R1,"&f8k2a2LNo"/)                        # define f8=No
*ENDIF
#
#.....CHECK PROFILE
      IF (KMSSG.EQ.2) [
           WRITE (KONSOL,3000) KBELL
3000       FORMAT(R1,"Check impurity profile (Yes=f7/No=f8) ? _")
           READ  (KEYBRD,2000) KANSWR
2000       FORMAT(A1)
*IF BATCH
           WRITE (KONSOL,2000) KANSWR
*ENDIF
      ]
#
#.....CHECK MESH
      IF (KMSSG.EQ.3) [
           WRITE (KONSOL,4000)
4000       FORMAT("Check the mesh (Yes=f7/No=f8) ? _")
           READ  (KEYBRD,2000) KANSWR
*IF BATCH
           WRITE (KONSOL,2000) KANSWR
*ENDIF
           IF (KANSWR.EQ.KYES) [
               ESIEOX=EPSI/EPSIO2
               DO N=1,NXMAX; DO M=1,2; DELOX(N,M)=DELOX(N,M)*ESIEOX
               NXM1=NXMAX-1
               NYM1=NYMAX-1
               WRITE (KONSOL,4010) (MSGDOT,K=1,3),(MSGRAD(K),K=1,3),NANX,^
                                   (MSGPOS(K),K=1,8),(XPOS(K),K=1,NXMAX)
               WRITE (KONSOL,4020) (MSGDOT,K=1,6),(MSGSIZ(K),K=1,8),^
                                   (DELX(K),K=1,NXM1)
               WRITE (KONSOL,4010) (MSGDOT,K=1,3),(MSGRAD(K),K=1,3),NANY,^
```

File: &MESSG

```
                                       (MSGPOS(K),K=1,8),(YPOS(K),K=1,NYMAX)
                WRITE (KONSOL,4020) (MSGDOT,K=1,6),(MSGSIZ(K),K=1,8),^
                                       (DELY(K),K=1,NYM1)
                WRITE (KONSOL,4010) (MSGDOT,K=1,3),(MSGOXD(K),K=1,3),NAMY,^
                                       (MSGNXT(K),K=1,8),(DELOX(K,1),K=1,NXMAX)
                WRITE (KONSOL,4020) (MSGDOT,K=1,6),(MSGTOP(K),K=1,8),^
                                       (DELOX(K,2),K=1,NXMAX)
4010            FORMAT(6A2,1X,A1,"-mesh defined. ",8A2,":"/,(1PE9.2,7(1PE10.2)))
4020            FORMAT(14A2,":"/,(1PE9.2,7(1PE10.2)))
                EOXESI=EPSIO2/EPSI
                DO N=1,NXMAX; DO M=1,2; DELOX(N,M)=DELOX(N,M)*EOXESI
            ]
      ]
*
*.....SAVE INPUT PARAMETERS
      IF (KMSSG.EQ.4) [
            WRITE (KONSOL,5000)
5000        FORMAT("Save input parameters (Yes=f7/No=f8) ? _")
            READ  (KEYBRD,2000) KANSWR
*IF BATCH
            WRITE (KONSOL,2000) KANSWR
*ENDIF
      ]
*
*.....CHECK IF ANOTHER RUN
      IF (KMSSG.EQ.5) [
            WRITE (KONSOL,8000)
8000        FORMAT("Another run (Yes=f7/No=f8) ? _")
            READ  (KEYBRD,2000) KANSWR
*IF BATCH
            WRITE (KONSOL,2000) KANSWR
*ENDIF
      ]
*
*.....UN-LOCK DISPLAY MEMORY
*IF HP2648A
      IF (KMSSG.EQ.6) WRITE (KONSOL,8900) KESC
8900  FORMAT(A1,"m")
*ENDIF
*
*.....CHECK OUTPUT LOOP
      IF (KMSSG.EQ.9) [
            WRITE (KONSOL,9900)
9900        FORMAT(/,"Another output (Yes=f7/No=f8) ? _")
            READ  (KEYBRD,2000) KANSWR
*IF BATCH
            WRITE (KONSOL,2000) KANSWR
*ENDIF
      ]
*
*.....DONE
      RETURN
      END
```


File: &OUTDA

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      FUNCTION OUTDA (KX,KY,KDISP,KSIGN,KFELD)
*
*     LOAD DATA
*
*CALL TEMA
*CALL TCOMMN
*
      IF (KDISP.EQ.1)   OUTDA=CONC(KX,KY)
      IF (KDISP.EQ.4) [ OUTDA=POTSI(KX,KY); IF (KSIGN.EQ.1) OUTDA=-OUTDA ]
      IF (KDISP.EQ.2) [ OUTDA=CARRIE(KX,KY); IF (KSIGN.EQ.1) OUTDA=-OUTDA ]
      IF (KDISP.EQ.3) [
         IF (KFELD.NE.2)
            IF ((KX.NE.1).AND.(KX.NE.NXMAX)) [
               DELTA=XPOS(KX+1)-XPOS(KX-1)
               XFLD=-(POTSI(KX+1,KY)-POTSI(KX-1,KY))/DELTA
            ]
            ELSE [
               IF (KX.EQ.1) XFLD=-(POTSI(2,KY) -POTSI(1,KY))    /DELX(1)
               ELSE         XFLD=-(POTSI(KX,KY)-POTSI(KX-1,KY))/DELX(KX-1)
            ]
         IF (KFELD.NE.1)
            IF ((KY.NE.1).AND.(KY.NE.NYMAX)) [
               DELTA=YPOS(KY+1)-YPOS(KY-1)
               YFLD=-(POTSI(KX,KY+1)-POTSI(KX,KY-1))/DELTA
            ]
            ELSE [
               IF (KY.EQ.1) YFLD=-(POTSI(KX,2) -POTSI(KX,1))    /DELY(1)
               ELSE         YFLD=-(POTSI(KX,KY)-POTSI(KX,KY-1))/DELY(KY-1)
            ]
         IF (KFELD.EQ.1) OUTDA=XFLD
         IF (KFELD.EQ.2) OUTDA=YFLD
         IF (KFELD.EQ.3) OUTDA=ABS(XFLD/YFLD)
         IF (KFELD.EQ.4) OUTDA=SQRT(XFLD*XFLD+YFLD*YFLD)
      ]
      RETURN
      END
```

```
*CALL TUDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE OUTGP (KGRPH,KDISP,KSIGN,KFELD)
#
#     GET GRAPHICS OUTPUT INFO'S
#
#       KDISP = OUTPUT                FLAG
#               0 -> SAVE INPUT
#               1 -> DOPING PROFILE
#               2 -> CARRIER
#               3 -> FIELD
#               4 -> POTENTIAL
#
#       KGRPH = GRAPHISC OUTPUT FLAG
#               0 -> END OF OUTPUT
#               2 -> 2-D                   : Z(N,1)=X, Z(N,2)=Y
#              -2 -> 2-D, CARRIER DENSITY
#               3 -> 3-D GRAPHICS DISPLAY: Z() = DATA
#               4 -> CONTOUR PLOT OF POTENTIAL
#
#       KL      = INDEX OF LAST WORD
#       KF      = INDEX OF LAST FONT
#       KFONTX(1,K) = # OF CHARACTERS IN FONT KFONTX(2,K)
#       KFONTY(1,K) = # OF CHARACTERS IN FONT KFONTY(2,K)
#       KFONTZ(1,K) = # OF CHARACTERS IN FONT KFONTZ(2,K)
#
#       2-D PLOT:
#               (NX1,NX2) INDEX OF POSITION ARRAY Z(N,1)
#               (NY1,   ) SET TO NAME OF CROSS SECTION, I.E. "X" OR "Y"
#               (   ,NY2) SET TO THE COMPLEMENT OF CROSS SECTION NAME
#               (X1, X2)  POSITION ARRAY LIMITS
#               (Y1, Y2)  CROSS SECTION POSITION (SET TO EQUAL)
#               (Z1, Z2)  FUNTION LIMITS
#
#       3-D PLOT:
#               (NX1,NX2) INDEX OF X LIMITS
#               (NY1,NY2) INDEX OF Y LIMITS
#               (X1, X2)  X LIMITS
#               (Y1, Y2)  Y LIMITS
#               (Z1, Z2)  Z LIMITS
#
*CALL TEMA
*CALL TCONNO
#
      DIMENSION LBUF(40)
      DATA NAMX/2HX /, NAMY/2HY /, NAMBLK/2H  /, KYES/2HY /
      DATA KSTICK/2/,  KROMAN/3/,  KMATH/5/,     NFONT/10/,    LUPLTR/9/
      DATA NUM/4/
#
      WRITE (KONSOL,6010)                                # SET LOG FLAG
      READ  (KEYBRD,2000) KANSUR
*IF BATCH
      WRITE (KONSOL,2000) KANSUR
*ENDIF
6010  FORMAT("Log scale? (Yes=f7/No=f8) _")
      IF (KANSUR.EQ.KYES) [
          KLOG=1
          IF (Z1.NE.0.0(EXP)0) Z1=ALOGT(ABS(Z1))            # LOG OF Z LIMITS
          IF (Z2.NE.0.0(EXP)0) Z2=ALOGT(ABS(Z2))
          IF (Z1.GE.Z2) [ T=Z1; Z1=Z2; Z2=T ]           # MAKE Z LIMITS VALID
      ]
      ELSE KLOG=0
      IF (IABS(KGRPH).EQ.2) [
          WRITE (KONSOL,6015)
          READ  (KEYBRD,2000) KANSUR
*IF BATCH
          WRITE (KONSOL,6015) KANSUR
*ENDIF
6015      FORMAT("Relative position (Yes=f7/No=f8) ? _")
          IF (KANSUR.EQ.KYES) [
              KRELC=1
              DOMAIN=1.0(EXP)0/(Z(NX2,1)-Z(NX1,1))
              NSTRT=NX1+1
              NSTOP=NX2-1
              DO N=NSTRT,NSTOP; Z(N,1)=(Z(N,1)-Z(NX1,1))*DOMAIN
              Z(NX1,1)=0.0(EXP)0
              Z(NX2,1)=1.0(EXP)0
              X1=0.0(EXP)0
```



```
              X2=1.0(EXP)0
          ]
          ELSE KRELC=0
      ]
      ELSE KRELC=0
      WRITE (KONSOL,6000)                                # GET DEVICE LU
      READ  (KEYBRD,2000) KANSUR
*IF BATCH
      WRITE (KONSOL,2000) KANSUR
*ENDIF
2000  FORMAT(A1)
6000  FORMAT("On the console (Yes=f7/No=f8) ? _")
      IF (KANSUR.EQ.KYES) LU=KONSOL
      ELSE                LU=LUPLTR
      IF (KGRPH.EQ.3) [                                  # CASE OF 3-D PLOT
          WRITE (KONSOL,6020) NAMX                       # NUMBER OF POINTS
          READ  (KEYBRD,*)    NX
*IF BATCH
          WRITE (KONSOL,6021) NX
6021      FORMAT(I3)
*ENDIF
          WRITE (KONSOL,6020) NAMY
          READ  (KEYBRD,*)    NY
*IF BATCH
          WRITE (KONSOL,6021) NY
*ENDIF
6020      FORMAT("How many points in ",A1,"-direction ? _")
          WRITE (KONSOL,6030)                            # TILTING ANGLE
          READ  (KEYBRD,*) TILT
*IF BATCH
          WRITE (KONSOL,6031) TILT
6031      FORMAT(1PG10.3)
*ENDIF
6030      FORMAT("Tilt angle (degree) ? _")
          WRITE (KONSOL,6040)                            # ROTATING ANGLE
          READ  (KEYBRD,*) ROTAT
*IF BATCH
          WRITE (KONSOL,6031) ROTAT
*ENDIF
6040      FORMAT("Rotation angle (degree) ? _")
          IF (KLOG.EQ.1) DO KX=NX1,NX2; DO KY=NY1,NY2 [  # LOG OF DATA
              ZZ=ABS((Z(KX,KY)))
              IF (ZZ.NE.0.0(EXP)0) Z(KX,KY)=ALOGT(ZZ)
          ]
      ]
      ELSE IF (KLOG.EQ.1) DO K=NX1,NX2 [                 # CASE OF 2-D PLOT
          ZZ=ABS((Z(K,2)))
          IF (ZZ.NE.0.0(EXP)0) Z(K,2)=ALOGT(ZZ)
      ]
#
#     PREPARE LABLES
#
      IF (KLOG.EQ.0) [                                   # IF NOT LOG SCALE
          LUNTZ=LUNIT(Z1,Z2)                             # GET POWER OF TEN'S
          IF (LUNTZ.NE.1) [
              ZUNIT=10.0(EXP)0**(-LUNTZ)                   # GET SCALER
              Z1=Z1*ZUNIT                                # SCALE Z
              Z2=Z2*ZUNIT
              IF (KGRPH.NE.3) DO K=NX1,NX2; Z(K,2)=Z(K,2)*ZUNIT            # 2-D
              ELSE DO KX=NX1,NX2; DO KY=NY1,NY2; Z(KX,KY)=Z(KX,KY)*ZUNIT # 3-D
              ITEM=0                                     # GET DIGIT COUNT
              IT=LUNTZ
              IF (IT.LT.0) ITEM=ITEM+1
              WHILE (IT.NE.0) [ ITEM=ITEM+1; IT=IT/10 ]
          ]
          ELSE ITEM=0
      ]
      ELSE ITEM=0                                        # RESET SCALER
      KL=0
      KF=1                                               # INITIALIZE FONT
      KFONTZ(1,1)=0
      KFONTZ(2,1)=KROMAN
      IF (KDISP.EQ.1) [                                  # LABEL OF DOPING CONC.
          CALL CODE; WRITE (LBUF,7010)
7010      FORMAT ("Doping Conc. (")
          LAB=7
      ]
      IF (KDISP.EQ.2) [
          IF ((KSIGN.EQ.1).AND.(KGRPH.LT.0)) [           # LABEL OF ELEC. DENS.
              CALL CODE; WRITE (LBUF,7021)
```

```
7021        FORMAT ("Electron Density (")
            LAB=9
         ]
         IF ((KSIGN.EQ.2).AND.(KGRPH.LT.0)) [                # LABEL OF HOLE DENS.
            CALL CODE; WRITE (LBUF,7022)
7022        FORMAT ("Hole Density (")
            LAB=7
         ]
         IF ((KSIGN.EQ.1).AND.(KGRPH.GT.0)) [                # LABEL OF ELEC. CONC.
            CALL CODE; WRITE (LBUF,7023)
7023        FORMAT ("Electron Conc. (")
            LAB=8
         ]
         IF ((KSIGN.EQ.2).AND.(KGRPH.GT.0)) [                # LABEL OF HOLE CONC.
            CALL CODE; WRITE (LBUF,7024)
7024        FORMAT ("Hole Conc. (")
            LAB=6
         ]
      ]
      IF (KDISP.EQ.3) [
         IF (KFELD.EQ.1) [                                   # LABEL OF X-FIELD
            CALL CODE; WRITE (LBUF,7031)
7031        FORMAT ("X-Field  (")
            LAB=5
         ]
         IF (KFELD.EQ.2) [                                   # LABEL OF Y-FIELD
            CALL CODE; WRITE (LBUF,7032)
7032        FORMAT ("Y-Filed  (")
            LAB=5
         ]
         IF (KFELD.EQ.3) [                                   # LABEL OF X/Y RATIO
            CALL CODE; WRITE (LBUF,7033)
7033        FORMAT ("X/Y  Field Ratio")
            LAB=8
         ]
         IF (KFELD.EQ.4) [                                   # LABEL OF FIELD MGN.
            CALL CODE; WRITE (LBUF,7034)
7034        FORMAT ("Field Mgn. (")
            LAB=6
         ]
      ]
      IF (KDISP.EQ.4) [
         IF (KSIGN.EQ.1) [                                   # LABEL OF ELEC. POT'L
            CALL CODE; WRITE (LBUF,7041)
7041        FORMAT ("Elec. Potential  (")
            LAB=9
         ]
         ELSE [                                              # LABEL OF POTENTIAL
            CALL CODE; WRITE (LBUF,7042)
7042        FORMAT ("Potential  (")
            LAB=6
         ]
      ]
      DO K=1,LAB; LABLZ(K+KL)=LBUF(K)                        # LOAD Z LABEL
      KL=KL+LAB                                              # UPDATE WORD COUNT
      KFONTZ(1,KF)=KFONTZ(1,KF)+LAB*2                        # UPDATE FONT
      IF ((KDISP.EQ.3).AND.(KFELD.EQ.3)) [                   # CASE OF FIELD RATIO
         IF (ITEM.NE.0) [                                    # ADD SCALING UNIT
            IF (ITEM.EQ.1) [                                 # OF ONE DIGIT
               CALL CODE; WRITE (LBUF,8051) LUNTZ
8051           FORMAT (" (x10_",I1," _)")
               LAB=5
            ]
            IF (ITEM.EQ.2) [                                 # OF TWO DIGITS
               CALL CODE; WRITE (LBUF,8052) LUNTZ
8052           FORMAT (" (x10_",I2, "_)")
               LAB=5
            ]
            IF (ITEM.EQ.3) [                                 # OF THREE DIGITS
               CALL CODE; WRITE (LBUF,8053) LUNTZ
8053           FORMAT (" (x10_",I3," _)")
               LAB=6
            ]
            KFONTZ(1,KF)=KFONTZ(1,KF)+6                      # UPDATE FONT
            KFONTZ(1,KF+1)=LAB*2-6-1                         # CHANGE SIZE
            KFONTZ(2,KF+1)=-KROMAN
            KFONTZ(1,KF+2)=1                                 # RESTORE SIZE
            KFONTZ(2,KF+2)=KROMAN
            KF=KF+2                                          # UPDATE FONT INDEX
         ]
```



```
      ]
      ELSE [
         IF (ITEM.NE.0) [                                    # ADD LABEL OF SCALE
            IF (ITEM.EQ.1) [                                 # OF ONE DIGIT
               CALL CODE; WRITE (LBUF,8001) LUNTZ
8001           FORMAT ("x10_",I1,"_")
               LAB=3
            ]
            IF (ITEM.EQ.2) [                                 # OF TWO DIGITS
               CALL CODE; WRITE (LBUF,8002) LUNTZ
8002           FORMAT ("x10_",I2," _")
               LAB=4
            ]
            IF (ITEM.GT.2) [                                 # AT MOST, THREE DIGITS
               CALL CODE; WRITE (LBUF,8003) LUNTZ
8003           FORMAT ("x10_",I3,"_")
               LAB=4
            ]
            KFONTZ(1,KF)=KFONTZ(1,KF)+4                      # UPDATE FONT
            KFONTZ(1,KF+1)=LAB*2-4                           # CHANGE SIZE
            KFONTZ(2,KF+1)=-KROMAN
            KFONTZ(1,KF+2)=0                                 # RESTORE SIZE
            KFONTZ(2,KF+2)=KROMAN
            KF=KF+2                                          # UPDATE FONT INDEX
            DO K=1,LAB; LABLZ(K+KL)=LBUF(K)                  # LOAD THE LABEL
            KL=KL+LAB                                        # UPDATE WORD INDEX
         ]
         IF ((KDISP.EQ.1).OR.(KDISP EQ.2)) [                 # ADD LABEL OF UNIT
            IF (KGRPH.GT.0) [ CALL CODE; WRITE (LBUF,8010) ]
            ELSE            [ CALL CODE; WRITE (LBUF,8020) ]
8010        FORMAT (" Cm_-3_)")
8020        FORMAT (" Cm_-2_)")
            LAB=4
            KFONTZ(1,KF)=KFONTZ(1,KF)+4                      # UPDATE FONT
            KFONTZ(1,KF+1)=LAB*2-4-1                         # CHANGE SIZE
            KFONTZ(2,KF+1)=-KROMAN
            KFONTZ(1,KF+2)=1                                 # RESTORE SIZE
            KFONTZ(2,KF+2)=KROMAN
            KF=KF+2                                          # UPDATE FONT INDEX
         ]
         IF (KDISP.EQ.3) [
            CALL CODE; WRITE (LBUF,8030)
8030        FORMAT (" Volts/Cm)")
            LAB=5
            KFONTZ(1,KF)=KFONTZ(1,KF)+2*LAB                  # UPDATE FONT
         ]
         IF (KDISP.EQ.4) [
            CALL CODE; WRITE (LBUF,8040)
8040        FORMAT (" Volts) ")
            LAB=4
            KFONTZ(1,KF)=KFONTZ(1,KF)+LAB*2                  # UPDATE FONT
         ]
      ]
      DO K=1,LAB; LABLZ(K+KL)=LBUF(K)                        # LOAD Z LABEL
      KL=KL+LAB                                              # UPDATE WORD INDEX
      IF (KF.LT.NFONT) [ KFONTZ(1,KF+1)=0; KFONTZ(2,KF+1)=0 ]
#
# END OF Z-LABEL AND BEGINNING OF X-LABEL
#
      IF (KRELC.EQ.1) [
         CALL CODE; WRITE (LABLX,9005)
9005     FORMAT (" Normalized ")
         KL=6
      ]
      ELSE KL=0
      KFONTX(1,1)=2*KL
      KFONTX(2,1)=KROMAN
      IF (KGRPH.NE.3) [ CALL CODE;  WRITE (LBUF,9010)  NY2;  LAB=4 ]  # X-LABEL
      ELSE            [ CALL CODE;  WRITE (LBUF,9010) NANX; LAB=4 ]
9010  FORMAT (A1," (_m_m)")
      DO K=1,LAB; LABLX(K+KL)=LBUF(K)                       # SET WORD INDEX
      IF (KRELC.EQ.0) [
         KFONTX(1,1)=KFONTX(1,1)+4                          # SET FONT
         KFONTX(1,2)=2                                      # CHANGE FONT
         KFONTX(2,2)=KMATH
         KFONTX(1,3)=2                                      # RESTORE FONT
         KFONTX(2,3)=KROMAN
         KFONTX(1,4)=0                                      # END OF X-LABEL
      ]
      ELSE [
```

```
          KFONTX(1,1)=KFONTX(1,1)+1
          KFONTX(1,2)=0
      ]
      IF (KGRPH.NE.3) [                                    # Y-LABEL
          IF (KGRPH.LT.0) KFONTY(1,1)=0
          ELSE [
              CALL CODE; WRITE (LABLY,9015) NY1,Y1
9015          FORMAT (A1," = ",F6.2,"_n_n")
              KL=7
              KFONTY(1,1)=11
              KFONTY(2,1)=KSTICK
              KFONTY(1,2)=2
              KFONTY(2,2)=KMATH
              KFONTY(1,3)=1
              KFONTY(2,3)=KSTICK
              KFONTY(1,4)=0
          ]
      ]
      ELSE [
          CALL CODE; WRITE (LABLY,9010) NANY
          KL=9
          KFONTY(1,1)=14                                    # SET FONT
          KFONTY(2,1)=KROMAN
          KFONTY(1,2)=2                                     # CHANGE FONT
          KFONTY(2,2)=KMATH
          KFONTY(1,3)=2                                     # RESTORE FONT
          KFONTY(2,3)=KROMAN
          KFONTY(1,4)=0                                     # END OF Y-LABEL
      ]
#
#.....DONE
      RETURN
      END
```



```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE OUTP1 (KOUNT,KFLAG)
#
#    PREPARE DISPLAY DATA AND PROCESS NON-GRAPHICS OUTPUT
#
#         KOUNT = OUTPUT LOOP KOUNT
#         KFLAG = OUTPUT REQUESTING FLAG
#               =  - -> QUESTION AND ANSWER IN ROOT
#                  0 -> SAVE INPUT
#                  1 -> OUTPUT AFTER DOPING    PROFILE GENERATED
#                  4 ->                        POTENTIAL
#
#         KDISP = DISPLAY FLAG
#                 0 -> SAVE INPUT
#                 1 -> DOPING PROFILE
#                 2 -> CARRIER
#                 3 -> FIELD
#                 4 -> POTENTIAL
#
#         KTRGT = DISPLAY TARGET FLAG
#                 1 -> 2-D PLOT
#                 2 -> 3-D PLOT
#                 3 -> DISC FILE
#                 4 -> PRINT
#
#         KGRPH = GRAPHISC OUTPUT FLAG
#                 0 -> END OF OUTPUT
#                 2 -> 2-D                  ; Z(N,1)=X, Z(N,2)=Y
#                 3 -> 3-D GRAPHICS DISPLAY; Z(NX,NY)=DATA
#                 4 -> CONTOUR PLOT OF POTENTIAL
#
#         2-D PLOT:
#                 (NX1,NX2) INDEX OF POSITION ARRAY Z(N,1)
#                 (NY1,    ) SET TO NAME OF CROSS SECTION, I.E. "X" OR "Y"
#                 (    ,NY2) SET TO THE COMPLEMENT OF CROSS SECTION NAME
#                 (X1, X2)  POSITION ARRAY LIMITS
#                 (Y1, Y2)  CROSS SECTION POSITION (SET TO EQUAL)
#                 (Z1, Z2)  FUNTION LIMITS
#
#         3-D PLOT:
#                 (NX1,NX2) INDEX OF X LIMITS
#                 (NY1,NY2) INDEX OF Y LIMITS
#                 (X1, X2)  X LIMITS
#                 (Y1, Y2)  Y LIMITS
#                 (Z1, Z2)  Z LIMITS
#
*CALL TEMA
*CALL TCONNO
#
      DIMENSION XPRNT(7), YPRNT(8), LBUF(40)
      DIMENSION NAMFLD(4),NAMDSP(4),NAMTGT(4)
#
      EQUIVALENCE (KYES,NANY)
#
      DATA KYES/2HY /, KSTAR/2H**/,^
           NAMDSP/2HD ,2HC ,2HF ,2HP /,^
           NAMTGT/2H2 ,2H3 ,2HS ,2HP /,^
           NAMFLD/2HX ,2HY ,2HR ,2HN /
      DATA NDSP/4/,     NTGT/4/,   NFLD/4/
      DATA UM/1.0(EXP)4/, NAMX/2HX /
#
#.....QUESTION/ANSWER OR SAVE INPUT
      IF      (KFLAG.LT.0) CALL MESSG (KOUNT,KFLAG)
      ELSE IF (KFLAG.EQ.0) CALL SAVIN
      ELSE [
#
#.....RESET INDICES
          KDISP=KFLAG
          KGRPH=0
#
#.....SAVE COMMON BLOCK
          IF (KOUNT.EQ.1) [
              WRITE (KONSOL,9000)
9000          FORMAT("Save common blocks (Yes=f7/No=f6) ? _")
              READ  (KEYBRD,2000) KANSWR
              IF (KANSWR.EQ.KYES) CALL WCOMM
          ]
```

```
#.....CHECK WHICH OUTPUT
          IF ((KOUNT.NE.1).AND.(KFLAG.NE.1)) [
*IF HP2640A
               CALL SKEYD
*ENDIF
               WRITE (KONSOL,1010)
1010           FORMAT("Which one ",^
                    "(Doping concentration/f1), (Carrier distribution/f2), "^
             /,10X,"(Field distribution  /f3), (Potential  profile  /f4). ? _")
               READ  (KEYBRD,2000) KANSWR
2000           FORMAT(A1)
*IF BATCH
               WRITE (KONSOL,2000) KANSWR
*ENDIF
               KDISP=1
               WHILE ((KANSWR.NE.NAMDSP(KDISP)).AND.(KDISP.LT.NDSP))
                    KDISP=KDISP+1
          ]
#
#.....CHECK WHICH FIELD COMPONENT
          IF (KDISP.EQ.3) [
*IF HP2640A
               CALL SKEYF
*ENDIF
               WRITE (KONSOL,3020)
3020           FORMAT("X-component(f1), Y-component(f2), ",^
                      "X/Y-ratio(f3) or magnitude(f4) ? _")
               READ  (KEYBRD,2000) KANSWR
*IF BATCH
               WRITE (KONSOL,2000) KANSWR
*ENDIF
               KFELD=1
               WHILE ((KANSWR.NE.NAMFLD(KFELD)).AND.(KFELD.LT.NFLD))
                    KFELD=KFELD+1
          ]
#
#.....CHECK WHICH DISPLAY
*IF HP2640A
          CALL SKEYT
*ENDIF
          WRITE (KONSOL,1015)
1015      FORMAT ("Which one ",^
               "(2-dimensional plots/f1),  (3-dimensional plots/f2),"^
        /,10X,"(Save in FMGR file   /f3),  (Print the numbers   /f4). ? _")
          READ (KEYBRD,2000) KANSWR
*IF BATCH
          WRITE (KONSOL,2000) KANSWR
*ENDIF
          KTRGT=NTGT
          WHILE ((KANSWR.NE.NAMTGT(KTRGT)).AND.(KTRGT.GT.1)) KTRGT=KTRGT-1
#
#.....2-D GRAPHICS DISPLAY
          IF (KTRGT.EQ.1) [
               IF (KDISP.EQ.2) [                         # CARRIER PER UNIT AREA
                    WRITE (KONSOL,2010)
2010                FORMAT("Carrier per unit area (Yes=f7/No=f8) ? _")
                    READ  (KEYBRD,2000) KANSWR
*IF BATCH
                    WRITE (KONSOL,2000) KANSWR
*ENDIF
                    IF (KANSWR.EQ.KYES) [
                         CALL CARRI
                         WRITE (KONSOL,3020)             # GET Z LIMITS
                         READ  (KEYBRD,*) Z1,Z2
                         IF (Z1.LT.Z2) [                 # SET OUTPUT PARAMETERS
                              KGRPH=-2
                              KSIGN=1
                              NX1=NSOURC;   NX2=NDRAIN
                              NY1=0;        NY2=NAMX
                              X1=Z(NX1,1);  X2=Z(NX2,1)
                              Y1=0.0(EXP)0;     Y2=0.0(EXP)0
                         ]
                         ELSE KGRPH=99
                    ]
               ]
               IF (KDISP.EQ.4) [                         # EQUAL POTENTIAL CONTOUR
                    WRITE (KONSOL,2030)
2030                FORMAT("Contour plot of equal potential lines ",^
                          "(Yes=f7/No=f8) ? _")
                    READ  (KEYBRD,2000) KANSWR
```



```
*IF BATCH
                    WRITE (KONSOL,2000) KANSWR
*ENDIF
                    IF (KANSWR EQ.KYES) KGRPH=4
               ]
          ]
#
#.....NOT 2-D GRAPHICS
          IF (KGRPH.EQ.0) [
               IF ((KDISP.EQ.2).OR.(KDISP.EQ.4)) [   # SET SIGN-FLIP FLAG
                    WRITE (KONSOL,3010)
3010                FORMAT("Referring to electron (Yes=f7/No=f8) ? _")
                    READ  (KEYBRD,2000) KANSWR
*IF BATCH
                    WRITE (KONSOL,2000) KANSWR
*ENDIF
                    IF (KANSWR.EQ.KYES) KSIGN=1
                    ELSE                KSIGN=2
               ]
               ELSE KSIGN=1
               DO K=1,NXMAX; Z(K,3)=XPOS(K)*UM        # SCALE X BY UM
               DO K=1,NYMAX; Z(K,4)=YPOS(K)*UM        # SCALE Y BY UM
#
#.....CASE OF 2-D GRAPHICS
          IF (KTRGT.EQ.1) [
               WRITE (KONSOL,4010)                     # GET DIRECTION INDEX
4010           FORMAT("Constant X or Y ? _")
               READ  (KEYBRD,2000) KROSS
*IF BATCH
               WRITE (KONSOL,2000) KROSS
*ENDIF
               IF (KROSS.EQ.NAMY) [                    # CONSTANT Y
*IF IBATCH
                    WRITE (KONSOL,4020) (NAMY,J=1,2),(K,Z(K,4),K=1,NYMAX)
4020                FORMAT(A1,"-mesh-> N,",A1,":",5(I2,",",F7.3,2X)^
                                      /,(1X,6(I2,",",F7.3,2X)))
*ENDIF
                    WRITE (KONSOL,4030)                # GET Y-CROSS SECTION INDEX
4030                FORMAT("Cross section index ? _")
                    READ  (KEYBRD,*) KY
                    KY=MIN0(NYMAX,MAX0(1,KY))
*IF BATCH
                    WRITE (KONSOL,4031) KY
4031                FORMAT(I4)
*ENDIF
*IF IBATCH
                    WRITE (KONSOL,4020) (NAMX,J=1,2),(K,Z(K,3),K=1,NXMAX)
*ENDIF
                    WRITE (KONSOL,4040)                # GET X DOMAIN INDICES
4040                FORMAT("From <> to <> (indices) ? _")
                    READ  (KEYBRD,*) NX1,NX2
                    NX1=MIN0(NXMAX,MAX0(1,NX1))
                    NX2=MIN0(NXMAX,MAX0(1,NX2))
                    IF (NX1.GT.NX2) [ NT=NX1; NX1=NX2; NX2=NT ]
*IF BATCH
                    WRITE (KONSOL,4041) NX1,NX2
4041                FORMAT(2I5)
*ENDIF
                    DO K=NX1,NX2 [                     # LOAD DATA
                         Z(K,1)=Z(K,3)
                         Z(K,2)=OUTDA (K,KY,KDISP,KSIGN,KFELD)
                    ]
                    NY1=NAMY;          NY2=NAMX    # SET SPECIFICATIONS
                    X1=Z(NX1,1);       X2=Z(NX2,1)
                    Y1=Z(KY,4);        Y2=Y1
               ]
               ELSE [                                  # CONSTNAT X
*IF IBATCH
                    WRITE (KONSOL,4020) (NAMX,J=1,2),(K,Z(K,3),K=1,NXMAX)
*ENDIF
                    WRITE (KONSOL,4030)                # GET X-CROSS SECTION INDEX
                    READ (KEYBRD,*) KX
                    KX=MIN0(NXMAX,MAX0(1,KX))
*IF BATCH
                    WRITE (KONSOL,2000) KX
*ENDIF
*IF IBATCH
                    WRITE (KONSOL,4020) (NAMY,J=1,2),(K,Z(K,4),K=1,NYMAX)
*ENDIF
                    WRITE (KONSOL,4040)                # GET Y DOMAIN INDICES
```

```
                        READ  (KEYBRD,*) NY1,NY2
                        NY1=MIN0(NYMAX,MAX0(1,NY1))
                        NY2=MIN0(NYMAX,MAX0(1,NY2))
                        IF (NY1.GT.NY2) [ NT=NY1; NY1=NY2; NY2=NT ]
*IF BATCH
                        WRITE (KONSOL,4041) NY1,NY2
*ENDIF
                        DO K=NY1,NY2 [                  # LOAD DATA
                           Z(K,1)=Z(K,4)
                           Z(K,2)=OUTDA (KX,K,KDISP,KSIGN,KFELD)
                        ]
                        NX1=NY1;           NX2=NY2      # SET SPECIFICATIONS
                        NY1=NAMX;          NY2=NAMY
                        X1=Z(NX1,1);       X2=Z(NX2,1)
                        Y1=Z(KX,3);        Y2=Y1
                    ]
                    WRITE (KONSOL,4050) (K,Z(K,2),K=NX1,NX2)
4050                FORMAT ("** N.Value **",6X,4(1X,I2,":",1PG10.3)^
                                  /,(4X,5(1X,I2,":",  G10.3)))
                    WRITE (KONSOL,5020)
                    READ (KEYBRD,*) Z1,Z2               # GET THE DATA RANGE
*IF BATCH
                    WRITE (KONSOL,5021) Z1,Z2
5021                FORMAT(1P2G15.3)
*ENDIF
                    IF (Z1.LT.Z2) KGRPH=2               # SET GRAPHICS INDEX
                 ]
#
#.....CASE OF NOT 2-D GRAPHICS
                 ELSE [
*IF IBATCH
                    WRITE (KONSOL,4020) (NAMX,J=1,2),(K,Z(K,3),K=1,NXMAX)
*ENDIF
                    WRITE (KONSOL,4040)                 # GET X-DOMAIN INDICES
                    READ  (KEYBRD,*) NX1,NX2
                    NX1=MIN0(NXMAX,MAX0(1,NX1))
                    NX2=MIN0(NXMAX,MAX0(1,NX2))
                    IF (NX1.GT.NX2) [  NT=NX1;   NX1=NX2; NX2=NT ]
*IF BATCH
                    WRITE (KONSOL,4041) NX1,NX2
*ENDIF
*IF IBATCH
                    WRITE (KONSOL,4020) (NAMY,J=1,2),(K,Z(K,4),K=1,NYMAX)
*ENDIF
                    WRITE (KONSOL,4040)                 # GET Y-DOMAIN INDICES
                    READ  (KEYBRD,*) NY1,NY2
                    NY1=MIN0(NYMAX,MAX0(1,NY1))
                    NY2=MIN0(NYMAX,MAX0(1,NY2))
                    IF (NY1.GT.NY2) [  NT=NY1;   NY1=NY2; NY2=NT ]
*IF BATCH
                    WRITE (KONSOL,4041) NX1,NX2
*ENDIF
*IF IBATCH
                    NXEND=MIN0(NX1+6,NXMAX)             # DEFINE CONSOLE DISP DOMAIN
                    NYEND=MIN0(NY1+7,NYMAX)
                    DO K=NX1,NXEND; XPRNT(K-NX1+1)=Z(K,3)
                    DO K=NY1,NYEND; YPRNT(K-NY1+1)=Z(K,4)
                    DO KX=NX1,NX2; DO KY=NY1,NY2        # LOOP OF DATA LOADING
                       Z(KX,KY)=OUTDA (KX,KY,KDISP,KSIGN,KFELD)
                    WRITE (KONSOL,5010) (K,K=NX1,NXEND),(XPRNT(K),K=1,7),^
                                  (KSTAR,K=1,40),(J,YPRNT(J-NY1+1),^
                                   KSTAR,(Z(K,J),K=NX1,NXEND),J=NY1,NYEND)
5010                FORMAT (5X,7I10/," YX(um) *",1PG9.2,6(1PG10.2)/,39A2,A1/,^
                               (I2,0PF7.2,A1,1PG9.2,6(1PG10.2)))
*ENDIF
                    WRITE (KONSOL,5020)                 # GET DATA RANGE
5020                FORMAT (/,"Estimated MIN/MAX function values ",^
                             "(MIN=)MAX->skip) ? _")
                    READ (KEYBRD,*) Z1,Z2
*IF BATCH
                    WRITE (KONSOL,5021) Z1,Z2
*ENDIF
                    IF (Z1.LT.Z2) [
                       X1=XPOS(NX1);      X2=XPOS(NX2)  # SET SPECIFICATIONS
                       Y1=YPOS(NY1);      Y2=YPOS(NY2)
                       IF (KTRGT.EQ.2) KGRPH=3          # SET 3-D PLOT FLAG
                       IF (KTRGT.EQ.3)                  # SAVE IN DISC FILE
                          CALL SAVOU (NX1,NX2,NY1,NY2,X1,X2,Y1,Y2,Z1,Z2,^
                                KDISP,KSIGN,KFELD)
                       IF (KTRGT.EQ.4)                  # PRINT
```



```
                          CALL TPRNT (NX1,NX2,NY1,NY2,X1,X2,Y1,Y2,Z1,Z2,^
                                KDISP,KSIGN,KFELD)
                    ]
                 ]
              ]
*IF GRAPHICS
              IF ((KGRPH.NE.0).AND.(KGRPH.NE.4).AND.(KGRPH.NE.99))
                 CALL OUTGP (KGRPH,KDISP,KSIGN,KFELD)
*ENDIF
              IF (KGRPH.EQ.99) KFLAG=0
              ELSE             KFLAG=IABS(KGRPH)
        ]
#
#.....DONE
        RETURN
        END
```

File: &OUTP2

```
*CALL TWDEFN
      SUBROUTINE OUTP2 (KFLAG)
*
*     GRAPHICS DATA DISPLAY
*       KFLAG = 0 -> PLOT MESH
*               2 -> 2-D PLOT OF Z(N,1) VERSUS Z(N,2)
*               3 -> OPAQUE 3-D PLOT OF ARRAY Z()
*               4 -> EQUAL POTENTIAL CONTOUR PLOT
*
*CALL TCOMMO
*
      IF (KFLAG.EQ.0) CALL PLOTM                          ! PLOT MESH
      ELSE CALL PLOT2 (X1,X2,Z1,Z2,NX1,NX2,LABLX,LABLY,LABLZ,^
                 KFONTX,KFONTY,KFONTZ,KLOG,LU,KONSOL)
      RETURN
      END
```


File: &OUTP3

```
*CALL TWDEFN
      SUBROUTINE OUTP3 (KFLAG)
*
*
*CALL TCOMMO
*
*
      CALL PLOT3 (X1,X2,Y1,Y2,Z1,Z2,NX,NY,ROTAT,TILT,LU,KONSOL)
      RETURN
      END
```

File: &OUTP4

```
*CALL TUDEFN
      SUBROUTINE OUTP4 (KFLAG)
#
#      GRAPHICS DATA DISPLAY
#         KFLAG = 0 -> PLOT MESH
#                 2 -> 2-D PLOT OF Z(N,1) VERSUS Z(N,2)
#                 3 -> OPAQUE 3-D PLOT OF ARRAY Z()
#                 4 -> EQUAL POTENTIAL CONTOUR PLOT
#
*CALL TCOMMO
#
      DATA KYES/2HY /, LUPLTR/9/
#
      WRITE (KONSOL,3000)
      READ  (KONSOL,2000) KANSWR
3000  FORMAT("On the console (Yes=f7/No=f8) _")
2000  FORMAT(A1)
      IF (KANSWR.EQ.KYES)  LU=KONSOL
      ELSE                 LU=LUPLTR
      WRITE (KONSOL,5020)
      READ  (KONSOL,*) VMIN,VMAX,NUMLIN
5020  FORMAT("Potential min, max and number of lines? _")
      CALL PLOTC (VMIN,VMAX,NUMLIN,LU)
      RETURN
      END
```


File: &OUTP5

```
*CALL TUDEFN
      SUBROUTINE OUTP5 (KFLAG)
#
#      OUTPUT CONTROLLER
#
#        KFLAG = -1 -> PLOT MESH
#                 0 -> SAVE INPUT
#                 1 -> OUTPUT AFTER DOPING    PROFILE GENERATED
#                 4 ->                 POTENTIAL
#
#        KOUNT = COUNT OF OUTPUT LOOP
#
#        KGO   = STATUS FLAG
#              =  0 -> END OF OUTPUT
#              =  1 -> CHECK WHICH OUTPUT AND GET OUTPUT DATA
#
*CALL TCOMMO
#
      DATA KYES/2HY /, KMSG/9/
#
*IF GRAPHICS&HP1000
      IF (KFLAG.EQ.-1) CALL LINK (1HE,0)       #.....PLOT MESH
*ENDIF
*IF GRAPHICS&!HP1000
      IF (KFLAG.EQ.-1) CALL OUTP1 (0)
*ENDIF
*IF HP1000&!BATCH
      IF (KFLAG.EQ.0)  CALL LINK (1HD,0,0)     #.....SAVE INPUT
*ENDIF
*IF HP1000&BATCH
      IF (KFLAG.EQ.0)  CALL LINK (1H4,0,0)
*ENDIF
*IF !HP1000
      IF (KFLAG.EQ.0) CALL OUTP1 (0,0)
*ENDIF
      IF ((KFLAG.NE.0).AND.(KFLAG.NE.-1)) [    #.....PROCESS OTHER OUTPUT
          KOUNT=1
          REPEAT [                             #.....OUTPUT LOOP
              KDISP=KFLAG
*IF HP1000&!BATCH
              CALL LINK (1HD,KOUNT,KDISP)      #.....GET OUTPUT DATA
*ENDIF
*IF HP1000&BATCH
              CALL LINK (1H4,KOUNT,KDISP)
*ENDIF
*IF !HP1000
              CALL OUTP1(KOUNT,KDISP)
*ENDIF
*IF GRAPHICS&HP1000
              IF (KDISP.NE.0) [
                  IF (KDISP.EQ.2) CALL LINK (1HE,KDISP)  # 2-D PLOT
                  IF (KDISP.EQ.3) CALL LINK (1HF,KDISP)  # 3-D PLOT
                  IF (KDISP.EQ.4) CALL LINK (1HG,KDISP)  # CONTOUR PLOT
              ]
*ENDIF
*IF GRAPHICS&!HP1000
              IF (KDISP.NE.0) [
                  IF (KDISP.EQ.2) CALL OUTP2 (KDISP)
                  IF (KDISP.EQ.3) CALL OUTP3 (KDISP)
                  IF (KDISP.EQ.4) CALL OUTP4 (KDISP)
              ]
*ENDIF
              KOUNT=KOUNT+1                    # CHECK IF ANOTHER DISPLAY
*IF HP1000&!BATCH
              CALL LINK (1HD,KANSWR,-KMSG)
*ENDIF
*IF HP1000&BATCH
              CALL LINK (1H4,KANSWR,-KMSG)
*ENDIF
*IF !HP1000
              CALL OUTP1 (KANSWR,-KMSG)
*ENDIF
              IF (KANSWR.EQ.KYES) KGO=1
              ELSE                KGO=0
          ] UNTIL (KGO.EQ.0)
      ]                                        #.....DONE
      RETURN
      END
```

```
*CALL TWDEFN
      SUBROUTINE PAPER (IGCB,XONY,KONSOL)
#
#     CHECK PAPER SIZE FOR THE PLOTTING ON HP9872
#
      DIMENSION IGCB(192)
#
      DATA KYES/2HY /
      DATA XLEFTO/ 25.0(EXP)0/,XRITEO/190.0(EXP)0/,^
           YLOWRO/ 10.0(EXP)0/,YUPPRO/260.0(EXP)0/,^
           XFULLO/215.9(EXP)0/,XHALFO/108.0(EXP)0/,^
           YFULLO/279.4(EXP)0/,YHALFO/139.7(EXP)0/,^
           EDGE  / 15.0(EXP)0/
#
#        279 mm = 11. inch,   215 mm = 8.5 inch
#         25 mm = 2.5 cm,     190 mm = 8.5 inch - 1 cm
#         10 mm = 1.0 cm,     260 mm = 11. inch - 1 cm
#
#     Thesis format : Horizontal -> Top margin = 1.5", Botton margin = 1.0"
#                                   Left         1.0", Right         = 1.0"
#
#
#.....CASE OF STANDARD SIZE 11X8.5
      WRITE (KONSOL,1000)
1000  FORMAT("Is it a 11'x8.5' paper (Yes=f7/No=f8) ? _")
      READ  (KONSOL,2000) KANSWR
2000  FORMAT(A1)
      IF (KANSWR.EQ.KYES) [
          WRITE (KONSOL,1020)
1020      FORMAT("Is it vertical (Yes=f7/No=f8) ? _")
          READ  (KONSOL,2000) KORNT
          WRITE (KONSOL,1030)
1030      FORMAT("1 or 2 figure(s) on the paper ? _")
          READ  (KONSOL,*) KNUMBR
          KNUMBR=MINO(2,MAXO(1,KNUMBR))
          IF (KORNT.EQ.KYES) [
              IF (KNUMBR.EQ.2) [
                  XLEFT=EDGE
                  XRITE=XFULLO-EDGE
                  WRITE (KONSOL,1040)
1040              FORMAT("On the upper part of the paper (Yes=f7/No=f8) ? _")
                  READ  (KONSOL,2000) KANSWR
                  IF (KANSWR.EQ.KYES) [
                      YLOWR=YHALFO-EDGE*0.5(EXP)0
                      YUPPR=YFULLO-EDGE-EDGE
                  ]
                  ELSE [
                      YLOWR=EDGE+EDGE
                      YUPPR=YHALFO+EDGE*0.5(EXP)0
                  ]
              ]
              ELSE [
                  XLEFT=0.0(EXP)0
                  XRITE=XFULLO-EDGE
                  YLOWR=EDGE+EDGE
                  YUPPR=YFULLO
              ]
          ]
          ELSE [
              IF (KNUMBR.EQ.2) [
                  YLOWR=XLEFTO
                  YUPPR=XRITEO
                  WRITE (KONSOL,1050)
1050              FORMAT("On the left part of the paper (Yes=f7/No=f8) ? _")
                  READ  (KONSOL,2000) KANSWR
                  IF (KANSWR.EQ.KYES) [
                      XLEFT=0.0(EXP)0
                      XRITE=YHALFO
                  ]
                  ELSE [
                      XLEFT=YHALFO-EDGE-EDGE
                      XRITE=YFULLO-EDGE-EDGE
                  ]
              ]
              ELSE [
                  XLEFT=YLOWRO
                  XRITE=YUPPRO
                  YLOWR=XLEFTO-EDGE*0.5(EXP)0
                  YUPPR=XRITEO-EDGE*0.5(EXP)0
              ]
```



```
          ]
          CALL LIMIT(IGCB,XLEFT,XRITE,YLOWR,YUPPR)
          XONY=(XRITE-XLEFT)/(YUPPR-YLOWR)
      ]
#
#.....CASE OF NON-STANDARD PAPER SIZE
      ELSE [
          WRITE (KONSOL,1010)
          CALL PEN(IGCB,4)
          CALL LIMIT(IGCB)
          CALL PEN(IGCB,0)
          WRITE (KONSOL,9000)
          READ  (KONSOL,*) XONY
      ]
1010  FORMAT ("When 'ENTER' key is lit, do the following:"^
             /,"        (1) Move pen to lower left corner of paper"^
             /,"        (2) Mark position for future alignment"^
             /,"                by pressing PEN-DOWN followed by PEN-UP"^
             /,"                (or align with previous mark on paper)"^
             /,"        (3) Press the ENTER key"^
             /,"        (4) Move pen to upper right corner"^
             /,"        (5) Mark or align as in step (2)"^
             /,"        (6) Press the ENTER key")
9000  FORMAT("X/Y plot ratio (suggest 1.4 if horizontal, else 0.7) ? _")
#
#.....DONE
      RETURN
      END
```

```
*CALL TUDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE PARMS
#
#     CALCULATE SOME USEFUL PARAMETERS
#
*CALL TEMA
*CALL TCOMMN
#
#     XJCT (1) -> ESTIMATED JUNCTION DEPTH (IN SUBROUTINE CHKDP) OF WELL IMPL
#           2                                                       LOCALIZED
#           3                                                       SRC/DRN
#
#     CSURF(1) -> AVERAGE CONC OF     WELL + LOCALIZED IMPL'NTS
#           2                         WELL + SRC/DRN
#           3                         WELL + LOCAL + SRC/DRN
#     WSURF(K) -> ESTIMATED DEPTH OF  CSURF(K)
#     CIMPL(K) -> AVERAGE CONC OF     WELL IMPL'NT
#           2                         LOCALIZED IMPL'NT
#           3                         WELL + LOCALIZED IMPL'NT
#     WIDTH(K) -> DEPTH OF            CIMPL(K)
#     STYPE(K) -> NET DOPANT TYPE OF  CIMPL(K)+CSUB
#     PHIF (K) -> PHIF             OF CIMPL(K)+CSUB
#     GAMM (K) -> GAMMA            OF CIMPL(K)+CSUB
#     ALPHA(K) -> ALPHA            OF CIMPL(K)
#     PHIMP(K) -> PHIMP            OF CIMPL(K)
#
      DIMENSION CIMP(3),WIDTH(3)
#
      EQUIVALENCE (WIDTH(1),XJCT(1))
#
      DATA UM/1(EXP)4/
#
#.....AVERAGE WIDTH AND CONCENTRATION OF COMBINED IMPLANT PROFILE
      WSURF(1)=AMAX1(XJCT(1),XJCT(2))
      IF (WSURF(1).NE.0.0(EXP)0) CSURF(1)=(DOSE(1)+DOSE(2))/WSURF(1)
      ELSE                       CSURF(1)=0.0(EXP)0
      WSURF(2)=AMAX1(XJCT(1),XJCT(3))
      IF (WSURF(2).NE.0.0(EXP)0) CSURF(2)=(DOSE(1)+DOSE(3))/WSURF(2)
      ELSE                       CSURF(2)=0.0(EXP)0
      WSURF(3)=AMAX1(WSURF(1),WSURF(2))
      IF (WSURF(3).NE.0.0(EXP)0) CSURF(3)=(DOSE(1)+DOSE(2)+DOSE(3))/WSURF(3)
      ELSE                       CSURF(3)=0.0(EXP)0
#
#.....SUBSTRATE RELATED PARAMETERS
      CSUB=ABS(CSUB)
      VFB=0.0(EXP)0
      PHIB=VT300K*ALOG(CSUB/CNI)
      TWOQE=(Q+Q)*EPSI
      QON2E=Q/(EPSI+EPSI)
      GAMMB=SQRT(TWOQE*CSUB)
      ALPHB=QON2E*CSUB
      CMINRO=CNI*CNI/CSUB
      PHIJ=VT300K*ALOG(ABS(CSUB*(CONC(NXMAX,1)))/(CNI*CNI))
#
#.....CHANNEL RELATED PARAMETERS
      CIMP(1)=CSTEP(1)
      CIMP(2)=CSTEP(2)
      CIMP(3)=CSURF(1)
      WIDTH(3)=WSURF(1)
      DO K=1,3 [
          CTOTL=CIMP(K)+SIGN(CSUB,TYPE)
          STYPE(K)=SIGN(1.0(EXP)0,CTOTL)
          CIMP(K)=ABS(CIMP(K))
          CTOTL=ABS(CTOTL)
          IF (CTOTL.NE.0.0(EXP)0) PHIF(K)=VT300K*ALOG(CTOTL/CNI)
          ELSE                    PHIF(K)=0.0(EXP)0
          GAMM(K)=SQRT(TWOQE*CTOTL)
          ALPHA(K)=QON2E*CIMP(K)
          W2=WIDTH(K)*WIDTH(K)
          PHIMP(K)=W2*QON2E*CIMP(K)
          VBCRT(K)=PHIMP(K)-(PHIB+PHIF(K))
      ]
#
#.....CORRECT SOURCE AND DRAIN INDICES
#     NSOURC & NDRAIN = INDEX OF THE LAST POINT WITHIN THE DIFFUSION
      NSOURC=NSRCO
      NDRAIN=NDRNO
```



```
      IF (((NSOURC.GE.1).OR.(NDRAIN.LE.NXMAX)).AND.(DOSE(3).NE.0.0(EXP)0)) [
          NC=(NSOURC+NDRAIN)/2
          TPC=SIGN( 1.0(EXP)0,(CONC(NC,1)) )
#
#.....ENHANCEMENT MOSFET
          IF (TYPE.EQ.TPC) [
              IF (NSOURC.GE.1) [
                  TPS=-TYPE
                  REPEAT [
                      NSOURC=NSOURC+1
                      TPS=SIGN( 1.0(EXP)0,(CONC(NSOURC,1)) )
                  ] UNTIL (TPS.EQ.TYPE)
                  NSOURC=NSOURC-1
              ]
              IF (NDRAIN.LE.NXMAX) [
                  TPS=-TYPE
                  REPEAT [
                      NDRAIN=NDRAIN-1
                      TPS=SIGN( 1.0(EXP)0,(CONC(NDRAIN,1)) )
                  ] UNTIL (TPS.EQ.TYPE)
                  NDRAIN=NDRAIN+1
              ]
          ]
#
#.....BURIED MOSFET
          ELSE [
              IF (NSOURC.GE.1) [
                  GRADO=0.0(EXP)0
                  CX=CONC(NSOURC,1)
                  CX1=CONC(NSOURC+1,1)
                  GRAD1=ABS( (CX1-CX)/(0.5(EXP)0*(CX1+CX)*DELX(NSOURC)) )
                  REPEAT [
                      NSOURC=NSOURC+1
                      GRADO=GRAD1
                      CX =CX1
                      CX1=CONC(NSOURC+1,1)
                      GRAD1=ABS( (CX1-CX)/(0.5(EXP)0*(CX1+CX)*DELX(NSOURC)) )
                  ] UNTIL (GRADO.GT.GRAD1)
              ]
              IF (NDRAIN.LE.NXMAX) [
                  GRADO=0.0(EXP)0
                  CX=CONC(NDRAIN,1)
                  CX1=CONC(NDRAIN-1,1)
                  GRAD1=ABS( (CX1-CX)/(0.5(EXP)0*(CX1+CX)*DELX(NDRAIN-1)) )
                  REPEAT [
                      NDRAIN=NDRAIN-1
                      GRADO=GRAD1
                      CX =CX1
                      CX1=CONC(NDRAIN-1,1)
                      GRAD1=ABS( (CX1-CX)/(0.5(EXP)0*(CX1+CX)*DELX(NDRAIN-1)) )
                  ] UNTIL (GRADO.GT.GRAD1)
              ]
          ]
#
#.....OUTPUT
          XSO=XPOS(NSRCO)*UM
          XDO=XPOS(NDRNO)*UM
          XS=XPOS(NSOURC)*UM
          XD=XPOS(NDRAIN)*UM
          DS=XS-XSO
          DD=XDO-XD
          XCH=XD-XS
          WRITE (KONSOL,1000) XSO,XDO,NSRCO,NDRNO,XS,XD,NSOURC,NDRAIN,DS,DD,XCH
1000      FORMAT (/,"Drawn source/drain junctions at ",^
                 "(",F7.3,"um,"F7.3,"um).....(",I3,",",I3,")",^
               /,"Corrected by side diffusions as ",^
                 "(",F7.3,"um,"F7.3,"um).....(",I3,",",I3,")",^
               /,"Lateral diffusion length of s/d:(",F7.3,"um,"F7.3,"um)"^
               /,"Effective channel length:",F7.3,"um"/)
      ]
#
#.....DONE
      RETURN
      END
```

File: &PHIF1

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE PHIF1 (NX,NY,KFLAG)
#
#     DEFINE QUASI-FERMI LEVEL FOR MAJORITY AND MINORITY CARRIERS
#
*CALL TEMA
*CALL TCOMMN
#
      DTOL=0.67(EXP)0*ABS((CONC(NX,NY)))
      PHIFP=0.0(EXP)0
      PHIFN=0.0(EXP)0
#
#.....SOURCE AND/OR DRAIN REGIONS, OR CHANNEL REGION IN INITIALIZATION
      IF ((NX.LE.NSOURC).OR.(NX.GE.NDRAIN) OR.(KFLAG.EQ.1)) [
          IF ((NX.LE.NSOURC).OR.^
             ((NX.GT.NSOURC).AND.(NX.LT.NDRAIN).AND.(KFLAG.EQ.1))) VBIAS=VSB
          ELSE                                                      VBIAS=VDB
#
#.....INSIDE DEPLETION LAYER
          DENSTY=CONC(NX,NY)+CARRIE(NX,NY)
          IF (ABS(DENSTY).GT.DTOL) [
              PHIFP=0.0(EXP)0
              PHIFN=VBIAS
          ]
#
#.....IN NEUTRAL N+ REGION
          ELSE [
              TPX=SIGN(1.0(EXP)0,(CONC(NX,NY)))
              IF (TPX.NE.TYPE) [
#
#.....IN BUIRED CHANNEL
                  IF (KFLAG.EQ.1) [
                      PHIFP=0.0(EXP)0
                      PHIFN=VBIAS
                  ]
#
#.....SOURC/DRAIN
                  ELSE [
                      PHIFP=VBIAS
                      PHIFN=VBIAS
                  ]
              ]
          ]
      ]
#
#.....CHANNEL REGION
      ELSE [
#          IF (POTSI(NX-1,NY).GE.POTSI(NX+1,NY)) PHIFN=VSB
#          ELSE                                  PHIFN=VDB
#       ]
          IF (KFLAG.EQ.0) CALL PHIF2
          ELSE [
              NX1=LOCPHF(1,NY)
              NX2=LOCPHF(2,NY)
              IF      (NX.LE.NX1) PHIFN=VSB
              ELSE IF (NX.GE.NX2) PHIFN=VDB
                   ELSE [
                        X1=XPOS(NX1)
                        X2=XPOS(NX2)
                        DX=X2-X1
                        PHIFN=COFPHF(2,NY)+COFPHF(1,NY)*EXP((XPOS(NX)-X1)/DX)
                   ]
          ]
      ]
#
#.....DONE
      RETURN
      END
```


File: &PHIF2

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE PHIF2
#
# ....DEFINE QUASI FERMI LEVEL
#
*CALL TEMA
*CALL TCOMMN
#
#
#.....SEARCH FOR POTENTIAL MINIMUM
      TWOVT=VT300K+VT300K
      DO M=1,NYMAX [
          PHIMIN=99.99(EXP)0
          NC0=NSOURC
          N1=NSOURC+1
          N2=NDRAIN-1
          DO N=N1,N2 [
              POTX=POTSI(N,M)
              IF (POTX.LT.PHIMIN) [
                  PHIMIN=POTX
                  NC0=N
              ]
          ]
#
#.....SEARCH FOR KT/Q DROP
          NC1=NC0
          POTX=PHIMIN
          WHILE (((POTX-PHIMIN).LE.VT300K).AND.(NC1.LE.NDRAIN)) [
              NC1=NC1+1
              POTX=POTSI(NC1,M)
          ]
          LOCPHF(1,M)=NC1
#
#.....SEARCH FOR DRAIN LIMIT OF CHANNEL REGION
          NC2=NC1
          WHILE (((POTX-PHIMIN).LE.TWOVT).AND.(NC2.LE.NDRAIN)) [
              NC2=NC2+1
              POTX=POTSI(NC2,M)
          ]
          LOCPHF(2,M)=NC2
#
#.....DEFINE COEFFICIENTS
          COFPHF(1,M)=(VDB-VSB)/(EXP(1.0(EXP)0)-1.0(EXP)0)
          COFPHF(2,M)=VSB-COFPHF(1,M)
      ]
#
#.....DONE
      RETURN
      END
```

File: &PHMIN

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE PHMIN (M,PMIN,LMIN)
#
#     LOCATE POTENTIAL MINIMUM
#
*CALL TEMA
*CALL TCOMMN
#
      TWOVT=VT300K+VT300K
      LMIN=NSOURC
      PMIN=POTSI(LMIN,M)
      N1=NSOURC+1
      DO N=N1,NDRAIN [
          POTNM=POTSI(N,M)
          IF (POTNM.LT.PMIN) [ LMIN=N;     PMIN=POTNM ]
      ]
      RETURN
      END
```


File: &PLOT2

```
*CALL TWDEFN
#
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE PLOT2 (X1,X2,Y1,Y2,N1,N2,LABLX,LABLY,LABLZ,^
                       KFONTX,KFONTY,KFONTZ,KLOG,LU,KONSOL)
#
#     2-DIMENSIONAL PLOT
#
#       DATA: X=Z(N,1),      Y=Z(N,2)      -> N=N1,N2
#       FONT: KFONTX(2,10),KFONTY(2,10) -> UP TO 10 CHANGES
#      LABEL: LABLX(40), LABLY(40), LABLZ(40) -> UP TO 80 CH (INCLUDING CTL CH)
#
*CALL TEMA
#
#.....PLOTTING PARAMETERS
      DIMENSION KGCB(192),KDCB(144),LABLX(1),LABLY(1),LABLZ(1),^
               KFONTX(2,11),KFONTY(2,10),KFONTZ(2,10)
#
      DATA LUPLTR/9/, ISPEED/2/, KESC/033B/,     KBELL/7/, KYES/2HY /
#
#
#.....LOOP OF PLOTTING
      REPEAT [
          IF (LU.NE.LUPLTR) [                           # SET UP THE KONSOL
              KTYPE=1; CALL PLOTR(KGCB ,KTYPE,1,LU)
              XONY=2.0(EXP)0
              WRITE (KONSOL,3000) KESC
3000          FORMAT(R1,"*dF_")
          ]
          ELSE [                                        # SET UP THE PLOTTOR
              KTYPE=2; CALL PLOTR (KGCB,KTYPE,1,LU)
              CALL PAPER (KGCB,XONY,KONSOL)
              WRITE (KONSOL,4000)
              READ  (KONSOL,*) KOLOR
4000          FORMAT ("Which color? (1-4, as set up on the plotter) _")
              KOLOR=MIN0(4,MAX0(KOLOR,1))
              CALL PEN (KGCB,KOLOR)
              CALL XMIT (KGCB)
              WRITE (LU,4010) ISPEED
4010          FORMAT("VS",I1,";VA;")
          ]
          CALL ASCAL (KGCB,XONY,X1,X2,Y1,Y2)          # SCALE SURFACE
          CALL LINE (KGCB,0)                          # SOLID LINE
          CALL MOVE (KGCB,X1,Y1)                      # DRAW THE FRAME
          CALL DRAW (KGCB,X1,Y2)
          CALL DRAW (KGCB,X2,Y2)
          CALL DRAW (KGCB,X2,Y1)
          CALL DRAW (KGCB,X1,Y1)
          YP=AMIN1(Y2,AMAX1(Y1,(Z(N1,2))))            # DRAW THE CURVE
          CALL MOVE (KGCB,X1,YP)
          NS=N1+1
          DO N=NS,N2 [
              XP=AMIN1(X2,AMAX1(X1,(Z(N,1))))
              YP=AMIN1(Y2,AMAX1(Y1,(Z(N,2))))
              CALL DRAW (KGCB,XP,YP)
          ]
          CALL PENUP (KGCB)
#
#.....CHECK IF ADD LABELS
          WRITE (KONSOL,4050) KBELL,KESC
          READ  (KONSOL,2000) KANSWR
4050      FORMAT(2R1,"*dEAdd labels (Yes=f7/No=f8) _")
          IF (KANSWR.EQ.KYES) [
              IF (LU.EQ.KONSOL) WRITE (KONSOL,3000) KESC
              IF (KLOG.EQ.0) CALL AXLIN (KGCB,KDCB,XONY,X1,X2,Y1,Y2)
              ELSE           CALL AXLOG (KGCB,KDCB,XONY,X1,X2,Y1,Y2)
              CALL AXLAB (KGCB,KDCB,XONY,X1,X2,Y1,Y2,LABLX,LABLZ,KFONTX,KFONTZ)
              WRITE (KONSOL,5000) KBELL,KESC
5000          FORMAT(2R1,"*dE_")
          ]
#
#.....FINISH THE PLOT
          IF (LU.EQ.LUPLTR) CALL PEN (KGCB,0)
          CALL PLOTR (KGCB ,KTYPE,0)
#
#.....CHECK IF DUMP TO PLOTTER
          KGO=0
          IF (LU.NE.LUPLTR) [
```

File: &PLOT2

```
            WRITE (KONSOL,6000)
            READ  (KONSOL,2000) KANSUR
            IF (KANSUR.EQ.KYES) {   KGO=1;  LU=LUPLTR }
         }
      } UNTIL (KGO.EQ.0)
6000  FORMAT("Replot on the plotter (Yes=f7/No=f8) ? _")
2000  FORMAT(A1)
#
#.....DONE
      RETURN
      END
```


File: &PLOT3

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE PLOT3 (X1,X2,Y1,Y2,Z1,Z2,NX,NY,ROTAT,TILT,LU,KONSOL)
#
#
*CALL TEMA
#
#.....PLOTTING PARAMETERS
      DIMENSION KGCB(192)
      DATA LUPLTR/9/,ISPEED/2/,KESC/033B/,KBELL/7/,KYES/2HY /
#
#***************     INITIALIZATION     ***************
#
      CALL ANGLE (ROTAT,KQR);       CALL ANGLE (TILT ,KQT)
      COSR=COS(ROTAT);              COST=COS(TILT)
      SINR=SIN(ROTAT);              SINT=SIN(TILT)
#
      XSPAN=X2-X1
      YSPAN=Y2-Y1;                  ZSPAN=Z2-Z1
      XOVRY=XSPAN/YSPAN;            XOVRZ=XSPAN/ZSPAN
#
      DXPDX= SINR;                  DXPDY=COSR*XOVRY
      DYPDX=-COSR*SINT;             DYPDY=SINR*SINT*XOVRY;     DYPDZ=COST*XOVRZ
#
      XPX1=X1*DXPDX;                XPX2=X2*DXPDX
      XPY1=Y1*DXPDY;                XPY2=Y2*DXPDY
      YPX1=X1*DYPDX;                YPX2=X2*DYPDX
      YPY1=Y1*DYPDY;                YPY2=Y2*DYPDY
#
      XP11=XPX1+XPY1;               XP12=XPX1+XPY2
      XP21=XPX2+XPY1;               XP22=XPX2+XPY2
      YP11=YPX1+YPY1;               YP12=YPX1+YPY2
      YP21=YPX2+YPY1;               YP22=YPX2+YPY2
#
      YPZ1=Z1*DYPDZ;                YPZ2=Z2*DYPDZ
#
      XLIM0=AMIN1(XP11,XP12,XP21,XP22)
      XLIM1=AMAX1(XP11,XP12,XP21,XP22)
      YLIM0=AMIN1(YP11,YP12,YP21,YP22)+YPZ1
      YLIM1=AMAX1(YP11,YP12,YP21,YP22)+YPZ2
#
      DX=XSPAN/(NX-1);              DY=YSPAN/(NY-1)
      DXPX=DX*DXPDX;                DXPY=DY*DXPDY
      DYPX=DX*DYPDX;                DYPY=DY*DYPDY
#
#.....INITIALIZE THE DEVICE
      REPEAT {
         IF (LU.NE.LUPLTR) {
            KTYPE=1; CALL PLOTR(KGCB ,KTYPE,1,LU)
            XONY=2.0(EXP)0
            WRITE (KONSOL,3000) KESC
3000        FORMAT(R1,"*dF_")
         }
         ELSE {
            KTYPE=2; CALL PLOTR (KGCB,KTYPE,1,LU)
            CALL PAPER (KGCB,XONY,KONSOL)
            WRITE (KONSOL,4000)
            READ (KONSOL,*) KOLOR
4000        FORMAT ("Which color? (1-4, as set up on the plotter) _")
            KOLOR=MIN0(4,MAX0(KOLOR,1))
            CALL PEN  (KGCB,KOLOR)
            CALL XMIT (KGCB)
            WRITE (LU,4010) ISPEED
4010        FORMAT("VS",I1,";VA;")
         }
         CALL LINE (KGCB,0)
#
#.....SCALE THE PLOTTING PLANE
         CALL ASCAL (KGCB,XONY,XLIM0,XLIM1,YLIM0,YLIM1)
#
#***************     PLOTTING SECTION     ***************
#
#.....DRAW Y-LINES
         KD=0
         X=X1-DX
         XPLOT=XP11-DXPX
         YPLOT=YP11-DYPX
         DO KX=1,NX {
```

144

```
            KD=1-KD
            X=X+DX
            XPLOT=XPLOT+DXPX
            YPLOT=YPLOT+DYPX
            IF (KD.EQ.1) [
                Y1S=Y1
                Y2S=Y2
                DYS=DY
                DXPS=DXPY
                DYPS=DYPY
            ]
            ELSE [
                Y1S=Y2
                Y2S=Y1
                DYS=-DY
                DXPS=-DXPY
                DYPS=-DYPY
            ]
            ZVAL=AMIN1(Z2,AMAX1(FUNC(X,Y1S),Z1))
            Y=Y1S
            YP=YPLOT+ZVAL*DYPDZ
            CALL MOVE (KGCB,XPLOT,YP)
            DO KY=2,NY [
                Y=Y+DYS
                XPLOT=XPLOT+DXPS
                YPLOT=YPLOT+DYPS
                ZVAL=AMIN1(Z2,AMAX1(FUNC(X,Y),Z1))
                YP=YPLOT+ZVAL*DYPDZ
                CALL DRAW (KGCB,XPLOT,YP)
            ]
            CALL PENUP (KGCB)
        ]
#
#.....DRAW X-LINES
        KD=0
        Y=Y1-DY
        XPLOT=XP21-DXPY
        YPLOT=YP21-DYPY
        DO KY=1,NY [
            KD=1-KD
            Y=Y+DY
            XPLOT=XPLOT+DXPY
            YPLOT=YPLOT+DYPY
            IF (KD.EQ.1) [
                X1S=X2
                X2S=X1
                DXS=-DX
                DXPS=-DXPX
                DYPS=-DYPX
            ]
            ELSE [
                X1S=X1
                X2S=X2
                DXS=DX
                DXPS=DXPX
                DYPS=DYPX
            ]
            ZVAL=AMIN1(Z2,AMAX1(FUNC(X1S,Y),Z1))
            X=X1S
            YP=YPLOT+ZVAL*DYPDZ
            CALL MOVE (KGCB,XPLOT,YP)
            DO KX=2,NX [
                X=X+DXS
                XPLOT=XPLOT+DXPS
                YPLOT=YPLOT+DYPS
                ZVAL=AMIN1(Z2,AMAX1(FUNC(X,Y),Z1))
                YP=YPLOT+ZVAL*DYPDZ
                CALL DRAW (KGCB,XPLOT,YP)
            ]
            CALL PENUP (KGCB)
        ]
#
#.....DRAW XZ-AND YZ-PLANES
        CALL LINE (KGCB,1)
        CALL MOVE (KGCB,XP11,YP11+YPZ1)
        CALL DRAW (KGCB,XP21,YP21+YPZ1)
        CALL DRAW (KGCB,XP21,YP21+YPZ2)
        CALL DRAW (KGCB,XP11,YP11+YPZ2)
        CALL DRAW (KGCB,XP11,YP11+YPZ1)
        CALL DRAW (KGCB,XP12,YP12+YPZ1)
```



```
        CALL DRAW (KGCB,XP12,YP12+YPZ2)
        CALL DRAW (KGCB,XP11,YP11+YPZ2)
#
#.....COMPLETE OTHER PLANES
        CALL LINE (KGCB,2)
        CALL MOVE (KGCB,XP12,YP12+YPZ2)
        CALL DRAW (KGCB,XP22,YP22+YPZ2)
        CALL DRAW (KGCB,XP22,YP22+YPZ1)
        CALL DRAW (KGCB,XP21,YP21+YPZ1)
        CALL MOVE (KGCB,XP21,YP21+YPZ2)
        CALL DRAW (KGCB,XP22,YP22+YPZ2)
        CALL MOVE (KGCB,XP22,YP22+YPZ1)
        CALL DRAW (KGCB,XP12,YP12+YPZ1)
#
#.....FINISH THE PLOT
        CALL PENUP (KGCB)
        IF (LU.EQ.LUPLTR) CALL PEN (KGCB,0)
        CALL PLOTR (KGCB ,KTYPE,0)
#
#.....RING THE BELL
        WRITE (KONSOL,5000) KBELL,KESC
5000    FORMAT(2R1,"*dE_")
#
#.....CHECK IF DUMP TO PLOTTER
        KGO=0
        IF (LU.NE.LUPLTR) [
            WRITE (KONSOL,6000)
            READ  (KONSOL,2000) KANSWR
            IF (KANSWR.EQ.KYES) [ KGO=1; LU=LUPLTR ]
        ]
      ] UNTIL (KGO.EQ.0)
6000  FORMAT("Replot on the plotter (Yes=f7/No=f8) ? _")
2000  FORMAT(A1)
#
#.....DONE
      RETURN
      END
```

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE PLOTC (VMIN,VMAX,NUMLIN,LU)
#
#     CONTOUR PLOT
#       Z(50,40) = LINE SEARCHING STATUS
#         STORED  BY SUBROUTINE         LHMRK
#         LOADED  BY INTEGER FUNCTION LNCHK
#
*CALL TEMA
*CALL TCOMMN
#
      DIMENSION VLINE(20)
#
#.....PLOTTING PARAMETERS
      DIMENSION KGCB(192), KDCB(144), LBUF(25)
      DATA LUPLTR/9/, ISPEED/2/,  KESC/033B/,    KBELL/7/, KYES/2HY /
      DATA UM/1.0(EXP)4/, CHARH/25.0(EXP)0/
      DATA NSEC/0/,  NCR/2HGG/, ARTIO/0.55(EXP)0/
      DATA NAMX/2HX /, NAMY/2HY /,    NAMVG/2Hgs/, NAMVD/2Hds/, NAMVB/2Hbs/
#
      DATA NLMAX/15/
#
2000  FORMAT (A1)
#
#.....DETERMINE POTENTIAL VALUES
      NUMLIN=MIN0(NLMAX,MAX0(2,NUMLIN))
      VLINE(1)=VMIN
      DVAL=(VMAX-VMIN)/(NUMLIN-1)
      DO K=2,NUMLIN; VLINE(K)=VLINE(K-1)+DVAL
      WRITE (KONSOL,1000) (VLINE(K),K=1,NUMLIN)
1000  FORMAT ("**Line value(K)"/(3X,10F7.4))
#
#.....INITIALIZE THE PLOTTING DEVICE
      REPEAT [
          IF (LU.NE.LUPLTR) [
              KTYPE=1; CALL PLOTR(KGCB ,KTYPE,1,LU)
              XONY=2.0(EXP)0
              WRITE (KONSOL,3000) KESC
3000          FORMAT(R1,"*dF_")
          ]
          ELSE [
              KTYPE=2; CALL PLOTR (KGCB,KTYPE,1,LU)
              CALL PAPER (KGCB,XONY,KONSOL)
              WRITE (KONSOL,4000)
              READ  (KONSOL,*) KOLOR
4000          FORMAT("Which color? (1-4, as set up on the plotter) _")
              KOLOR=MIN0(4,MAX0(KOLOR,1))
              CALL PEN  (KGCB,KOLOR)
              CALL XMIT (KGCB)
              WRITE (LU,4010) ISPEED
4010          FORMAT("VS",I1,";VA;")
          ]
#
#.....OUTLINE DEVICE, SCALE AS (XPOS(1),XPOS(NXMAX),-YMAX,TOX)
#           AND SET TEXT SIZE AND QUALITY
          CALL PLOTD (KGCB,XONY)
          CALL TSCAL (KGCB,XONY,CHARH,CHITE,XLENG,YLENG)
          CALL LINE (KGCB,0)
#
#.....INITIALIZE INDEX ARRAY
          N1=NXMAX-1
          M1=NYMAX-1
          DO L=1,NUMLIN; CALL LHMRK (0,0,0)
#
#.....LOOP THROUGH ALL MESHES
          DO N=1,N1 [
              POT1=POTSI(N,1)
              DO M=1,M1 [
                  POT0=POT1
                  POT1=POTSI(N,M+1)
                  POT2=POTSI(N+1,M)
#
#.....LOOP THROUGH ALL LINES
                  DO L=NUMLIN,1,-1 [
                      POTL=VLINE(L)
                      SIGN0=SIGN(1.0(EXP)0,POTL-POT0)
#
```



```
#.....CHECK IF PASSING THROUGH THE NEXT Y-MESH
                      IF (LNCHK(N,M,L).EQ.0) [
                          SIGN1=SIGN(1.0(EXP)0,POTL-POT1)
                          IF ((POTL.EQ.POT0).OR.(SIGN0.NE.SIGN1))
                              CALL TRACE (KGCB,KDCB,N,M,L,POTL)
                      ]
#
#.....CHECK IF PASSING THROUGH THE NEXT X-MESH
                      IF ( (LNCHK(N,M,-L).EQ.0).AND.(POTL.NE.POT0) ) [
                          SIGN2=SIGN(1.0(EXP)0,POTL-POT2)
                          IF (SIGN0.NE.SIGN2)
                              CALL TRACE (KGCB,KDCB,N,M,-L,POTL)
                      ]
                  ]
              ]
          ]
#
#.....LABLING
          IF (LU.EQ.KONSOL) WRITE (KONSOL,6000) KBELL,KESC
          WRITE (KONSOL,5000)
          READ (KONSOL,2000) KANSWR
5000      FORMAT("Add labels (Yes=f7/No=f8) ? _")
          IF (KANSWR.EQ.KYES) [
          IF (LU.EQ.KONSOL) WRITE (KONSOL,3000) KESC
#
#.....DRAW UPPER BRACKET
              TOX=AMAX1(TOX1,TOX0)
              CHITY=CHITE*(TOX+YMAX)/YLENG
              CHITX=CHITY*(XMAX-XSOURC)/(YMAX*XONY)
              CHYON2=CHITY*0.5(EXP)0
              CHXON2=CHITX*0.5(EXP)0
              YU0=TOX+CHYON2
              YU1=YU0+CHITY
              XU0=XPOS(NSOURC+1)
              XU1=XPOS(NDRAIN-1)
              CALL LINE (KGCB,1)
              CALL MOVE (KGCB,0.0(EXP)0,YU0)
              CALL DRAW (KGCB,0.0(EXP)0,YU1)
              XP=0.0(EXP)0
              DX=XU0*0.1(EXP)0
              DO K=1,10 [ XP=XP+DX; CALL DRAW (KGCB,XP,YU1) ]
              CALL DRAW (KGCB,XU0,YU0)
              CALL MOVE (KGCB,XU0,YU1)
              XP=XU0
              DX=(XU1-XU0)*0.1(EXP)0
              DO K=1,10 [ XP=XP+DX; CALL DRAW (KGCB,XP,YU1) ]
              CALL DRAW (KGCB,XU1,YU0)
              CALL MOVE (KGCB,XU1,YU1)
              XP=XU1
              DX=(XCHANL-XU1)*0.1(EXP)0
              DO K=1,10 [ XP=XP+DX; CALL DRAW (KGCB,XP,YU1) ]
              CALL DRAW (KGCB,XCHANL,YU0)
#
#.....TOP LABLE
              ZPRT1=XCHANL*UM
              ZPRT2=(XPOS(NDRAIN)-XPOS(NSOURC))*UM
              CALL CODE; WRITE (LBUF,5010) ZPRT1,ZPRT2
5010          FORMAT ("Drawn L =",F5.2," _m_m,    Effective L =",F5.2," _m_m")
#
#.....GET TEXT LENGTH
              CALL LDIR (KGCB,0.0(EXP)0)
              TXTLEN=0.0(EXP)0
              CALL GFONT (KGCB,6HFONT2 ,NSEC,NCR,KDCB)
              NS=1
              NC=16
              CALL GLEN  (KGCB,LBUF,NS,NC,XT,YT,KDCB)
              TXTLEN=TXTLEN+XT
              CALL GFONT (KGCB,6HFONT5 ,NSEC,NCR,KDCB)
              NS=17
              NC=2
              CALL GLEN  (KGCB,LBUF,NS,NC,XT,YT,KDCB)
              TXTLEN=TXTLEN+XT
              CALL GFONT (KGCB,6HFONT2 ,NSEC,NCR,KDCB)
              NS=19
              NC=25
              CALL GLEN  (KGCB,LBUF,NS,NC,XT,YT,KDCB)
              TXTLEN=TXTLEN+XT
              CALL GFONT (KGCB,6HFONT5 ,NSEC,NCR,KDCB)
              NS=44
              NC=2
```

```
              YP=0.0(EXP)0
              DY=-YMAX*0.1(EXP)0
              DO K=1,10 [ YP=YP+DY; CALL DRAW (KGCB,XL1,YP) ]
              CALL DRAW (KGCB,XL0,-YMAX)
#
#.....LEFT LABLE
              ZPRT=YMAX*UM
              CALL CODE; WRITE (LBUF,5020) NAMY,ZPRT
#
#.....MOVE TO THE LOWEST POINT
              XLZ=XL1-CHXON2
              CALL MOVE (KGCB,XLZ,-YMAX)
#
#.....IN SIMPLEX ROMAN
              PION2=2.0(EXP)0*ATAN2(1.0(EXP)0,1.0(EXP)0)
              CALL LDIR (KGCB,PION2)
              CALL LINE (KGCB,0)
              NS=1
              NC=15
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
              CALL GFONT (KGCB,6HFONT5 ,NSEC,NCR,KDCB)
              NS=16
              NC=2
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
              CALL GFONT (KGCB,6HFONT2 ,NSEC,NCR,KDCB)
              NS=18
              NC=1
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
#
#.....POINT TO OXIDE
              CALL LINE (KGCB,1)
              XL3=XLZ-CHITX
              IF (XONY.GT.1.0(EXP)0) YL3=-YMAX*0.1(EXP)0
              ELSE                   YL3=-YMAX*0.5(EXP)0
              CALL MOVE (KGCB,XL0,TOX)
              CALL DRAW (KGCB,XL3,TOX)
              YP=TOX
              DY=(YL3-TOX)*0.1(EXP)0
              DO K=1,10 [ YP=YP+DY; CALL DRAW (KGCB,XL3,YP) ]
#
#.....OXIDE LABEL
              ZPRT1=TOX0*UM
              ZPRT2=TOX1*UM
              CALL CODE; WRITE (LBUF,5035) ZPRT1,ZPRT2
5035          FORMAT ("Oxide thicknesses = ",F5.3,", ",F5.3," _m_m")
#
#.....MOVE TO THE LOWEST POINT
              XL4=XL3-CHXON2
              CALL MOVE (KGCB,XL4,-YMAX)
#
#.....IN SIMPLEX ROMAN
              CALL LINE (KGCB,0)
              NS=1
              NC=34
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
              CALL GFONT (KGCB,6HFONT5 ,NSEC,NCR,KDCB)
              NS=35
              NC=2
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
              CALL GFONT (KGCB,6HFONT2 ,NSEC,NCR,KDCB)
              NS=37
              NC=1
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
#
#.....POSITION RIGHT LABELS
#              XRO=XCHANL+XDRAIN+CHXON2
#              YRO=0.5(EXP)0*(-YMAX)
#              DYR=YRO*0.25(EXP)0
#              YRO=YRO+DYR
#              CALL LDIR (KGCB,0.0(EXP)0)
#              CALL MOVE (KGCB,XRO,YRO)
#
#.....LABEL VGS
#              VDS=VDB-VSB
#              VGS=VGB-VBS
#              VSB=-VBS
#              CALL CODE; WRITE (LBUF,5030) NAMVG,VGS
#5030          FORMAT("V",A2," = ",F6.2," V")
#              IF (XONY.GT.1.0(EXP)0) [
#                 NS=1
```



```
              CALL GLEN  (KGCB,LBUF,NS,NC,XT,YT,KDCB)
              TXTLEN=TXTLEN+XT
              CALL GFONT (KGCB,6HFONT2 ,NSEC,NCR,KDCB)
              NS=46
              NC=1
              CALL GLEN  (KGCB,LBUF,NS,NC,XT,YT,KDCB)
              TXTLEN=TXTLEN+XT
#
# ....MOVE TO THE LEFT MOST POSITION
              XUZ=(XCHANL-TXTLEN)*0.5(EXP)0
              YUZ=YU1+CHYON2
              CALL MOVE (KGCB,XUZ,YUZ)
#
#.....WRITE IN SIMPLEX ROMAN
              CALL LINE (KGCB,0)
              CALL LDIR (KGCB,0.0(EXP)0)
              CALL LORG (KGCB,1)
              CALL GFONT (KGCB,6HFONT2 ,NSEC,NCR,KDCB)
              NS=1
              NC=16
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
              CALL GFONT (KGCB,6HFONT5 ,NSEC,NCR,KDCB)
              NS=17
              NC=2
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
              CALL GFONT (KGCB,6HFONT2 ,NSEC,NCR,KDCB)
              NS=19
              NC=25
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
              CALL GFONT (KGCB,6HFONT5 ,NSEC,NCR,KDCB)
              NS=44
              NC=2
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
              CALL GFONT (KGCB,6HFONT2 ,NSEC,NCR,KDCB)
              NS=46
              NC=1
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
#
#.....BOTTOM BRACKET
              XD0=-XSOURC
              XD1=XCHANL+XDRAIN
              YD0=-YMAX-CHYON2
              YD1=YD0-CHITY
              CALL LINE (KGCB,1)
              CALL MOVE (KGCB,XD0,YD0)
              CALL DRAW (KGCB,XD0,YD1)
              XP=XD0
              DX=(XD1-XD0)*0.1(EXP)0
              DO K=1,10 [ XP=XP+DX; CALL DRAW (KGCB,XP,YD1) ]
              CALL DRAW (KGCB,XD1,YD0)
#
#.....BOTTOM LABLE
              ZPRT=XMAX*UM
              CALL CODE; WRITE (LBUF,5020) NAMX,ZPRT
5020          FORMAT (A1,"-max = ",F5.2," _m_m")
#
#.....MOVE TO THE LEFT MOST POINT
              YDZ=YD1-CHYON2-CHITY
              CALL MOVE (KGCB,XD0,YDZ)
#
#.....IN SIMPLEX ROMAN
              CALL LINE (KGCB,0)
              NS=1
              NC=15
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
              CALL GFONT (KGCB,6HFONT5 ,NSEC,NCR,KDCB)
              NS=16
              NC=2
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
              CALL GFONT (KGCB,6HFONT2 ,NSEC,NCR,KDCB)
              NS=18
              NC=1
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
#
#.....LEFT BRACKET
              XL0=XD0-CHXON2*XONY
              XL1=XL0-CHITX
              CALL LINE (KGCB,1)
              CALL MOVE (KGCB,XL0,0.0(EXP)0)
              CALL DRAW (KGCB,XL1,0.0(EXP)0)
```

```
#                   NC=14
#                   CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
#                ]
#                ELSE [
#                   NSA=1
#                   NCA=5
#                   CALL GTEXT (KGCB,LBUF,NSA,NCA,KDCB)
#                   NSV=6
#                   NCV=9
#                   CALL GTEXT (KGCB,LBUF,NSV,NCV,KDCB)
#                ]
#
#.....LABEL VDS
#                YRO=YRO+DYR
#                CALL MOVE  (KGCB,XRO,YRO)
#                CALL CODE;  WRITE (LBUF,5030) NAMVD,VDS
#                IF (XONY.GT.1.0(EXP)0) CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
#                ELSE [
#                   CALL GTEXT (KGCB,LBUF,NSA,NCA,KDCB)
#                   CALL GTEXT (KGCB,LBUF,NSV,NCV,KDCB)
#                ]
#
#.....LABEL VBS
#                YRO=YRO+DYR
#                CALL MOVE  (KGCB,XRO,YRO)
#                CALL CODE;  WRITE (LBUF,5030) NAMVB,VBS
#                IF (XONY.GT.1.0(EXP)0) CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
#                ELSE [
#                   CALL GTEXT (KGCB,LBUF,NSA,NCA,KDCB)
#                   CALL GTEXT (KGCB,LBUF,NSV,NCV,KDCB)
#                ]
#
#.....SUBSTRATE DOPING LABEL
#                YRO=YRO+DYR
#                CALL MOVE  (KGCB,XRO,YRO)
#                CALL CODE;  WRITE (LBUF,5040) (IDEV(K),K=1,3)
#5040            FORMAT("Nsub = ",3A2)
#                CALL GFONT (KGCB,6HFONT2 ,NSEC,NCR,KDCB)
#                NS=1
#                NC=13
#                CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
#.....CLOSE FONT FILE
                 CALL GFONT (KGCB,0,0,0,KDCB)
          ]
#
#.....FINISH UP
          CALL PENUP (KGCB)
          IF (LU.EQ.LUPLTR) CALL PEN (KGCB,0)
          CALL PLOTR (KGCB ,KTYPE,0)
#
#.....RING THE BELL
          WRITE (KONSOL,6000) KBELL,KESC
6000      FORMAT(2R1,"*dE_")
#
#.....CHECK IF DUMP TO PLOTTER
          KGO=0
          IF (LU.NE.LUPLTR) [
             WRITE (KONSOL,7000)
             READ (KONSOL,2000) KANSWR
             IF (KANSWR.EQ.KYES) [
                KGO=1
                LU=LUPLTR
             ]
          ]
       ] UNTIL (KGO.EQ.0)
7000   FORMAT("Replot on the plotter (Yes=f7/No=f8) ? _")
#
#.....DONE
       RETURN
       END
```



```
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE LNMRK (N,M,L)
#
#     STORE LINE INDEX IN ARRAY Z(50,48)
#       IF N=M=L=0 -> RESET THE ARRAY
#       ELSE    L>0 -> Y-DIRECTION: SET (L       )TH BIT =1
#               L<0 -> X-DIRECTION: SET (L+NLMAX)TH BIT =1
#
*CALL TEMA
*CALL TCOMMN
#
      DIMENSION    KEY(2)
      EQUIVALENCE (KEY(1),BITS)
#
      DATA NLMAX/15/
#
#.....CHECK IF RESET
      IF ( (N.EQ.0).AND.(M.EQ.0).AND.(L.EQ.0) ) [
          DO KN=1,NXMAX; DO KM=1,NYMAX [ KEY(1)=0; KEY(2)=0; Z(KN,KM)=BITS ]
      ]
#
#.....CHECK LINE INDEX
      ELSE [
          KK=IABS(L)-1
          IF ((KK.GE.0).AND.(KK.LT.NLMAX)) [
#
#.....CHECK DIRECTION
              KDIR=1; IF (L.LT.0) KDIR=2
#
#.....SET THE BIT
              NB=1; IF (KK.GT.0) DO K=1,KK; NB=NB*2
#
#.....STORE THE BIT
              BITS=Z(N,M)
              KEY(KDIR)=KEY(KDIR)+NB
              Z(N,M)=BITS
          ]
      ]
#
#.....DONE
      RETURN
      END
```

File: &PLOTC

```
*IF HP1000
EMA (XYZ,0)
*ENDIF
#
      INTEGER FUNCTION LNCHK (N,M,L)
#
#     LOAD LINE INDEX FROM ARRAY Z(50,48)
#          IF L > 0  -> Y-DIRECTION, LOAD (L       )TH BIT
#             L < 0  -> X-DIRECTION, LOAD (L+NLMAX)TH BIT
#
#
*CALL TEMA
*CALL TCOMMN
#
      DIMENSION    KEY(2)
      EQUIVALENCE (KEY(1),BITS)
#
      DATA NLMAX/15/
#
#.....CHECK LINE INDEX
      KK=IABS(L)-1
      IF ((KK.GE.0).AND.(KK.LT.NLMAX)) [
#
#.....CHECK LINE DIRECTION
           KDIR=1; IF (L.LT.0) KDIR=2
#
#.....LOAD THE FLAG
           BITS=Z(N,M)
           NB=KEY(KDIR)
#
#.....GET THE BIT
           IF (KK.GT.0)  DO K=1,KK; NB=NB/2
           LNCHK=MOD(NB,2)
      ]
#
#.....DONE
      RETURN
      END
```


File: &PLOTD

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE PLOTD (KGCB,XONY)
#
#     OUTLINE THE DEVICE
#
*CALL TEMA
*CALL TCOMMN
#
#.....PLOTTING PARAMETERS
      DIMENSION KGCB (1)
#
#.....SCALE THE PLOTTING PLANE
      TOX=AMAX1(TOX0,TOX1)
      CALL LINE (KGCB,0)
      CALL ASCAL (KGCB,XONY,-XSOURC,XMAX-XSOURC,-YMAX,TOX)
#
#.....OUTLINE THE BOUNDARY AND THE INTERFACE
      REPS=EPSI/EPSIO2
      X1=-XSOURC
      X2=XMAX-XSOURC
      Y0=-YMAX
      Y1=REPS*(DELOX(1,1)+DELOX(1,2))
      Y2=REPS*(DELOX(NXMAX,1)+DELOX(NXMAX,2))
      CALL MOVE (KGCB,X1,Y1)
      CALL DRAW (KGCB,X1,Y0)
      CALL DRAW (KGCB,X2,Y0)
      CALL DRAW (KGCB,X2,Y2)
      IF ((TOX0.NE.0.0<EXP>0).OR.(TOX1.NE.0.0<EXP>0)) DO KX=NXMAX,1,-1 [
           X=XPOS(KX)
           Y=REPS*(DELOX(KX,1)+DELOX(KX,2))
           CALL DRAW (KGCB,X,Y)
      ]
      CALL MOVE (KGCB,X1,0.0<EXP>0)
      CALL DRAW (KGCB,X2,0.0<EXP>0)
      CALL PENUP(KGCB)
#
#.....OUTLINE THE JUNCTIONS
      CALL LINE (KGCB,1)
      ND1=NDRAIN-1
      NS1=NSOURC+1
      DO N=NXMAX,ND1,-1 [
           MJ=0
           REPEAT [
                MJ=MJ+1
                TYPC=SIGN(1.0<EXP>0,(CONC(N,MJ)))
           ] UNTIL ((TYPC.EQ.TYPE).OR.(MJ.GE.NYMAX))
           XJ=XPOS(N)
           YJ=YPOS(MJ)
           CJ=CONC(N,MJ)
           IF (MJ.NE.1) YJ=YJ+DELY(MJ-1)*CJ/(CONC(N,MJ-1)-CJ)
           YJ=-YJ
           IF (N.EQ.NXMAX) CALL MOVE (KGCB,XJ,YJ)
           ELSE            CALL DRAW (KGCB,XJ,YJ)
      ]
      DO N=NS1,1,-1 [
           MJ=0
           REPEAT [
                MJ=MJ+1
                TYPC=SIGN(1.0<EXP>0,(CONC(N,MJ)))
           ] UNTIL ((TYPC.EQ.TYPE).OR.(MJ.GE.NYMAX))
           XJ=XPOS(N)
           YJ=YPOS(MJ)
           CJ=CONC(N,MJ)
           IF (MJ.NE.1)  YJ=YJ+DELY(MJ-1)*CJ/(CONC(N,MJ-1)-CJ)
           YJ=-YJ
           IF(N.EQ.NS1) CALL MOVE (KGCB,XJ,YJ)
           ELSE         CALL DRAW (KGCB,XJ,YJ)
      ]
      CALL PENUP (KGCB)
#
#.....DONE
      RETURN
      END
```

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE PLOTN
$
$     PLOT MESH
$
*CALL TEMA
*CALL TCOMMN
$
$.....PLOTTING PARAMETERS
      DIMENSION KGCB (192)
      DATA LUPLTR/9/,KESC/033B/,KBELL/007B/,KYES/2HY /,ISPEED/2/,^
           KUPMOV/-2/,KDNMOV/-1/,KMOVUP/2/,KMOVDN/1/
$
$****************     INITIALIZATION     ****************
$
$.....INITIALIZE DEVICE
      KGO=1
      LU=KONSOL
      WHILE (KGO.EQ.1) [
          IF (LU.NE.LUPLTR) [
              KTYPE=1; CALL PLOTR(KGCB,KTYPE,1,LU)
              XONY=2.0(EXP)0
              WRITE (KONSOL,3000) KESC
3000          FORMAT(R1,"*dF_")
          ]
          ELSE [
              KTYPE=2; CALL PLOTR(KGCB,KTYPE,1,LU)
              CALL PAPER(KGCB,XONY,KONSOL)
              WRITE (KONSOL,4000)
              READ  (KONSOL,*) KOLOR
4000          FORMAT("Which color? (1-4, as set up on the plotter) _")
              KOLOR=MIN0(4,MAX0(KOLOR,1))
              CALL PEN  (KGCB,KOLOR)
              CALL XMIT (KGCB)
              WRITE (LU,4010) ISPEED
4010          FORMAT("VS",I1,";VA;")
          ]
$
$.....SCALE PLOTTING PLANE AND OUTLINE DEVICE
          CALL PLOTD (KGCB,XONY)
          CALL LINE  (KGCB,0)
$
$****************     PLOTTING SECTION     ****************
          TOX=AMAX1(TOX0,TOX1)
$
$.....DRAW Y-LINES
          REPS=EPSI/EPSI02
          Y2=-YMAX
          K1=NXMAX-1
          DO KX=2,K1 [
              X=XPOS(KX)
              Y1=REPS*(DELOX(KX,1)+DELOX(KX,2))
              CALL PLOT (KGCB,X,Y1,KUPMOV)
              CALL PLOT (KGCB,X,Y2,KDNMOV)
          ]
          CALL PENUP(KGCB)
$
$.....DRAW X-LINES
          X1=XPOS(1)
          X2=XPOS(NXMAX)
          K1=NYMAX-1
          IF ((TOX0.NE.0.0(EXP)0).OR.(TOX1.NE.0.0(EXP)0)) [
              DO KX=NXMAX,1,-1 [
                  X=XPOS(KX)
                  Y=REPS*DELOX(KX,1)
                  CALL PLOT (KGCB,X,Y,KMOVDN)
              ]
              CALL PENUP (KGCB)
          ]
          DO KY=2,K1 [
              Y=-YPOS(KY)
              CALL PLOT (KGCB,X1,Y,KUPMOV)
              CALL PLOT (KGCB,X2,Y,KDNMOV)
          ]
          CALL PENUP(KGCB)
$
$.....FINISH UP
```



```
          IF (LU.EQ.LUPLTR) CALL PEN (KGCB,0)
          CALL PLOTR (KGCB ,KTYPE,0)
$
$ ....RING BELL AND TURN ON ALPHA DISPLAY
          WRITE (KONSOL,5000) KBELL,KESC
5000      FORMAT(2R1,"*dE_")
$
$.....FROM CONSOL TO PLOTTER
          KGO=0
          IF (LU.NE.LUPLTR) [
              WRITE (KONSOL,6000)
              READ  (KONSOL,2000) KANSWR
6000          FORMAT("Replot on the plotter (Yes=f7/No=f8) ? _")
2000          FORMAT(A1)
              IF (KANSWR.EQ.KYES) [
                  KGO=1
                  LU=LUPLTR
              ]
          ]
      ]
$
$.....DONE
      RETURN
      END
```

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE POSN1 (LOC,KPASS,KLITR,KMSG)
#
#     SOLVE POISSON EQUATION IN 1-D
#
*CALL TEMA
*CALL TCOMMN
#
      DIMENSION KITR8(49),NORDER(4)
#
      DATA KMAX/50/
      DATA NORDER/2Hst,2Hnd,2Hrd,2Hth/
#
#.....ASSIGN OXIDE THICKNESS
      IF ((LOC.GE.NOX0).AND.(LOC.LE.NOX1)) TOX=TOX0
      ELSE                                 TOX=TOX1
#
#.....ASSIGN GATE POTENTIAL
      IF ((LOC.GE.NGATE0).AND.(LOC.LE.NGATE1)) PHIGB=VGB
      ELSE                   IF (LOC.LE.NSOURC)  PHIGB=VSB+PHIJ
                             ELSE                PHIGB=VDB+PHIJ
#
#.....INITIALIZE BETWEEN SOURCE AND DRAIN
      IF ((LOC.GT.NSOURC).AND.(LOC.LT.NDRAIN)) [
          KTYPE=0
          IF ((DOSE(2).NE.0.0(EXP)0).AND.(LOC.GE.NIMPL0).AND.(LOC.LE.NIMPL1))
              IF (DOSE(1).NE.0.0(EXP)0) KTYPE=3
              ELSE                      KTYPE=2
          ELSE IF (DOSE(1).NE.0.0(EXP)0) KTYPE=1
          IF (KTYPE.EQ.0)                CALL  INENH (LOC,KTYPE,TOX,PHIGB)
          ELSE IF (STYPE(KTYPE).EQ.TYPE) CALL  INENH (LOC,KTYPE,TOX,PHIGB)
              ELSE                       CALL  INDEP (LOC,KTYPE,TOX,PHIGB)
      ]
#
#.....INITIALIZE SOURCE AND/OR DRAIN
      ELSE IF (ABS(DOSE(3)).LT.1.0(EXP)0) [
              DO M=1,NYMAX; POTSI(LOC,M)=0.0(EXP)0
              POTOX(LOC,2)=PHIGB
              POTOX(LOC,1)=PHIGB
              RETURN
          ]
          ELSE CALL INSAD (LOC,PHIGB)
#
#.....LOAD POTENTIAL ON TOP OF OXIDE
      POTOX(LOC,2)=PHIGB
      POTOX(LOC,1)=PHIGB                      \
      IF      (DELOX(LOC,1).EQ.0.0(EXP)0) POTSI(LOC,1)=PHIGB
      ELSE IF (DELOX(LOC,2).NE.0.0(EXP)0) POXM=PHIGB*CMOX1(LOC,1)
#
      DYM=DELY(NYMAX-1)
#
#.....INITIALIZE EVENT TABLE KITR8
      NY1=NYMAX-1
      NY2=NYMAX+1
      DO M=1,NY2; KITR8(M)=8
      IF (DELOX(LOC,2).EQ.0.0(EXP)0) KITR8(1)=9
      IF (DELOX(LOC,1).EQ.0.0(EXP)0) KITR8(2)=9
#
#.....PREPARE FOR ITERATION
      KOUNT=0
      REPEAT [
          ABSERR=0.0(EXP)0
#
#.....ON TOP OF OXIDE
          IF ((LOC.LT.NGATE0).OR.(LOC.GT.NGATE1)) POTOX(LOC,2)=POTOX(LOC,1)
#
#.....INSIDE OXIDE
          IF (DELOX(LOC,2).NE.0.0(EXP)0) [
              IF ((KOUNT.LE.KPASS).OR.(KITR8(2).NE.9)) [
                  OLDPHY=POTOX(LOC,1)
                  PHIY=POXM+POTSI(LOC,1)*CSOX1(LOC,1)
                  POTOX(LOC,1)=PHIY
                  ABSERR=ABS(OLDPHY-PHIY)
                  IF ((KOUNT.GE.KPASS).AND.(ABSERR.LE.ATOL1)) [
                      IF ((KITR8(1).EQ.1).OR.(KITR8(1).EQ.9)) KITR8(1)=9
                      ELSE                                    KITR8(1)=1
                  ]
```



```
                      ELSE                                    KITR8(1)=8
                  ]
                  ELSE [
                      IF ((KITR8(1).NE.1).AND.(KITR8(1).NE.9)) KITR8(1)=1
                      ELSE                                     KITR8(1)=9
                  ]
              ]
#
#.....IF ONE OF POINTS M-1, M, M+1 DID NOT CONVERGE IN PREVIOUS TWO CONSECUTIV
#         LOOPS -> ITERATE FOR NEW SOLUTION
          DO M=1,NY1 [
          IF (((M.EQ.1).AND.(DELOX(LOC,1).NE.0.0(EXP)0)).OR.(M.NE.1)) [
              IF ((KOUNT.LT.KPASS).OR.(KITR8(M  ).NE.9).OR.^
                  (KITR8(M+1).NE.9).OR.(KITR8(M+2).NE.9)) [
                  IF (M.EQ.1) [
                      PHIA=POTOX(LOC,1)*CMS1(LOC)+POTSI(LOC,2)*CSS1(LOC)
                      QOVRE=QONES1(LOC)
                  ]
                  ELSE [
                      PHIA=POTSI(LOC,M-1)*CM1(M-1)+POTSI(LOC,M+1)*CS1(M-1)
                      QOVRE=QONE1(M-1)
                  ]
                  PHIY=POTSI(LOC,M)
                  OLDPHY=PHIY
                  CDOPE=CONC(LOC,M)
                  CALL PHIF1 (LOC,M,1)
                  IF (KOUNT.LT.KLITR) KI=1
                  ELSE                KI=6
                  CALL ITER8 (PHIY,PHIA,QOVRE,CDOPE,ATOL1,NITR8,KI)
*IF HP1000
                  IF ((KIMSG0.EQ.1).AND.(NITR8.GE.(KI-1)).AND.(KI.NE.1)) [
                      DELPHY=PHIY-OLDPHY
                      RATIO=100.0E0*ABS(DELPHY/OLDPHY)
                      WRITE (KONSOL,3000) KOUNT,NITR8,LOC,M,OLDPHY,PHIY,^
                                          DELPHY,RATIO
3000                  FORMAT(I2,":",I2,"th iteration at (",I2,",",I2,"), ",^
                             "old=",F7.3,", new=",F7.3,", dph=",F7.3,", %=",^
                             F7.3)
                  ]
*ENDIF
                  IF (RELAX1.NE.1.0(EXP)0) PHIY=RELAX1*(PHIY-OLDPHY)+OLDPHY
                  POTSI(LOC,M)=PHIY
                  CARRIE(LOC,M)=CDOPE
                  ABSERR=AMAX1(ABSERR,ABS(OLDPHY-PHIY))
                  IF ((KOUNT.GE.KPASS).AND.(ABSERR.LE.ATOL1)) [
                      IF ((KITR8(M+1).EQ.1).OR.(KITR8(M+1).EQ.9))
                          KITR8(M+1)=9
                      ELSE KITR8(M+1)=1
                  ]
                  ELSE      KITR8(M+1)=8
              ]
          ]
          ]
#
#.....SUBSTRATE BOUNDARY
          IF ((KOUNT.LT.KPASS).OR.(KITR8(NYMAX).NE.9)) [
              OLDPHY=POTSI(LOC,NYMAX)
              IF (PHIY.GT.0.0(EXP)0) WDPL=SQRT(PHIY/ALPHB)
              ELSE                   WDPL=0.0(EXP)0
              IF (WDPL.LT.DYM) [
                  PHIY=0.0(EXP)0
                  CARRIE(LOC,NYMAX)=-CONC(LOC,NYMAX)
              ]
              ELSE [
                  PHIY=(WDPL-DYM)*(WDPL-DYM)*ALPHB
                  CARRIE(LOC,NYMAX)=0.0(EXP)0
              ]
              POTSI(LOC,NYMAX)=PHIY
              ABSERR=AMAX1(ABSERR,ABS(OLDPHY-PHIY))
              IF ((KOUNT.GE.KPASS).AND.(ABSERR.LE.ATOL1)) [
                  IF ((KITR8(NYMAX+1).EQ.1).OR.(KITR8(NYMAX+1).EQ.9))
                      KITR8(NYMAX+1)=9
                  ELSE KITR8(NYMAX+1)=1
              ]
              ELSE      KITR8(NYMAX+1)=8
          ]
          ELSE [
              IF ((KITR8(NYMAX+1).NE.9).AND.(KITR8(NYMAX+1).NE.1))
                  KITR8(NYMAX+1)=1
              ELSE KITR8(NYMAX+1)=9
```

File: &POSHI

```
         ]
         KOUNT=KOUNT+1
      ] UNTIL ((ABSERR.LE.ATOL1).OR.(KOUNT.GE.KMAX))
#
#.....WRITE OUT PROPER MESSAGE
      IF (KMSG.NE.0) [
      LDGIT=MOD(KOUNT,10)
      IF (LDGIT.EQ.1) WRITE (KONSOL,1000) LOC,KOUNT,NORDER(1),ABSERR
      ELSE [
           IF (LDGIT.EQ.2) WRITE (KONSOL,1000) LOC,KOUNT,NORDER(2),ABSERR
           ELSE [
                IF (LDGIT.EQ.3) WRITE (KONSOL,1000) LOC,KOUNT,NORDER(3),ABSERR
                ELSE            WRITE (KONSOL,1000) LOC,KOUNT,NORDER(4),ABSERR
           ]
      ]
1000  FORMAT ("Initialize column",I3,":",^
              "Converged at",I3,A2," iteration, maximum deviation =",G10.3)
      ]
#
#.....DONE
      RETURN
      END
```


File: &POSSN

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE POSSN
#
#     SOLVE 2-D POISSON DIFFERENCE EQUATION
#
*CALL TEMA
*CALL TCOMMN
#
      DIMENSION KITRB(50,49),NORDER(4)
#
#.....EQUIVALENCE EMA ARRAYS INTO ONE-DIMENSIONAL ARRAYS
      DIMENSION ODCONC(1),ODPOTS(1),ODCNOR(1),ODCSOU(1),ODCEAS(1),ODCWES(1),^
                ODQONE(1),ODCARR(1)
      EQUIVALENCE (ODCONC(1),CONC(1,1)),  (ODPOTS(1),POTSI(1,1)),^
                  (ODCNOR(1),CNORTH(1,1)),(ODCSOU(1),CSOUTH(1,1)),^
                  (ODCEAS(1),CEAST(1,1)), (ODCWES(1),CWEST(1,1)),^
                  (ODQONE(1),QONE(1,1)),  (ODCARR(1),CARRIE(1,1))
#
      DATA NORDER/2Hst,2Hnd,2Hrd,2Hth/, KYES/2HY /
*IF HP1000
      DATA KESC/033B/, KCR/015B/
*ENDIF
#
      KPASS=1
#
#.....CHECK IF ENVOKE LOCAL ITERATION
      KLITR=KMAX2+1
      IF (RELAX2.EQ.1) [
         WRITE (KONSOL,1050)
1050     FORMAT("Envoke local iteration (Yes=f7/No=f8) ? _")
         READ  (KEYBRD,2000) KANSWR
*IF BATCH
         WRITE (KONSOL,2000) KANSWR
*ENDIF
         IF (KANSWR.EQ.KYES) [
            WRITE (KONSOL,1060)
1060        FORMAT("Start from ? iteration ? _")
            READ  (KEYBRD,*) KLITR
*IF BATCH
            WRITE (KONSOL,1061) KLITR
1061        FORMAT(I4)
*ENDIF
            WRITE (KONSOL,1070)
1070        FORMAT("Convergence information per grid (Yes=f7/No=f8) ? _")
            READ  (KEYBRD,2000) KANSWR
*IF BATCH
            WRITE (KONSOL,2000) KANSWR
*ENDIF
            IF (KANSWR.EQ.KYES) KIMSG1=1
            ELSE                KIMSG1=0
         ]
      ]
2000  FORMAT(A1)
#
#.....INITIALIZE EVENT TABLE
      NY1=NYMAX+1
      DO N=1,NXMAX [
         DO M=1,NY1; KITRB(N,M)=0
         IF (DELOX(N,2).EQ.0.0(EXP)0) KITRB(N,1)=9
         IF (DELOX(N,1).EQ.0.0(EXP)0) KITRB(N,2)=9
      ]
#
#.....PREPARE FOR ITERATION
      NX2=NXMAX-1
      NY2=NYMAX-1
      DYM=DELY(NY2)
      JUNK=IFBRK(0)                              # reset BReak flag
      TOTSK=0.0E0                                # reset SKIP count
      PRCNT0=100.0E0/FLOAT(NXMAX*NYMAX)
      KOUNT=0                                    # reset loop count
      KGO=1                                      # set GO flag
      NC0=(NSOURC+NDRAIN)/2                      # set channel center indice
      NC0=1
#
#.....ALTERNATE X-LOOP DIRECTION EVERY OTHER ONE
      REPEAT [
         IF (IFBRK(0).EQ.0) [
```

```
          CALL PHIF1 (NCO,NCO,0)        # DEFINE X'TION RGN OF Q-F-LEVEL
          IF ((KOUNT-MOD(KOUNT,2)).NE.KOUNT) [ N1=2;   N2=NX2;  ND=1   ]
          ELSE                                [ N1=NX2; N2=2;    ND=-1  ]
#
#.....CLEAR CONVERGENCE MESSAGE
          NOLD=NMAX;        MOLD=MMAX
          NMAX=0;           MMAX=0
          EROLD=ABSERR
          ABSERR=0.0E0
          NSKIP=0
          KOUNT=KOUNT+1
#
#.....INSIDE OXIDE
          N=N1-ND
          NSGN=ISIGN(1,N2-N)
          NSGNO=ISIGN(1,N2-N1)
          WHILE ((N2.NE.N).AND.(NSGNO.EQ.NSGN)) [
              N=N+ND
              NSGN=ISIGN(1,N2-N)
              IF ((N.LT.NGATE0).OR.(N.GT.NGATE1))
                  POTOX(N,2)=POTOX(N,1)*CSOX(N,2)+POTOX(N-1,2)*CWOX(N,2)^
                                               +POTOX(N+1,2)*CEOX(N,2)
              IF (DELOX(N,2).NE.0.0(EXP)0) [
                  IF ((KOUNT.LT.KPASS).OR.(KITR8(N,2) NE.9)) [
                      OLDPHI=POTOX(N,1)
                      PHI=POTOX(N   ,2)*CNOX(N,1)+POTSI(N   ,1)*CSOX(N,1)^
                          +POTOX(N-1,1)*CWOX(N,1)+POTOX(N+1,1)*CEOX(N,1)
                      POTOX(N,1)=PHI
                      ERR=ABS(OLDPHI-PHI)
                      IF ((KOUNT.GE.KPASS).AND.(ERR.LE.ATOL2)) [
                          IF ((KITR8(N,1).EQ.1).OR.(KITR8(N,1).EQ.9))
                                KITR8(N,1)=9
                          ELSE KITR8(N,1)=1
                      ]
                      ELSE      KITR8(N,1)=8
                      IF (ABSERR.LT.ERR) [  ABSERR=ERR; NMAX=N; MMAX=-1  ]
                  ]
                  ELSE [
                      IF ((KITR8(N,1).NE.1).AND.(KITR8(N,1).NE.9))
                            KITR8(N,1)=1
                      ELSE KITR8(N,1)=9
                  ]
              ]
#
#.....IN SILICON
          DO M=1,NY2 [
          IF (((M.EQ.1).AND.(DELOX(N,1)\NE.0.0(EXP)0)).OR.(M.NE.1)) [
              IF (KOUNT.LT.KPASS)        GO TO 800
              IF (KITR8(N,M).NE.9)       GO TO 800
              IF (KITR8(N,M+1).NE.9)     GO TO 800
              IF (KITR8(N,M+2).NE.9)     GO TO 800
              IF (KITR8(N-1,M+1).NE.9)   GO TO 800
              IF (KITR8(N+1,M+1).NE.9) [
800               CONTINUE
                  IF (M.EQ.1) PHIA=POTOX(N,1)
                  ELSE        PHIA=POTSI(N,M-1)
#
# PERFORM SUBSCRIPT CALCULATION EXACTLY ONCE: array(I,J) -> array(K)
#         ONE-DIMENSIONAL OFFSET VALUE : K=(J-1)*50+I
#
                  NM=(M-1)*50+N
                  PHIA=PHIA*ODCNOR(NM)+POTSI(N,M+1)*ODCSOU(NM)+^
                        POTSI(N-1,M)*ODCWES(NM)+POTSI(N+1,M)*ODCEAS(NM)
                  CDOPE=ODCONC(NM)
                  PHI=ODPOTS(NM)
                  OLDPHI=PHI
                  CALL PHIF1 (N,M,2)
                  IF (KOUNT.LT.KLITR) KI=1
                  ELSE                KI=6
                  CALL ITER8(PHI,PHIA,(ODQONE(NM)),CDOPE,ATOL2,NITR8,^
                             KI)
*IF IBATCH
                  IF ((KI.NE.1).AND.(NITR8.GE.(KI-1)).AND.^
                      (KIMSG1.NE.0)) [
                      DELPHI=PHI-OLDPHI
                      RATIO=100.0(EXP)0*ABS(DELPHI/OLDPHI)
                      WRITE (KONSOL,2010) KOUNT,NITR8,N,M,OLDPHI,PHI,^
                                          DELPHI,RATIO
2010                  FORMAT(I3,":",I2,"th iteration at (",I2,",",I2,^
                             "), old=",F7.3,", new=",F7.3,", ",^
```



```
                      .     "dph=",F7.3,", %=",F7.3)
                  ]
*ENDIF
                  IF (RELAX2 NE.1) PHI=RELAX2*(PHI-OLDPHI)+OLDPHI
                  ODPOTS(NM)=PHI
                  ODCARR(NM)=CDOPE
                  ERR=ABS(OLDPHI-PHI)
                  IF ((KOUNT.GE.KPASS).AND.(ERR.LE.ATOL2)) [
                      IF ((KITR8(N,M+1).EQ.1).OR.(KITR8(N,M+1).EQ.9))
                            KITR8(N,M+1)=9
                      ELSE KITR8(N,M+1)=1
                  ]
                  ELSE      KITR8(N,M+1)=8
                  IF (ABSERR.LT.ERR) [  ABSERR=ERR; NMAX=N; MMAX=M  ]
              ]
              ELSE NSKIP=NSKIP+1
          ]
          ]
#
#.....THE  LAST POINT AT THE SUBSTRATE BOUNDARY
          IF ((KOUNT.LT.KPASS).OR.(KITR8(N,NYMAX).NE.9)) [
              OLDPHI=POTSI(N,NYMAX)
              WDPL=SQRT(AMAX1((POTSI(N,NYMAX-1)),0.0(EXP)0)/ALPHB)
              IF (WDPL.LT.DYM) [
                  PHI=0.0(EXP)0
                  CARRIE(N,NYMAX)=-CONC(N,NYMAX)
              ]
              ELSE [
                  PHI=(WDPL-DYM)*(WDPL-DYM)*ALPHB
                  CARRIE(N,NYMAX)=0.0(EXP)0
              ]
              POTSI(N,NYMAX)=PHI
              ERR=ABS(OLDPHI-PHI)
              IF ((KOUNT.GE.KPASS).AND.(ERR.LE.ATOL2)) [
                  IF ((KITR8(N,NYMAX+1).EQ.1).OR.^
                      (KITR8(N,NYMAX+1).EQ.9)) KITR8(N,NYMAX+1)=9
                  ELSE                         KITR8(N,NYMAX+1)=8
              ]
              IF (ABSERR.LT.ERR) [ ABSERR=ERR; NMAX=N; MMAX=NYMAX ]
          ]
          ELSE [
              IF ((KITR8(N,NYMAX+1).NE.9).AND.(KITR8(N,NYMAX+1).NE.1))
                    KITR8(N,NYMAX+1)=1
              ELSE KITR8(N,NYMAX+1)=9
          ]
          ]
#
#.....IF NO SOURCE/DRAIN, FLOAT THE BOUNDARY CONDITION (CASE OF NARROW CHANNEL)
          IF ((ND.LT.0).AND.((NSOURC.LT.1).OR.(DOSE(3).EQ.0.0(EXP)0))) [
              POTOX(1,1)=POTOX(2,1)
              POTOX(1,2)=POTOX(2,2)
              DO M=1,NYMAX; POTSI(1,M)=POTSI(2,M)
          ]
          IF ((ND.GT.0).AND.((NDRAIN.GT.NXMAX).OR.(DOSE(3).EQ.0.0(EXP)0)))[
              POTOX(NXMAX,1)=POTOX(NXMAX-1,1)
              POTOX(NXMAX,2)=POTOX(NXMAX-1,2)
              DO M=1,NYMAX; POTSI(NXMAX,M)=POTSI(NXMAX-1,M)
          ]
#
#.....PRINT  CONVERGENCE MESSAGE
          PERCNT=PRCNTO*FLOAT(NSKIP)
          TOTSK=TOTSK+PERCNT
*IF IBATCH&HP2648A
          LDGIT=MOD(KOUNT,10)
          IF (LDGIT.EQ.1)
              IF (KIMSG2.EQ.0) WRITE (KONSOL,3001) KOUNT,NORDER(1),^
                                      ABSERR,NMAX,MMAX,PERCNT,KCR
              ELSE             WRITE (KONSOL,3000) KOUNT,NORDER(1),^
                                      ABSERR,NMAX,MMAX,PERCNT
          ELSE [
              IF (LDGIT.EQ.2)
                  IF (KIMSG2.EQ.0) WRITE (KONSOL,3001) KOUNT,NORDER(2),^
                                          ABSERR,NMAX,MMAX,PERCNT,KCR
                  ELSE             WRITE (KONSOL,3000) KOUNT,NORDER(2),^
                                          ABSERR,NMAX,MMAX,PERCNT
              ELSE [
                  IF (LDGIT.EQ.3)
                      IF (KIMSG2.EQ.0) WRITE (KONSOL,3001) KOUNT,^
                                          NORDER(3),ABSERR,NMAX,MMAX,^
                                          PERCNT,KCR
```

File: &POSSN

```
                    ELSE                  WRITE (KONSOL,3000) KOUNT,^
                                                 NORDER(3),ABSERR,NMAX,MMAX,^
                                                 PERCNT
                ELSE
                    IF (KIMSG2.EQ.0) WRITE (KONSOL,3001) KOUNT,^
                                                 NORDER(4),ABSERR,NMAX,MMAX,^
                                                 PERCNT,KCR
                    ELSE                  WRITE (KONSOL,3000) KOUNT,^
                                                 NORDER(4),ABSERR,NMAX,MMAX,^
                                                 PERCNT
                  ]
              ]
3000          FORMAT (I3,A2," loop! max deviation =",1PG9.3,", at (",I2,",",^
                      I2,"), by-pass",0PF6.2,"% !")
3001          FORMAT (I3,A2," loop! max deviation =",1PG9.3,", at (",^
                      I2,",",I2,"), by-pass",0PF6.2,"% !",R1,"_")
*ENDIF
            ]
            ELSE KGO=0
        ] UNTIL ( (ABSERR.LE.ATOL2).OR.(KOUNT.GE.KMAX2).OR.(KGO.EQ.0) )
#
#.....DONE
      TOTSK=TOTSK/(KOUNT-1)
      KNTX=KOUNT-1
      WRITE (KONSOL,3010) KNTX,EROLD,NOLD,MOLD,TOTSK
3010  FORMAT("** 2-D iteration stops at loop ",I3,"!",14X^
             /,7X,"last max deviation =",1PG9.3,", at (",I2,",",I2,"), ",^
                 "by pass",0PF6.2,"% ave per loop")
      RETURN
      END
```


File: &RDFDP

```
*CALL TWDEFN
      SUBROUTINE RDFDP
#
#     REDEFINE DOPING PROFILE
#
*CALL TCOMMN
#
      DIMENSION KDOPE(6)
#
      DATA KYES/2HY /, UM/1.0<EXP>-4/, SECND/60 0<EXP>0/
      DATA KDOPE/2HB ,2HA ,2HP ,2HS ,2H+ ,2H- /, KDSUP/2HF /
#
#.....CHANGE ALL
      WRITE (KONSOL,1000)
1000  FORMAT("Change all profile parameters (Yes=f7/No=f8) ? _")
      READ  (KEYBRD,2000) KANSUR
*IF BATCH
      WRITE (KONSOL,2000) KANSUR
*ENDIF
2000  FORMAT(A1)
      IF (KANSUR.EQ.KYES) CALL REDDP
#
#.....CHANGE PART OF THEM
      ELSE [
          WRITE (KONSOL,1010)
1010      FORMAT(6X,"substrate dopant: B,As,Ph,Sb,+(n-type),-(p-type),",^
                                          "F(Suprem)... 1"^
                /6X,"substrate doping concentration (*1E15 cm-3)........... 2"^
                /6X,"overall implant dopant: B, As, Ph, Sb, +, -............ 3"^
                /6X,"                  range(um), stndev(um) and dose(cm-2)... 4"^
                /6X,"                  diffusion constant (cm2/sec)........... 5"^
                /6X,"                  drive in temperature (oC).............. 6"^
                /6X,"                  drive in time (min).................... 7"^
                /6X,"localized implant location: from ? to ? (um)........... 8"^
                /6X,"                    dopant: B, As, Ph, Sb, +, -.......... 9"^
                /6X,"                    range(um), stndev(um) and dose (cm-2) 10"^
                /6X,"                    diffusion constant (cm2/sec)......... 11"^
                /6X,"                    drive in temperature (oC)............ 12"^
                /6X,"                    drive in time (min).................. 13"^
                /6X,"source/drain implant dopant: B, As, Ph, Sb, +, -........ 14"^
                /6X,"                       rng(um), stndev(um) and dose(cm-2) 15"^
                /6X,"                       diffusion constant (cm2/sec)...... 16"^
                /6X,"                       drive in temperature (oC)......... 17"^
                /6X,"                       drive in time (min)............... 18")
#
#.....READ INDEX
          REPEAT [
              WRITE (KONSOL,1020)
1020          FORMAT("Which one! (Enter index) ? _")
              READ  (KEYBRD,*) KPARM
*IF BATCH
              WRITE (KONSOL,1021) KPARM
1021          FORMAT(I3)
*ENDIF
#
#.....GET IMPLANT FLAG
              IF             (KPARM.LE.2)  KKIMP=0
              ELSE IF        (KPARM.LE.7)  KKIMP=1
                  ELSE IF (KPARM.LE.13) KKIMP=2
                      ELSE              KKIMP=3
#
#.....GET PARAMETERS
              WRITE (KONSOL,1030)
1030          FORMAT("Value(s) ? _")
#
#.....READ SYMBLE
              IF ((KPARM.EQ.1).OR.(KPARM.EQ.3).OR.^
                  (KPARM.EQ.9).OR.(KPARM.EQ.14)) [
                  READ  (KEYBRD,2000) KANSUR
*IF BATCH
                  WRITE (KONSOL.2000) KANSUR
*ENDIF
#
#.....DOPING SPECIES
                  K=1; WHILE ((KANSUR.NE.KDOPE(K)).AND.(K.LT.6)) K=K+1
                  IF ((K.EQ.1).OR.(K.EQ.6)) DTYPE=-1 0<EXP>0
                  ELSE                      DTYPE= 1.0<EXP>0
                  IF (KPARM.EQ.1) [
                      IF (KANSUR.EQ.KDSUP) KSUPRM=1
                      ELSE [               KSUPRM=0; TYPE=DTYPE ]
```

```
                ]
                ELSE DOPE(KKIMP)=K
            ]
#
#.....READ VALUES
            ELSE [
                READ (KEYBRD,*) ANS0,ANS1,ANS2
*IF BATCH
                IF ((KPARM.EQ.4).OR.(KPARM.EQ.10).OR.(KPARM.EQ.15))
                                        WRITE (KONSOL,2001) ANS0,ANS1,ANS2
                ELSE IF (KPARM.EQ.8) WRITE (KONSOL,2001) ANS0,ANS1
                    ELSE                WRITE (KONSOL,2001) ANS0
2001            FORMAT(1P3G13.3)
*ENDIF
#
#.... SUBSTRATE CONCENCTRAION
                IF (KPARM.EQ.2) CSUB=ANS0*1.0(EXP)15
#
#.....IMPLANT WINDOW
                ELSE IF (KPARM.EQ.8) [
                        XIMPL0=ANS0*UM
                        XIMPL1=ANS1*UM
                        IF (XIMPL0.GT.XIMPL1) [
                            T=XIMPL0; XIMPL0=XIMPL1; XIMPL1=T
                        ]
                    ]
#
#.....IMPLANT PARAMETERS
                    ELSE [
                        IF (KPARM.LT.8) KK=KPARM
                        ELSE            KK=KPARM-1
                        KK=(KK-2)/5
                        IF (KK.EQ.2) [
                            RANGE(KKIMP)=ANS0*UM
                            STNDV(KKIMP)=AMAX1(ANS1*UM,0.0(EXP)0)
                            DOSE(KKIMP)=SIGN(ANS2,DTYPE)
                        ]
                        IF (KK.EQ.3) DCOEF(KKIMP)=AMAX1(ANS0,0.0(EXP)0)
                        IF (KK.EQ.4) TEMP(KKIMP)=ANS0
                        IF (KK.EQ.5) DRVIN(KKIMP)=AMAX1(ANS0*SECND,0.0(EXP0)
                    ]
            ]
#
#.....ANY MORE CHANGES?
            WRITE (KONSOL,1040)
1040        FORMAT("More changes (Yes=f7/No=f8) ? _")
            READ  (KEYBRD,2000) KANSUR
*IF BATCH
            WRITE (KONSOL,2000) KANSUR
*ENDIF
            IF (KANSUR.EQ.KYES) KGO=1
            ELSE                KGO=0
        ] UNTIL (KGO.EQ.0)
#
#.....PROCESS PARAMETERS OF STEP APPROXIMATION
        DO KINPL=1,3; IF (ABS(DOSE(KINPL)).GT.1.0(EXP)0) CALL ASTEP(KINPL)
      ]
#
#.....DONE
      RETURN
      END
```



```
*CALL TWDEFN
      SUBROUTINE RDFGM
#
#     REDEFINE DEVICE GEOMETRY
#
*CALL TCOMMN
#
      DATA KYES/2HY /, UM/1.0(EXP)-4/
#
#.....CHANGE ALL
      WRITE (KONSOL,1000)
1000  FORMAT("Change all geometry parameters (Yes=f7/No=f6) ? _")
      READ  (KEYBRD,2000) KANSUR
*IF BATCH
      WRITE (KONSOL,2000) KANSUR
*ENDIF
2000  FORMAT(A1)
      IF (KANSUR.EQ.KYES) CALL REDGM
#
#.....CHANGE PART OF THEM
      ELSE [
          WRITE (KONSOL,1010)
1010      FORMAT(6X,"drawn channel length   (um)............... 1"^
                /6X,"lateral span of source (um)............... 2"^
                /6X,"lateral span of drain  (um)............... 3"^
                /6X,"oxide thickness: thin? and thick? (um).... 4"^
                /6X,"thin oxide location: from ? to ?  (um).... 5"^
                /6X,"lateral span of oxide ramp (um)........... 6"^
                /6X,"drawn gate location: from ? to ? (um)..... 7"^
                /6X,"depth of the simulated structure (um)..... 8")
#
#.....READ INDEX AND VALUE(S)
          REPEAT [
              WRITE (KONSOL,1020)
1020          FORMAT("Which one: (Enter index) ? _")
              READ  (KEYBRD,*) KPARM
*IF BATCH
              WRITE (KONSOL,1021) KPARM
1021          FORMAT(I3)
*ENDIF
              WRITE (KONSOL,1030)
1030          FORMAT("Value(s) ? _")
              READ  (KEYBRD,*) ANS0,ANS1
#
#.....SCALE AND LOAD NEW VALUES
              PARM0=AMAX1(ABS(ANS0)*UM,0.0(EXP)0)
              PARM1=AMAX1(ABS(ANS1)*UM,0.0(EXP)0)
              IF (KPARM.EQ.1) XCHANL=PARM0
              IF (KPARM.EQ.2) XSOURC=PARM0
              IF (KPARM.EQ.3) XDRAIN=PARM0
              IF (KPARM.EQ.4) [ TOX0=PARM0; TOX1=PARM1 ]
              IF (KPARM.EQ.5) [ XOX0=PARM0; XOX1=PARM1
                  IF (XOX0.GT.XOX1) [ T=XOX0; XOX0=XOX1; XOX1=T ]
              ]
              IF (KPARM.EQ.6) XOXR=PARM0
              IF (KPARM.EQ.7) [ XGATE0=PARM0; XGATE1=PARM1
                  IF (XGATE0.GT.XGATE1) [ T=XGATE0; XGATE0=XGATE1; XGATE1=T ]
              ]
              IF (KPARM.EQ.8) YMAX=PARM0
#
#.....ANY MORE ?
              WRITE (KONSOL,1040)
1040          FORMAT("More changes (Yes=f7/No=f8) ? _")
              READ  (KEYBRD,2000) KANSUR
*IF BATCH
              WRITE (KONSOL,2000) KANSUR
*ENDIF
              IF (KANSUR.EQ.KYES) KGO=1
              ELSE                KGO=0
          ] UNTIL (KGO.EQ.0)
#
#.....UPDATE XMAX
          XMAX=XCHANL+XSOURC+XDRAIN
      ]
#
#.....DONE
      RETURN
      END
```

```
*CALL TUDEFN
      SUBROUTINE REDDP
#
#     READ IN DOPING PARAMETERS FROM THE CONSOLE
#
*CALL TCOMMM
#
      DIMENSION MESSG0(5,2),MESSG1(3,2),MESSG2(4,3),KDOPE(6)
      DATA KYES/2HY /, UM/1.0<EXP>-4/
      DATA LNMSG0/5/,^
           MESSG0/2HSu,2Hbs,2Htr,2Hat,2He , 2HIm,2Hpl,2Han,2Ht ,2H  /
      DATA LNMSG1/3/,^
           MESSG1/2HAm,2Hy ,2H  ,2HCh,2Han,2Hge/
      DATA LNMSG2/4/,^
           MESSG2/2Hov,2Her,2Hal,2Hl ,2Hle,2Hca,2Hl',2Hd ,2Hsr,2Hc/,2Hdr,2Hn /
      DATA KDOPE  /2HB ,2HA ,2HP ,2HS ,2H+ ,2H- /
#
#.....DEFINE SOFT KEYS 1-6
*IF HP2640A
      CALL SKEYP
*ENDIF
#
#.....GET THE SUBSTRATE CONCENTRATION
      WRITE (KONSOL,1000) (MESSG0(K,1),K=1,LNMSG0)
1000  FORMAT(5A2," dopant (B=f1,As=f2,Ph=f3,Sb=f4,",^
                     "+(N-type)=f5,-(P-type)=f6) ? _")
      READ  (KEYBRD,2000) KANSUR
*IF BATCH
      WRITE (KONSOL,2000) KANSUR
*ENDIF
2000  FORMAT(A1)
      K=1; WHILE ((KANSUR.NE.KDOPE(K)).AND.(K.LT.6)) K=K+1
      IF ((K.EQ.1).OR.(K.EQ.6)) TYPE=-1.0<EXP>0
      ELSE                      TYPE= 1.0<EXP>0
      WRITE (KONSOL,1010)
1010  FORMAT("Substrate concentration (in unit of 1E15 cm-3) ? _")
      READ  (KEYBRD,*) CSUB
*IF BATCH
      WRITE (KONSOL,1011) CSUB
1011  FORMAT(1PG10.3)
*ENDIF
      CSUB=SIGN(CSUB*1.0<EXP>15,TYPE)
#
#.....GET PARAMETERS OF INPLANTATIONS
      IF (KLOOP.EQ.1) KMSG=1
      ELSE            KMSG=2
      DO KIMPL=1,3 [
          KANSUR=KYES
          WRITE (KONSOL,1020) (MESSG1(K,KMSG), K=1,LNMSG1),^
                              (MESSG2(K,KIMPL),K=1,LNMSG2)
1020      FORMAT(3A2,1X,4A2," implant (Yes=f7/No=f8) ? _")
          READ  (KEYBRD,2000) KANSUR
*IF BATCH
          WRITE (KONSOL,2000) KANSUR
*ENDIF
          IF (KANSUR.EQ.KYES) [
              IF (KIMPL.EQ.2) [
                  WRITE (KONSOL,1025)
1025              FORMAT ("Localized implant: from () to () (um) ? _")
                  READ  (KEYBRD,*) XIMPL0,XIMPL1
*IF BATCH
                  WRITE (KONSOL,1026) XIMPL0,XIMPL1
1026              FORMAT(1P2G13.3)
*ENDIF
              ]
              WRITE (KONSOL,1030)
1030          FORMAT("Implant dopant (B=f1,As=f2,Ph=f3,Sb=f4,",^
                                       "+(N-type)=f5,-(P-type)=f6) ? _")
              READ  (KEYBRD,2000) KANSUR
*IF BATCH
              WRITE (KONSOL,2000) KANSUR
*ENDIF
              K=1; WHILE ((KANSUR.NE.KDOPE(K)).AND.(K.LT.6)) K=K+1
              DOPE(KIMPL)=K
              WRITE (KONSOL,1040)
1040          FORMAT("Implant parameters: ",^
                  "RANGE(um) ? STNDV(um) ? DOSE(cm-2) ? _")
              READ  (KEYBRD,*) RANGE(KIMPL),STNDV(KIMPL),DOSE(KIMPL)
*IF BATCH
              WRITE (KONSOL,1041) RANGE(KIMPL),STNDV(KIMPL),DOSE(KIMPL)
```



```
1041          FORMAT(1P3G13.3)
*ENDIF
              IF (K.GE.5) [
                  WRITE (KONSOL,1050)
1050              FORMAT("Diffusion coefficient at ",^
                         "drive-in temperature (cm2/sec) ? _")
                  READ  (KEYBRD,*) DCOEF(KIMPL)
*IF BATCH
                  WRITE (KONSOL,1051) DCOEF(KIMPL)
1051              FORMAT(1PG10.3)
*ENDIF
              ]
              ELSE [
                  WRITE (KONSOL,1055)
1055              FORMAT("Drive-in temperature (oC) ? _")
                  READ  (KEYBRD,*) TEMP(KIMPL)
*IF BATCH
                  WRITE (KONSOL,1056) TEMP(KIMPL)
1056              FORMAT(1PG10 3)
*ENDIF
              ]
              WRITE (KONSOL,1060)
1060          FORMAT("Drive-in time (minutes) ? _")
              READ  (KEYBRD,*) DRVIN(KIMPL)
*IF BATCH
              WRITE (KONSOL,1061) DRVIN(KIMPL)
1061          FORMAT(1PG10.3)
*ENDIF
#
#.....SCALE AND CHECK INPUT PARAMETERS
              CALL CHKDP (KIMPL)
          ]
          ELSE [
              DOPE(KIMPL)=0.0<EXP>0
              RANGE(KIMPL)=0.0<EXP>0
              STNDV(KIMPL)=0.0<EXP>0
              DOSE(KIMPL)=0.0<EXP>0
              DCOEF(KIMPL)=0.0<EXP>0
              TEMP(KIMPL)=0.0<EXP>0
              DRVIN(KIMPL)=0.0<EXP>0
              XJCT(KIMPL)=0.0<EXP>0
              CSTEP(KIMPL)=0.0<EXP>0
              IF (KIMPL.EQ.2) [ XIMPL0=-XSOURC-XSOURC; XIMPL1=XIMPL0 ]
          ]
      ]
#
#.....DONE
      RETURN
      END
```

```
*CALL TWDEFN
      SUBROUTINE REDF1
#
#     READ INPUT PARAMETERS FROM A FILE
#
*CALL TCOMMN
#
      DIMENSION KDOPE(6),NAMINP(4,3)
#
#.....FILE MANAGEMENT PARAMETERS
*IF HP1000
      DIMENSION NAMBUF(10),NSTRNG(20),NAMFIL(3),^
                KDCB(144), KSIZE(2),  KBUF(40)
#
      EQUIVALENCE (NAMBUF(1),NAMFIL(1)),(NAMBUF(5),NSECU),(NAMBUF(6),NCR)
#
      DATA KSIZE/20,80/,KTYPE/3/,KDCBS/128/,^
           LRECL/40/,   NSECU/0/,NCR/2HXX/
*ENDIF
#
      DATA KDOPE/2HB ,2HA ,2HP ,2HS ,2H+ ,2H- /, KDSUP/2HF /,^
           NAMINP/2Hev,2Her,2Hal,2Hl ,^
                  2Hlo,2Hca,2Hl',2Hd ,^
                  2Hsr,2Hc/,2Hdr,2Hn /
      DATA NAMSTR/2H**/
*IF HP1000
      DATA KESC /033B/
*ENDIF
#
#.....GET THE DATA FILE NAME
*IF HP1000
      KOPEN=1
      WHILE (KOPEN.EQ.1) [
          WRITE (KONSOL,1000)
1000      FORMAT("Input data file name? _")
          READ (KEYBRD,1010) NSTRNG
          NPOS=1; CALL NAMR (NAMBUF,NSTRNG,40,NPOS)
          DO K=1,10; INPFIL(K)=NAMBUF(K)
#
#.....OPEN THE FILE
          CALL OPEN  (KDCB,KERR,NAMFIL,0,NSECU,NCR,KDCBS)
          KOPEN=0
          IF (KERR.LT.0) CALL TFERR(1,KERR,NAMBUF,KOPEN)
        ]
*ELSE
      READ (KEYBRD,1010) NSTRNG
      KERR=0
*ENDIF
*IF BATCH
      WRITE (KONSOL,1000)
      WRITE (KONSOL,1010) NSTRNG
*ENDIF
1010  FORMAT(20A2)
#
#.....RESET ERROR FLAG AND SKIP TITLE
      IF (KERR.GE.0) [
          KEXIT=0
*IF HP1000
          CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
*ELSE
          READ  (LURD1,1010) IDUMMY; IF (EOF(LURD1).NE.0.0E0) GO TO 900
*ENDIF
#
#.....GEOMETRY PARAMETERS
*IF HP1000
          CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
          CALL CODE  (KEND*2); READ (KBUF,*) XCHANL
*ELSE
          READ (LURD1,*) XCHANL; IF (EOF(LURD1).NE.0.0E0) GO TO 900
*ENDIF
#
#.....CHECK IF FATAL ERROR
          IF (XCHANL.LE.0.0) GO TO 800
#
#.....CONTINUE READING
*IF HP1000
          CALL READF (KDCB,KERR,KBUF,LRECL,KEND) ; IF (KEND.EQ.-1) GO TO 900
          CALL CODE  (KEND*2); READ (KBUF,*) XSOURC
          CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
          CALL CODE  (KEND*2); READ (KBUF,*) XDRAIN
```


File: &REDF1

```
          CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
          CALL CODE  (KEND*2); READ (KBUF,*) TOXO,TOX1
          CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
          CALL CODE  (KEND*2); READ (KBUF,*) XOXO,XOX1
          CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
          CALL CODE  (KEND*2); READ (KBUF,*) XOXR
          CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
          CALL CODE  (KEND*2); READ (KBUF,*) XGATEO,XGATE1
          CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
          CALL CODE  (KEND*2); READ (KBUF,*) YMAX
*ELSE
          READ (LURD1,*) XSOURC;          IF (EOF(LURD1).NE.0.0E0) GO TO 900
          READ (LURD1,*) XDRAIN;          IF (EOF(LURD1).NE.0.0E0) GO TO 900
          READ (LURD1,*) TOXO,TOX1;       IF (EOF(LURD1).NE.0.0E0) GO TO 900
          READ (LURD1,*) XOXO,XOX1;       IF (EOF(LURD1).NE.0.0E0) GO TO 900
          READ (LURD1,*) XOXR;            IF (EOF(LURD1).NE.0.0E0) GO TO 900
          READ (LURD1,*) XGATEO,XGATE1;   IF (EOF(LURD1).NE.0.0E0) GO TO 900
          READ (LURD1,*) YMAX;            IF (EOF(LURD1).NE.0.0E0) GO TO 900
*ENDIF
#
#.....PRINT INPUT SUMMARY
          WRITE (KONSOL,3010) (NAMSTR,K=1,28),XCHANL,XSOURC,XDRAIN,^
                              TOXO,TOX1,XOXO,XOX1,XOXR,XGATEO,XGATE1,YMAX
3010      FORMAT(14A2,5X,"INPUT SUMMARY",5X,14A2^
                /,5X,"Dawn  channel   length ",F7.3,"um"^
                /,5X,"Lateral span of source ",F7.3,"um"^
                /,5X,"Lateral span of  drain ",F7.3,"um"^
                /,5X,"Oxide thickness: thin/thick: ",F7.3,", ",F7.3,"um"^
                /,5X,"Thin oxide  loc:  from/to:    ",F7.3,", ",F7.3,"um"^
                /,5X,"Lateral  oxide    ramp ",F7.3,"um"^
                /,5X,"Drawn gate  loc:  from/to:    ",F7.3,", ",F7.3,"um"^
                /,5X,"Structure depth         ",F7.3,"um")
#
#.....SCALE AND LIMIT GEOMETRY PARAMETERS
          CALL CHKGM
#
#.....READ SUPREM FLAG
*IF HP1000
          CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
          CALL CODE  (KEND*2); READ (KBUF,2000) KB
*ELSE
          READ (LURD1,2000) KB; IF (EOF(LURD1).NE.0.0E0) GO TO 900
*ENDIF
2000      FORMAT(A1)
          IF (KB.EQ.KDSUP) [
              KSUPRM=1
              WRITE (KONSOL,2010)
2010          FORMAT("Use SUPREM generated profile.")
          ]
          ELSE [
              KSUPRM=0
#
#.....READ IN IMPLANT PARAMETERS
              K=1; WHILE ((KB.NE.KDOPE(K)).AND.(K.LE.6))   K=K+1
              IF (K.GT.6) GO TO 801
              IF ((K.EQ.1).OR.(K.EQ.6)) TYPE=-1.0
              ELSE                      TYPE= 1.0
*IF HP1000
              CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
              CALL CODE (KEND*2); READ (KBUF,*) CSUB
*ELSE
              READ (LURD1,*) CSUB; IF (EOF(LURD1).NE.0.0E0) GO TO 900
*ENDIF
              IF (CSUB.EQ.0.0E0) GO TO 802
              CSUB=SIGN(CSUB*1E15,TYPE)
              WRITE (KONSOL,3000) KB,CSUB
3000          FORMAT(5X,"Substrate dopant: ",A1,^
                     ", concentration=",1PG10.3,"cm-3")
              DO KIMPL=1,3 [
                  IF (KIMPL.EQ.2) [
*IF HP1000
                      CALL READF (KDCB,KERR,KBUF,LRECL,KEND)
                      IF (KEND.EQ.-1) GO TO 900
                      CALL CODE  (KEND*2); READ (KBUF,*) XIMPLO,XIMPL1
*ELSE
                      READ (LURD1,*) XIMPLO,XIMPL1
                      IF (EOF(LURD1).NE.0.0E0) GO TO 900
*ENDIF
                  ]
*IF HP1000
```

```
                    CALL READF (KDCB,KERR,KBUF,LRECL,KEND)
                    IF (KEND.EQ.-1) GO TO 900; CALL CODE  (KEND*2)
                    READ (KBUF,2000) KD
*ELSE
                    READ (LURD1,2000) KD; IF (EOF(LURD1).NE.0.0E0) GO TO 900
*ENDIF
                    K=1; WHILE ((KD.NE.KDOPE(K)).AND.(K.LE.6)) K=K+1
                    DOPE(KIMPL)=K
*IF HP1000
                    CALL READF (KDCB,KERR,KBUF,LRECL,KEND)
                    IF (KEND.EQ.-1) GO TO 900; CALL CODE  (KEND*2)
                    READ (KBUF,*) RANGE(KIMPL),STNDV(KIMPL),DOSE(KIMPL)
                    CALL READF (KDCB,KERR,KBUF,LRECL,KEND)
                    IF (KEND.EQ.-1) GO TO 900; CALL CODE  (KEND*2)
                    READ (KBUF,*) DCOEF(KIMPL)
                    CALL READF (KDCB,KERR,KBUF,LRECL,KEND)
                    IF (KEND.EQ.-1) GO TO 900; CALL CODE  (KEND*2)
                    READ (KBUF,*) TEMP(KIMPL)
                    CALL READF (KDCB,KERR,KBUF,LRECL,KEND)
                    IF (KEND.EQ.-1) GO TO 900; CALL CODE  (KEND*2)
                    READ (KBUF,*) DRVIN(KIMPL)
*ELSE
                    READ (LURD1,*) RANGE(KIMPL),STNDV(KIMPL),DOSE(KIMPL)
                    IF (EOF(LURD1).NE.0.0E0) GO TO 900
                    READ (LURD1,*) DCOEF(KIMPL)
                    IF (EOF(LURD1).NE.0.0E0) GO TO 900
                    READ (LURD1,*) TEMP(KIMPL)
                    IF (EOF(LURD1).NE.0.0E0) GO TO 900
                    READ (LURD1,*) DRVIN(KIMPL)
                    IF (EOF(LURD1).NE.0.0E0) GO TO 900
*ENDIF
#
#.....PRINT INPUT SUMMARY
                    IF ((DOPE(KIMPL).LE.6.0).AND.(DOSE(KIMPL).NE.0.0E0)) [
                         WRITE (KONSOL,3020) (NAMIMP(K,KIMPL),K=1,4),KD,^
                                             RANGE(KIMPL),STNDV(KIMPL),^
                                             DOSE(KIMPL),DCOEF(KIMPL),^
                                             TEMP(KIMPL),DRVIN(KIMPL)
3020                     FORMAT(3X,4A2,"implant: dopant=",A1,^
                         /,10X,"Range(um),        Stndv(um),    Dose(cm-2)=",^
                           2X,F7.3,",",F7.3,",",1PG10.3^
                         /,10X,"D-coeff(cm2/sec), Temp(oC), Drive-in(min)=",^
                           G10.3,",",0PF7.1,",",F7.3)
#
#.....SCALE AND LIMIT IMPLANT PARAMETERS
                         CALL CHKDP (KIMPL)
                    ]
                    ELSE [
                         DOSE(KIMPL)=0.0E0
                         XJCT(KIMPL)=0.0E0
                    ]
               ]
          ]
     ]
#
#.....SET ERROR EXIT FLAG
      ELSE KEXIT=1
#
#.....CLOSE THE FILE
500   CONTINUE
*IF HP1000
      CALL CLOSE (KDCB,KERR)
      IF (KERR.LT.0) CALL TFERR (2,KERR,NAMBUF)
*ENDIF
#
#.....DONE
      IF (KEXIT.EQ.0) RETURN
#
#.....ERRORous EXIT, REMEMBER TO UNLOCK DISPLAY MEMORY
      ELSE [
*IF HP1000&!BATCH
          WRITE (KONSOL,5000) KESC
5000      FORMAT(R1,"m")
          CALL EXEC (6)
*ELSE
          STOP
*ENDIF
      ]
#
#.....ERROR IN INPUT DATA
```



```
800   CONTINUE
      KEXIT=1
      WRITE (KONSOL,8000)
8000  FORMAT("*****  Channel length <= 0! Program terminated!  *****")
      GO TO 500
801   CONTINUE
      KEXIT=1
      WRITE (KONSOL,8001)
8001  FORMAT("***** Substrate dopant undefined! Program terminated! *****")
      GO TO 500
802   CONTINUE
      KEXIT=1
      WRITE (KONSOL,8002)
8002  FORMAT("** Zero substrate doping concentration! Program terminated! **")
      GO TO 500
#
#.....ERROR IN FILE FORMAT
900   CONTINUE
      KEXIT=1
      WRITE (KONSOL,9000)
      WRITE (KONSOL,9010)
      WRITE (KONSOL,9020)
      GO TO 500
9000  FORMAT("Input file is not properly created.",^
          /,"The format should be:",^
          /,"line  1: title line"^
          /,"line  2: drawn channel length (um)"^
          /,"line  3: lateral span of source (um)"^
          /,"line  4: lateral span of drain  (um)"^
          /,"line  5: oxide thickness: thin? and thick ? (um)"^
          /,"line  6: thin oxide location: from ? to ? (um)"^
          /,"line  7: lateral span of oxide ramp (um)"^
          /,"line  8: drawn gate location: from ? to ? (um)"^
          /,"line  9: depth of the simulated structure? (um)"^
          /,"line 10: substrate dopant: B,As,Ph,Sb,+(n-type),-(p-type),",^
                                   "F(Supren)")
9010  FORMAT("line 11: substrate doping concentration (*1E15 cm-3)"^
          /,"line 12: overall implant dopant: B, As, Ph, Sb, +, -"^
          /,"line 13:                       range(um), stndev(um) and dose (cm-2)"^
          /,"line 14:                       diffusion constant (cm2/sec)"^
          /,"line 15:                       drive in temperature (oC)"^
          /,"line 16:                       drive in time (min)"^
          /,"line 17: localized implant location: from ? to ? (um)"^
          /,"line 18:                         dopant: B, As, Ph, Sb, +, -"^
          /,"line 19:                         range, stndev and dose "^
          /,"line 20:                         diffusion constant (cm2/sec)")
9020  FORMAT("line 21:                         drive in temperature (oC)"^
          /,"line 22:                         drive in time (min)"^
          /,"line 23: source/drain implant dopant: B, As, Ph, Sb, +, -"^
          /,"line 24:                            range, stndev and dose "^
          /,"line 25:                            diffusion constant (cm2/sec)"^
          /,"line 26:                            drive in temperature (oC)"^
          /,"line 27:                            drive in time (min)")
      END
```

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE REDF2
#
#     READ INPUT PARAMETERS FROM A FILE
#
*CALL TEMA
*CALL TCOMMN
#
      DIMENSION KDOPE(6)
      DIMENSION SUPBUF(5),SUPC(400),SUPY(400)
*IF HP1000
#
#.....FILE MANAGEMENT PARAMETERS
      DIMENSION NAMBUF(10),NSTRNG(20),NAMFIL(3),^
                KDCB1(144),KSIZE(2),  KBUF(40)
#
      EQUIVALENCE (NAMBUF(1),NAMFIL(1)),(NAMBUF(5),NSECU),(NAMBUF(6),NCR)
#
      DATA KSIZE/20,80/,KTYPE/3/,KDCBS/128/,^
           LRECL/40/,   NSECU/0/,NCR/2HXX/
*ENDIF
#
#     KSUB=0 -> UNIFORM SUBSTRATE,   KSUB=1 -> NON-UNIFORM SUBSTRATE
#     KB=DOPANT OF NON-UNIFORM SUBSTRATE DEVICE
#
      DATA KDOPE/2HB ,2HA ,2HP ,2HS ,2H+ ,2H- /, KDNUL/2H$ /
      DATA NOUNIF/2HN /, UM/1<EXP>-4/, SECND/60.0<EXP>0/, LENSUP/400/
*IF HP1000
      DATA KESC /033B/
*ENDIF
#
#.....GET THE DATA FILE NAME
*IF HP1000
      DO K=1,10; NAMBUF(K)=INPFIL(K)
      CALL OPEN  (KDCB1,KERR,NAMFIL,0,NSECU,NCR,KDCBS)
      KOPEN=0
      IF (KERR.LT.0) CALL TFERR(1,KERR,NAMBUF,KOPEN)
      WHILE (KOPEN.EQ.1) [
          WRITE (KONSOL,1000)
          READ  (KEYBRD,1010) NSTRNG
1000      FORMAT("Input data file name? _")
          NPOS=1; CALL NAMR (NAMBUF,NSTRNG,40,NPOS)
          DO K=1,10; INPFIL(K)=NAMBUF(K)
#
#.....OPEN THE FILE
          CALL OPEN  (KDCB1,KERR,NAMFIL,0,NSECU,NCR,KDCBS)
          KOPEN=0
          IF (KERR.LT.0) CALL TFERR(1,KERR,NAMBUF,KOPEN)
      ]
*ELSE
      KERR=0
*ENDIF
#
#.....RESET ERROR FLAG, SKIP FIRST 10 LINES
1010  FORMAT(20A2)
      IF (KERR.GE.0) [
          KEXIT=0
*IF HP1000
          DO K=1,10 [
             CALL READF (KDCB1,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
          ]
*ENDIF
2000        FORMAT(A1)
#
#.....READ IN SUBSTRATE INFORMATIONS
*IF HP1000
        CALL READF (KDCB1,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
        CALL CODE (KEND*2); READ (KBUF,2000) KANSUB
*ELSE
        READ (LURD1,2000) KANSUB; IF (EOF(LURD1).NE.0.0<EXP>0) GO TO 900
*ENDIF
        K=1; WHILE ((KANSUB.NE.KDOPE(K)).AND.(K.LE.6)) K=K+1
        IF (K.GT.6) GO TO 801
        IF ((K.EQ.1).OR.(K.EQ.6)) TYPE=-1.0<EXP>0
        ELSE                      TYPE=1.0<EXP>0
        IF (TYPE.EQ.-1.0<EXP>0) KB=6
        ELSE                KB=5
```



```
*IF HP1000
          CALL READF (KDCB1,KERR,KBUF,LRECL,KEND);  IF (KEND.EQ.-1) GO TO 900
          CALL CODE  (KEND*2); READ (KBUF,2000) KANSUR
*ELSE
          READ (LURD1,2000) KANSUR; IF (EOF(LURD1).NE 0.0<EXP>0) GO TO 900
*ENDIF
          IF (KANSUR.NE.NOUNIF) [
*IF HP1000
             CALL READF (KDCB1,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
             CALL CODE (KEND*2); READ (KBUF,*) CSUB
*ELSE
             READ (LURD1,*) CSUB; IF (EOF(LURD1).NE.0.0<EXP>0) GO TO 900
*ENDIF
             IF (CSUB.EQ.0.0<EXP>0) GO TO 802
             CSUB=SIGN(CSUB*1<EXP>15,TYPE)
             DO N=1,NXMAX; DO M=1,NYMAX; CONC(N,M)=CSUB
             KSUB=0
             WRITE (KONSOL,3000) KANSUB,CSUB
3000         FORMAT(5X,"Uniform substrate of dopant: ",A1,^
                    ", concentration=",1PG10.3,"cm-3")
          ]
          ELSE [
             KSUB=1
             CSUB=0.0<EXP>0
          ]
#
#.....GET IMPLANT PROFILE
          DO KIMPL=1,3 [
             IF (KSUB.LE.0) [                        # CASE OF UNIFORM SUBSTRATE
                IF (KIMPL.EQ.2) [
*IF HP1000
                   CALL READF (KDCB1,KERR,KBUF,LRECL,KEND)
                   IF (KEND.EQ.-1) GO TO 900
                   CALL CODE (KEND*2); READ (KBUF,*) XIMPLO,XIMPL1
*ELSE
                   READ (LURD1,*) XIMPLO,XIMPL1
                   IF (EOF(LURD1).NE.0.0<EXP>0) GO TO 900
*ENDIF
                   XIMPLO=XIMPLO*UM
                   XIMPL1=XIMPL1*UM
                   IF (XIMPLO.GT.XIMPL1) [
                      T=XIMPLO; XIMPLO=XIMPL1; XIMPL1=T
                   ]
                   NIMPLO=LOCX(XIMPLO)
                   NIMPL1=LOCX(XIMPL1)
                   IF (NIMPLO.EQ.0) [ IF (NIMPL1.NE.0) NIMPLO=1 ]
                   ELSE [ IF (XIMPLO.NE.XPOS(NIMPLO)) NIMPLO=NIMPLO+1 ]
                ]
*IF HP1000
                CALL READF(KDCB1,KERR,KBUF,LRECL,KEND);IF (KEND.EQ.-1) GO TO 900
                CALL CODE (KEND*2); READ (KBUF,2000) KANSW1
*ELSE
                READ (LURD1,2000) KANSW1; IF (EOF(LURD1).NE.0.0<EXP>0) GO TO 900
*ENDIF
                KK=1; WHILE ((KANSW1.NE.(KDOPE(KK))).AND.(KK.LE.6)) KK=KK+1
                IF ((KK.EQ.1).OR.(KK.EQ.6)) DOPE(KIMPL)=6.0<EXP>0
                IF ((KK.GT.1).AND.(KK.LT.6)) DOPE(KIMPL)=5.0<EXP>0
                IF (KK.GT.6) DOPE(KIMPL)=0.0<EXP>0
             ]
             ELSE DOPE(1)=KB
             IF (DOPE(KIMPL).NE.0.0<EXP>0) [
*IF HP1000
                CALL READF(KDCB1,KERR,KBUF,LRECL,KEND);IF (KEND.EQ.-1) GO TO 900
                CALL CODE (KEND*2); READ (KBUF,*) SD
*ELSE
                READ (LURD1,*) SD; IF (EOF(LURD1).NE.0.0<EXP>0) GO TO 900
*ENDIF
                STNDV(KIMPL)=SD*UM
#
#.....READ IN PROFILE
*IF HP1000
                CALL READF(KDCB1,KERR,KBUF,LRECL,KEND)
                IF(KEND.EQ.-1) GO TO 900
                CALL CODE (KEND*2); READ (KBUF,1010) NSTRNG
*ELSE
                READ (LURD1,1010) NSTRNG; IF (EOF(LURD1).NE.0.0<EXP>0) GO TO 900
*ENDIF
#
#.....GET COLUMN INDEX
*IF HP1000
```

```
                  CALL READF(KDCB1,KERR,KBUF,LRECL,KEND)
                  IF(KEND.EQ.-1) GO TO 900
                  CALL CODE (KEND*2); READ (KBUF,*) ICOLMN
*ELSE
                  READ (LURD1,*) ICOLMN; IF (EOF(LURD1) NE 0.0(EXP)0) GO TO 900
*ENDIF
                  CALL REDSP (KINPL,ICOLMN,NSTRNG)
               }
               ELSE {
                  DO K=1,3 {
*IF HP1000
                     CALL READF(KDCB1,KERR,KBUF,LRECL,KEND)
                     IF (KEND.EQ.-1) GO TO 900
*ELSE
                     READ (LURD1,1010) DUMMY; IF (EOF(LURD1).NE.0.0(EXP)0) GO TO 900
*ENDIF
                     DOSE(K)=0.0(EXP)0
                     XJCT(K)=0.0(EXP)0
                     CSTEP(K)=0.0(EXP)0
                  }
               }
               IF (KSUB.EQ.1) KSUB=-1
            }
#
#.....PREPROCESS PARAMETERS, RESET SUPREM FLAG
         CALL PARMS
         KSUPRM=0
      }
#
#.....SET ERROR EXIT FLAG
      ELSE KEXIT=1
#
#.....CLOSE THE FILE
500   CONTINUE
*IF HP1000
      CALL CLOSE (KDCB1,KERR); IF (KERR.LT.0) CALL TFERR (2,KERR,NAMBUF)
*ENDIF
#
#.....DONE
      IF (KEXIT.EQ.0) RETURN
#
#.....ERROROUS EXIT, REMEMBER TO UNLOCK DISPLAY MEMORY
      ELSE {
*IF HP1000&!BATCH
         WRITE (KONSOL,5000) KESC; CALL EXEC (6)
5000     FORMAT(R1,"m")
*ELSE
         STOP
*ENDIF
      }
#
#.....ERROR IN INPUT PARAMETERS
801   CONTINUE
      KEXIT=1
      WRITE (KONSOL,8001) KANSUB
8001  FORMAT("** Un-recognizable substrate dopant**",A1,"**! ",^
         "Program terminated! **")
      GO TO 500
802   CONTINUE
      KEXIT=1
      WRITE (KONSOL,8002)
8002  FORMAT("** Zero substrate doping concentration! Program terminated! **")
      GO TO 500
#
#.....ERROR IN FILE FORMAT
900   CONTINUE
      KEXIT=1
      WRITE (KONSOL,9000)
      WRITE (KONSOL,9010)
      WRITE (KONSOL,9020)
      GO TO 500
9000  FORMAT("Input file is not properly created."^
         /,"The format should be!",^
         /,"line  1! title line"^
         /,"line  2! drawn channel length (um)"^
         /,"line  3! lateral span of source (um)"^
         /,"line  4! lateral span of drain  (um)"^
         /,"line  5! oxide thickness: thin? and thick ? (um)"^
         /,"line  6! thin oxide location! from ? to ? (um)"^
         /,"line  7! lateral span of oxide ramp (um)"^
```



```
         /,"line  8! drawn gate location: from ? to ? (um)"^
         /,"line  9! depth of the simulated structure? (um)"^
         /,"line 10! index of SUPREM input F")
9010  FORMAT("line 11! substrate dopant! B,As,Ph,Sb,+(n-type),-(p-type),"^
         /,"line 12! index of non-uniform substrate N"^
         /,"line 12.a if line 12 is not N in 1st column:"^
         /,"                 substrate doping concentration (*1E15 cm-3)"^
         /,"line 13. overall    inlant dopant! B, As, Ph, Sb, +, -"^
         /,"line 14!                   standard deviation(um)"^
         /,"line 15!                   SUPREM save file name"^
         /,"line 16!                   column index"^
         /,"line 17! localized implant location! from ? to ? (um)"^
         /,"line 18!                   dopant! B, As, Ph, Sb, +, -"^
         /,"line 19!                   standard deviation(um)"^
         /,"line 20!                   file name"^
         /,"line 21!                   column index")
9020  FORMAT("line 22! source/drain implant dopant! B, As, Ph, Sb, +, -"^
         /,"line 23!                      standard deviation(um)"^
         /,"line 24!                      file name"^
         /,"line 25!                      column index")
      END
```

```
*CALL TUDEFN
      SUBROUTINE REDGM
*
*     READ IN GEOMETRY PARAMETERS
*
*CALL TCOMMN
*
      DIMENSION MESSG1(3,2),MESSG2(5,2)
*
      DATA LNMSG1/3/,^
           MESSG1/2Hso,2Hur,2Hce,^
                 2Hdr,2Hai,2Hn /,^
           LNMSG2/5/,^
           MESSG2/2HDr,2Haw,2Hn ,2Hga,2Hte,^
                 2HGa,2Hte,2H o,2Hxi,2Hde/
*IF HP1000
      DATA KESC/033B/
*ENDIF
*
      WRITE (KONSOL,1000)
1000  FORMAT("Drawn channel length (um) ? _")
      READ  (KEYBRD,*) XCHANL
*IF BATCH
      WRITE (KONSOL,1001) XCHANL
1001  FORMAT(1P2G10.3)
*ENDIF
      IF (XCHANL.LE.0.0(EXP)0) GO TO 900
      WRITE (KONSOL,1010) (MESSG1(K,1),K=1,LNMSG1)
      READ  (KEYBRD,*) XSOURC
*IF BATCH
      WRITE (KONSOL,1001) XSOURC
*ENDIF
      WRITE (KONSOL,1010) (MESSG1(K,2),K=1,LNMSG1)
1010  FORMAT("Lateral span of ",3A2," ? (um) _")
      READ  (KEYBRD,*) XDRAIN
*IF BATCH
      WRITE (KONSOL,1001) XDRAIN
*ENDIF
      WRITE (KONSOL,1030)
1030  FORMAT("Oxide thicknesses -> gate and field (um) ??_")
      READ  (KEYBRD,*) TOX0,TOX1
*IF BATCH
      WRITE (KONSOL,1001) TOX0,TOX1
*ENDIF
      WRITE (KONSOL,1020) (MESSG2(K,2),K=1,LNMSG2)
      READ  (KEYBRD,*) XOX0,XOX1
*IF BATCH
      WRITE (KONSOL,1001) XOX0,XOX1
*ENDIF
      WRITE (KONSOL,1040)
1040  FORMAT("Span of oxide ramp (um) ? _")
      READ  (KEYBRD,*) XOXR
*IF BATCH
      WRITE (KONSOL,1001) XOXR
*ENDIF
      WRITE (KONSOL,1020) (MESSG2(K,1),K=1,LNMSG2)
1020  FORMAT(3A2," location! from () to () (um) ? _")
      READ (KONSOL,*) XGATE0,XGATE1
*IF BATCH
      WRITE (KONSOL,1001) XGATE0,XGATE1
*ENDIF
      WRITE (KONSOL,1060)
1060  FORMAT("Depth of the simulated structure (um) ? _")
      READ  (KEYBRD,*) YMAX
*IF BATCH
      WRITE (KONSOL,1001) YMAX
*ENDIF
*
*.....SCAEL INPUT PARAMETERS
      CALL CHKGM
*
*.....DONE
      RETURN
*
*.....ERROROUS EXIT -> REMEMBER TO UNLOCK DISPLAY MEMORY
900   CONTINUE
      WRITE (KONSOL,9000)
9000  FORMAT("*****  Channel length <= 0! Program terminated!  *****")
*IF HP2640A
      WRITE (KONSOL,9010) KESC
```



```
9010  FORMAT(R1,"m")
*ENDIF
*IF HP1000&!BATCH
      CALL EXEC (6)
*ELSE
      STOP
*ENDIF
      END
```

```
*CALL TWDEFN
*IF HP1000
EMA (XY2,0)
*ENDIF
      SUBROUTINE REDSP (KIMPL,ICOLMN,NSTRNG)
#
#     READ PROFILE DISTRIBUTION FROM SUPREM'S SAVE FILE
#
*CALL TEMA
*CALL TCOMMN
#
#.....BUFFER
      DIMENSION SUPBUF(5),SUPC(400),SUPY(400),NAMIMP(4,3)
*IF HP1000
#
#.....FILE MANAGEMENT PARAMETERS
      DIMENSION NAMBUF(10),NSTRNG(20),NAMFIL(3),^
                KDCB(144), KSIZE(2),  KBUF(40)
#
      EQUIVALENCE (NAMBUF(1),NAMFIL(1)),(NAMBUF(5),NSECU),(NAMBUF(6),NCR)
#
      DATA KSIZE/20,80/,KTYPE/3/,KDCBS/128/,LRECL/40/,NSECU/0/,NCR/2HXX/
*ENDIF
#
#.....SUPREM ARRAY LENGTH
      DATA LENSUP/400/
#
#.....NAME OF IMPLANTS
      DATA NAMIMP/2Hov,2Her,2Hel,2Hl ,^
                  2Hlo,2Hca,2Hl',2Hd ,^
                  2Hsr,2Hc/,2Hdr,2Hn /
#
#.....SCALING FACTORS
      DATA UM/1(EXP)-4/,  SECND/60.0(EXP)0/
*IF HP1000
#
#.....ESCAPE CODE
      DATA KESC/033B/
#
#.....OPEN SUPREM SAVE FILE
      NPOS=1; CALL NAMR (NAMBUF,NSTRNG,40,NPOS)
      CALL OPEN (KDCB,KERR,NAMFIL,0,NSECU,NCR,KDCBS)
      KOPEN=0; IF (KERR.LT.0) CALL TFERR (1,KERR,NAMBUF,KOPEN)
      WHILE (KOPEN.EQ.1) [
          WRITE (KONSOL,3000) KIMPL
          READ  (KEYBRD,1010) NSTRNG
3000      FORMAT("Implant",I2,": data file name? _")
          NPOS=1; CALL NAMR (NAMBUF,NSTRNG,40,NPOS)
#
#.....OPEN THE FILE
          CALL OPEN  (KDCB,KERR,NAMFIL,0,NSECU,NCR,KDCBS)
          KOPEN=0; IF (KERR.LT.0) CALL TFERR(1,KERR,NAMBUF,KOPEN)
      ]
*ELSE
      KERR=0
*ENDIF
#
#.....EXIT IF ERROR
      IF (KERR.LT.0) GO TO 900
1010  FORMAT(20A2)
#
#.....SKIP TITLE, COMMENT AND STEP INFO
      DO K=1,3 [
*IF HP1000
          CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
*ELSE
          READ (LURD2,1010) DUMMY; IF (EOF(LURD2).NE.0.0(EXP)0) GO TO 900
*ENDIF
      ]
#
#.....GRID INFO
*IF HP1000
      CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
      CALL CODE  (KEND*2); READ (KBUF,5010) DYSI,INTF,IPNTI,NCC,LEVEL
*ELSE
      READ (LURD2,5010) DYSI,INTF,IPNTI,NCC,LEVEL
      IF (EOF(LURD2).NE.0.0(EXP)0) GO TO 900
*ENDIF
5010  FORMAT(G13.4,4I3)
#
```



```
#.....SKIP OXIDE CONCENTRATION
      INTF1=INTF-1
      DO K=1,INTF1 [
*IF HP1000
          CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
*ELSE
          READ (LURD2,*) (DUMMY,KK=1,5); IF (EOF(LURD2).NE.0.0(EXP)0) GO TO 900
*ENDIF
      ]
#
#.....INITIALIZE BUFFER
*IF HP1000
      CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
      CALL CODE  (KEND*2); READ (KBUF,5020) (SUPBUF(K),K=1,5)
*ELSE
      READ (LURD2,5020) (SUPBUF(K),K=1,5)
      IF (EOF(LURD2).NE.0.0(EXP)0) GO TO 900
*ENDIF
5020  FORMAT(5G13.6)
      SUPY(1)=0.0(EXP)0
      SUPC(1)=ABS(SUPBUF(ICOLMN))
#
#.....READ IN THE PROFILE, SEARCH FOR CONCENTRATION PEAK
      KSTOP=LENSUP-INTF1
      KGRID=IPNTI-INTF1
      KMAX=0
      DYSI=DYSI*UM
      DYSI4=4.0(EXP)0*DYSI
      CMAX=0.0(EXP)0
      DO KP=2,KSTOP [
*IF HP1000
          CALL READF (KDCB,KERR,KBUF,LRECL,KEND); IF (KEND.EQ.-1) GO TO 900
          CALL CODE  (KEND*2); READ (KBUF,5020) (SUPBUF(K),K=1,5)
*ELSE
          READ (LURD2,5020) (SUPBUF(K),K=1,5)
          IF(EOF(LURD2).NE.0.0(EXP)0) GO TO 900
*ENDIF
          SUPC(KP)=ABS(SUPBUF(ICOLMN))
          IF (KP.LT.KGRID) SUPY(KP)=SUPY(KP-1)+DYSI
          ELSE             SUPY(KP)=SUPY(KP-1)+DYSI4
          IF (CMAX.LT.SUPC(KP)) [
              KMAX=KP
              CMAX=SUPC(KP)
          ]
      ]
      RANGE(KIMPL)=SUPY(KMAX)
      CPK=CMAX
*IF HP1000
#
#.....CLOSE THE SAVE FILE
      CALL CLOSE(KDCB,KERR); IF(KERR.LT.0) CALL TFERR(2,KERR,NAMBUF)
*ENDIF
#
#.....SEARCH FOR THE JUNCTON DEPTH
      IF (CSUB.EQ.0.0(EXP)0) [
          CSUB=SIGN(SUPC(KSTOP),TYPE)
          ABSUB=ABS(CSUB)
          CPK=CPK-ABSUB
          DO K=1,KSTOP [
              SUPC(K)=AMAX1(SUPC(K)-ABSUB,0.0(EXP)0)
              DO N=1,NXMAX; DO M=1,NYMAX; CONC(N,M)=CSUB
          ]
              WRITE (KONSOL,5030) CSUB
5030          FORMAT(5X,"Non-uniform substrate with ",^
                     "background concentration = ",1PG10.3,"cm-3")
      ]
      ELSE ABSUB=ABS(CSUB)
      KJCT=KMAX
      CPEAK(KIMPL)=CPK
      REPEAT [
         KJCT=KJCT+1
         IF (SUPC(KJCT).LE.ABSUB) KGO=0
         ELSE                     KGO=1
         IF ((KGO.EQ.1).AND.(KJCT.GE.KSTOP))   KGO=-1
      ] UNTIL (KGO.LE.0)
      XJCT(KIMPL)=SUPY(KJCT)
#
#.....GET STANDARD DEVIATION OF APPROXIMATED GAUSSIAN DISTRIBUTION
      SIGMA=ABS(XJCT(KIMPL)-RANGE(KIMPL))/SQRT(2.0(EXP)0*ALOG(CPK/ABSUB))
      TWOPI=44.0(EXP)0/7.0(EXP)0
```

File: &REDSP

```
      TWODT=SIGMA*SIGMA
      DOSE(KIMPL)=SQRT(TWODT*TWOPI)*CPK
      IF (XJCT(KIMPL).NE.0.0(EXP)0) CSTEP(KIMPL)=DOSE(KIMPL)/XJCT(KIMPL)
      ELSE                          CSTEP(KIMPL)=0.0(EXP)0
#
#.....PARAMETERS
      KDOPE=DOPE(KIMPL)
      IF (KDOPE.EQ.6) DTYPE=-1.0(EXP)0
      ELSE            DTYPE= 1.0(EXP)0
      DO K=1,KSTOP; SUPC(K)=SIGN(SUPC(K),DTYPE)
      DRVIN(KIMPL)=1200.0(EXP)0
      STNDV2=STNDV(KIMPL)*STNDV(KIMPL)
      FDT=AMAX1(2.0(EXP)0*(TWODT-STNDV2),0.0(EXP)0)
      DCOEF(KIMPL)=0.25(EXP)0*FDT/DRVIN(KIMPL)
#
#.....INTERPOLATEION
      CALL DOPS2 (KIMPL,KSTOP,FDT,SUPY,SUPC)
#
#.....WRITE OUT APPROXIMATED IMPLANT PARAMETERS
      WRITE (KONSOL,1050) (NAMIMP(K,KIMPL),K=1,4),^
                          DOSE(KIMPL),SIGMA,^
                          CPEAK(KIMPL),RANGE(KIMPL),^
                          CSTEP(KIMPL),XJCT(KIMPL)
1050  FORMAT(/,"..... profile parameters of ",4A2,"implant : ",^
             /,10X,"total dose =",1PE10.3,"(cm-3)",^
               6X,"standard D =",1PE10.3,"(cm)",^
             /,10X,"peak  conc =",1PE10.3,"(cm-3)",^
               6X,"impl range =",1PE10.3,"(cm)",^
             /,10X,"ave   conc =",1PE10.3,"(cm-3)",^
               6X,"jct  depth =",1PE10.3,"(cm)")
      DOSE(KIMPL)=SIGN(DOSE(KIMPL),DTYPE)
      CSTEP(KIMPL)=SIGN(CSTEP(KIMPL),DTYPE)
#
#.....DONE
      RETURN
#
#.....ERROROUS EXIT, REMEMBER TO UNLOCK DISPLAY MEMORY
900   CONTINUE
*IF HP1000&!BATCH
      WRITE (KONSOL,9000) KIMPL,KESC; CALL EXEC (6)
9000  FORMAT("Data file of implant",I2," is not properly created.",R1,"m")
*ELSE
      WRITE (KONSOL,9001) KIMPL
9001  FORMAT("Data file of implant",I2," is not properly created.")
      STOP
*ENDIF
      END
```


File: &SADDL

```
*CALL TWDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE SADDL
#
#     CHARACTERIZE SADDLE POINT
#
*CALL TEMA
*CALL TCOMMN
#
      DATA UM/1.0(EXP)4/,  USO/700.0(EXP)0/
#
      TWOVT=VT300K+VT300K
#
#.....SEARCH THE LOCAL MAXIMA IN Y-DIRECTION
      DO N=NSOURC,NDRAIN [
           LMAX=1
           PMAX=POTSI(N,1)
           DO M=2,NYMAX; IF (POTSI(N,M).GT.PMAX) [
                LMAX=M
                PMAX=POTSI(N,M)
           ]
#
#.....IF IT IS ALSO THE MINIMUM IN X-DIRECTION -> THE SADDLE
           IF ((LMAX.NE.1).AND.(POTSI(N+1,LMAX).GT.PMAX)^
                          .AND.(POTSI(N-1,LMAX).GT.PMAX)) [
                XSADDL=XPOS(N)*UM
                YSADDL=YPOS(LMAX)*UM
                PBARR=POTSI(1,1)-PMAX
#
#.....DETERMINE THE WIDTH OF THE SADDLE
                CALL WBASE (N,LMAX,NB1,NB2)
                WBARR=XPOS(NB2)-XPOS(NB1)
#
#.....CALCULATE THE INJECTION CURRENT
                CALL INTGR (N,LMAX,NB1,NB2,SUM)
                CINJO=Q*VT300K*USO/WBARR
                CINJ=ABS(CINJO*SUM)
#
#.....SCALE AND WRITE THE RESULTS
                CINJ=CINJ*UM
                CINJO=CINJO*CNI*CNI/ABS((CONC(N,LMAX)))
                HBY1=YPOS(NB1)*UM
                HBY2=YPOS(NB2)*UM
                HBARR=HBY2-HBY1
                WBX1=XPOS(NB1)*UM
                WBX2=XPOS(NB2)*UM
                WBARR=WBARR*UM
                WRITE (KONSOL,1000) PMAX,PBARR,XSADDL,YSADDL,N,LMAX,CINJO,^
                                    WBARR,WBX1,WBX2,NB1,NB2
                IF (CINJ.NE.0.0(EXP)0)
                     WRITE (KONSOL,1020) HBARR,HBY1,HBY2,NB1,NB2,CINJ
           ]
      ]
#
#.....FORMAT STATEMENTS
1000  FORMAT("**Saddle potential =",F6.3,", barrier height=",F6.3,^
         " at (",F6.3,"um,",F6.3,"um)...(",I2,",",I2,")"^
         /,3X,"current density =",1PG10.3," Amp/cm2"^
         /,3X,"Barrier width =",0PF6.3,"um, from ",F6.3,"um to ",F6.3,^
                                         "um...(",I2,",",I2,")")
1020  FORMAT(3X,"Barrier depth =",F6.3,"um, from ",F6.3,"um to ",F6.3,^
                                         "um...(",I2,",",I2,")"^
         /,3X,"Injection current /width =",1PG10.3," Amp/um",^
         ", with USO=700.0 cm2/sec-V")
#
#.....DONE
      RETURN
      END
```

```
*CALL TWDEFN
      SUBROUTINE SAVIN
#
#      SAVE INPUT PARAMETERS INTO DISC FILE
#
*CALL TCOMMN
#
      DIMENSION KDOPE(6), MSGIMP(4,3)
*IF HP1000
#
#.....FILE MANAGEMENT PARAMETERS
      DIMENSION NAMBUF(10),NSTRNG(20),NAMFIL(3),^
                KDCB(144), KSIZE(2),  KBUF(40)
#
      EQUIVALENCE (NAMBUF(1),NAMFIL(1)),(NAMBUF(5),NSECU),(NAMBUF(6),NCR)
#
      DATA KSIZE/20,80/,KTYPE/3/,KDCBS/128/,^
           LRECL/40/,   NSECU/0/,NCR/2HXX/
*ENDIF
#
      DATA KDOPE/2HB ,2HA ,2HP ,2HS ,2H+ ,2H- /,^
           MSGIMP/2Hov,2Her,2Hal,2Hl ,^
                  2Hlo,2Hce,2Hl',2Hd ,^
                  2Her,2Hc/,2Hdr,2Hn /,^
           NAMBLK/2H  /, NAMSTR/2H**/, NAMNUL/2H0 /,^
           UM/1(EXP)4/,       SECND/60.0(EXP)0/
#
#.....GET THE DATA FILE NAME
*IF HP1000
      KFILE=1
      WHILE (KFILE.EQ.1) [
          WRITE (KONSOL,1000)
          READ (KEYBRD,1010) NSTRNG
1000      FORMAT("File name ? _")
          NPOS=1
          CALL NAMR (NAMBUF,NSTRNG,40,NPOS)
#
#.....CREAT/OPEN THE FILE
          CALL CREAT (KDCB,KERR,NAMFIL,KSIZE,KTYPE,NSECU,NCR,KDCBS)
          KFILE=0
          IF (KERR.LT.0) [
              CALL OPEN (KDCB,KERR,NAMFIL,0,NSECU,NCR,KDCBS)
              IF(KERR.LT.0) CALL TFERR(1,KERR,NAMBUF,KFILE)
          ]
      ]
      IF (KERR.LT.0) GO TO 900
*ELSE
      READ (KEYBRD,1010) NSTRNG
*ENDIF
*IF BATCH
      WRITE (KONSOL,1000)
      WRITE (KONSOL,1010) NSTRNG
*ENDIF
1010  FORMAT(20A2)
#
#.....RESET ERROR FLAG, PRINT THE TITLE
*IF HP1000
          DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
          WRITE (KBUF,3000) (NAMSTR,K=1,34); CALL WRITF(KDCB,KERR,KBUF,LRECL)
*ELSE
          WRITE (LUWR,3000) (NAMSTR,K=1,34)
*ENDIF
#
#.....PRINT GEOMETRY PARAMETERS
          XPRNT=XCHANL*UM
*IF HP1000
          DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
          WRITE (KBUF,3010) XPRNT; CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
          WRITE (LUWR,3010) XPRNT
*ENDIF
#
          XPRNT=XSOURC*UM
*IF HP1000
          DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
          WRITE (KBUF,3020) XPRNT; CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
          WRITE (LUWR,3020) XPRNT
*ENDIF
#
```



```
          XPRNT=XDRAIN*UM
*IF HP1000
          DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
          WRITE (KBUF,3030) XPRNT; CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
          WRITE (LUWR,3030) XPRNT
*ENDIF
#
          XP1=TOX0*UM
          XP2=TOX1*UM
*IF HP1000
          DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
          WRITE (KBUF,3050) XP1,XP2; CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
          WRITE (LUWR,3050) XP1,XP2
*ENDIF
#
          XP1=XOX0*UM
          XP2=XOX1*UM
*IF HP1000
          DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
          WRITE (KBUF,3060) XP1,XP2; CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
          WRITE (LUWR,3060) XP1,XP2
*ENDIF
#
          XPRNT=XOXR*UM
*IF HP1000
          DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
          WRITE (KBUF,3070) XPRNT; CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
          WRITE (LUWR,3070) XPRNT
*ENDIF
#
          XP1=XGATE0*UM
          XP2=XGATE1*UM
*IF HP1000
          DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
          WRITE (KBUF,3040) XP1,XP2; CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
          WRITE (LUWR,3040) XP1,XP2
*ENDIF
#
          XPRNT=YMAX*UM
*IF HP1000
          DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
          WRITE (KBUF,3090) XPRNT; CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
          WRITE (LUWR,3090) XPRNT
*ENDIF
#
#.....PRINT IMPLANT PARAMETERS
          IF (TYPE.GT.0.0(EXP)0) NPRNT=KDOPE(5)
          ELSE                   NPRNT=KDOPE(6)
*IF HP1000
          DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
          WRITE (KBUF,4000) NPRNT; CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
          WRITE (LUWR,4000) NPRNT
*ENDIF
#
          XPRNT=ABS(CSUB)*1(EXP)-15
*IF HP1000
          DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
          WRITE (KBUF,4010) XPRNT; CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
          WRITE (LUWR,4010) XPRNT
*ENDIF
#
          DO KIMPL=1,3 [
              IF (KIMPL.EQ.2) [
                  XP1=XIMPL0*UM
                  XP2=XIMPL1*UM
*IF HP1000
                  DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
                  WRITE (KBUF,3080) XP1,XP2
                  CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
                  WRITE (LUWR,3080) XP1,XP2
*ENDIF
```

```
                ]
                KPRNT=DOPE(KIMPL)
                IF (KPRNT.GT.0) NPRNT=KDOPE(KPRNT)
                ELSE            NPRNT=NAMNUL
*IF HP1000
                DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
                WRITE (KBUF,4020) NPRNT,(MSGIMP(K,KIMPL),K=1,4)
                CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
                WRITE (LUWR,4020) NPRNT,(MSGIMP(K,KIMPL),K=1,4)
*ENDIF
C
                XP1=RANGE(KIMPL)*UM
                XP2=STNDV(KIMPL)*UM
                XP3=ABS(DOSE(KIMPL))
*IF HP1000
                DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
                WRITE (KBUF,4030)XP1,XP2,XP3; CALL WRITF (KDCB,KERR,KBUF,LRECL)
C
                DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
                WRITE (KBUF,4040) DCOEF(KIMPL)
                CALL WRITF (KDCB,KERR,KBUF,LRECL)
C
                DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
                WRITE (KBUF,4050) TEMP(KIMPL)
                CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
                WRITE (LUWR,4030) XP1,XP2,XP3
                WRITE (LUWR,4040) DCOEF(KIMPL)
                WRITE (LUWR,4050) TEMP(KIMPL)
*ENDIF
C
                XPRNT=DRVIN(KIMPL)/SECND
*IF HP1000
                DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
                WRITE (KBUF,4060) XPRNT; CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
                WRITE (LUWR,4060) XPRNT
*ENDIF
          ]
C
C.....CLOSE THE FILE
900   CONTINUE
*IF HP1000
      CALL LOCF  (KDCB,KERR,JUNK,KRB,JUNK,JSEC)
      CALL CLOSE (KDCB,KERR,JSEC/2-KRB-1)
      IF (KERR.LT.0) CALL TFERR (2,KERR,NAMBUF)
*ENDIF
C
C.....DONE
      RETURN
C
C.....FORMAT STATEMENTS
3000  FORMAT(16A2,A1,4X,"TWIST",4X,16A2,A1)
3010  FORMAT(1PG10.2,23X,"..drawn channel length (um)")
3020  FORMAT(1PG10.2,23X,"..lateral span of source (um)")
3030  FORMAT(1PG10.2,23X,"..lateral span of drain  (um)")
3040  FORMAT(2(1PG10.2),13X,"..drawn gate location: from ? to ? (um)")
3050  FORMAT(2(1PG10.2),13X,"..oxide thickness: thin? and thick ? (um)")
3060  FORMAT(2(1PG10.2),13X,"..thin oxide location: from ? to ? (um)")
3070  FORMAT(1PG10.2,23X,"..lateral span of oxide ramp (um)")
3080  FORMAT(2(1PG10.2),13X,"..localized implant: from ? to ? (um)")
3090  FORMAT(1PG10.2,23X,"..depth of the simulated structure? (um)")
4000  FORMAT(A1,32X,"..substrate type: (+)=n-type, (-)=p-type")
4010  FORMAT(1PG10.2,23X,"..substrate doping concentration (*1E15 cm-3)")
4020  FORMAT(A1,32X,"..",4A2,"implant dopant: B, As, Ph, Sb, +, -")
4030  FORMAT(3(1PG10.2),3X,".....range(um), stndev(um) and dose (cm-2)")
4040  FORMAT(1PG10.2,23X,".....diffusion constant (cm2/sec)")
4050  FORMAT(1PG10.2,23X,".....drive-in temperature (oC)")
4060  FORMAT(1PG10.2,23X,".....drive in time (min)")
      END
```



```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE SAVOU (NX1,NX2,NY1,NY2,X1,X2,Y1,Y2,Z1,Z2,KDISP,KELEC,KFELD)
C
C     SAVE RESULTS IN A FILE
C
*CALL TEMA
*CALL TCOMMN
C
      DIMENSION MSGDSP(10,4), MSGEAH(4,2), MSGFLD(6,4)
      DIMENSION KDOPE(6),MSGIMP(4,3)
*IF HP1000
C
C.....FILE MANAGEMENT PARAMETERS
      DIMENSION NAMBUF(10),NSTRNG(20),NAMFIL(3),^
               KDCB(400), KSIZE(2),  KBUF(134)
C
      EQUIVALENCE (NAMBUF(1),NAMFIL(1)),(NAMBUF(5),NSECU),(NAMBUF(6),NCR)
C
      DATA KSIZE/200,134/,KTYPE/3/,KDCBS/384/,^
           LRECL/67/,      NSECU/0/,NCR/2HXX/, NAMBLK/2H  /
*ENDIF
C
      DATA LENDSP/10/, NUMDSP/4/,^
           MSGDSP/2HDO,2HPI,2HNG,2H C,2HON,2HCE,2HNT,2HRA,2HTI,2HON,^
                 2HFR,2HEE,2H C,2HAR,2HRI,2HER,2H P,2HRO,2HFI,2HLE,^
                 2HFI,2HEL,2HD ,2HDI,2HSR,2HRI,2HBU,2HTI,2HON,2H  ,^
                 2HPO,2HTE,2HNT,2HIA,2HL ,2H P,2HRO,2HFI,2HLE,2H  /,^
           MSGEAH/2Hel,2Hec,2Htr,2Hon,^
                 2Hho,2Hle,2H  ,2H  /,^
           LENFLD/6/, NUMFLD/4/,^
           MSGFLD/2HX-,2Hco,2Hmp,2Hon,2Hen,2Ht ,^
                 2HY-,2Hco,2Hmp,2Hon,2Hen,2Ht ,^
                 2Hra,2Hti,2Ho ,2Hof,2H X,2H/Y,^
                 2Hma,2Hgn,2Hit,2Hud,2He ,2H  /
      DATA KDOPE/2HB ,2HA ,2HP ,2HS ,2H+ ,2H- /,^
           MSGIMP/2Hov,2Her,2Hal,2Hl ,^
                 2Hlo,2Hca,2Hl',2Hd ,^
                 2Her,2Hc/,2Hdr,2Hn /,^
           NAMBLK/2H  /, NAMSTR/2H**/, NAMDOT/2H../, NCPAGE/9/
C
C.....GET THE FILE NAME
*IF HP1000
      KFILE=1
      WHILE (KFILE.EQ.1) [
           WRITE (KONSOL,1000)
           READ (KEYBRD,2000) NSTRNG
2000       FORMAT(20A2)
           NPOS=1
           CALL NAMR (NAMBUF,NSTRNG,40,NPOS)
C
C.....CREAT/OPEN THE FILE
           CALL CREAT (KDCB,KERR,NAMFIL,KSIZE,KTYPE,NSECU,NCR,KDCBS)
           KFILE=0
           IF (KERR.LT.0) [
                CALL OPEN (KDCB,KERR,NAMFIL,0,NSECU,NCR,KDCBS)
                IF(KERR.LT.0) CALL TFERR(1,KERR,NAMBUF,KFILE)
           ]
      ]
      IF (KERR.LT.0) GO TO 900
*ELSE
      READ (KEYBRD,2000) NSTRNG
*ENDIF
*IF BATCH
      WRITE (KONSOL,1000)
      WRITE (KONSOL,2000) NSTRNG
*ENDIF
1000  FORMAT("File name ? _")
C
C.....THE TITLE
*IF HP1000
      DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
      WRITE (KBUF,3000) (NAMSTR,K=1,60); CALL WRITF(KDCB,KERR,KBUF,LRECL)
*ELSE
      WRITE (LUWR,3000) (NAMSTR,K=1,60)
*ENDIF
3000  FORMAT(29A2,A1,4X,"TWIST",4X,29A2,A1)
C
```

```
#.....FUNCTION NAME
*IF HP1000
      DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
      WRITE (KBUF,4000) NAMDOT,(MSGDSP(K,KDISP),K=1,LENDSP),^
                        (MSGEAH(K,KELEC),K=1,4),(NAMDOT,K=1,41)
      CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
      WRITE (LUWR,4000) NAMDOT,(MSGDSP(K,KDISP),K=1,LENDSP),^
                        (MSGEAH(K,KELEC),K=1,4),(NAMDOT,K=1,41)
*ENDIF
4000  FORMAT(A2,2X,10A2,2X," referring to ",4A2,1X,41A2)
#
      IF (KDISP.EQ.3) [
*IF HP1000
            DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
            WRITE (KBUF,4010) NAMDOT,(MSGFLD(K,KFELD),K=1,LENFLD)
            CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
            WRITE (LUWR,4010) NAMDOT,(MSGFLD(K,KFELD),K=1,LENFLD)
*ENDIF
4010        FORMAT(A2,34X,"(",6A2,")")
      ]
#
#.....BAIS VOLTAGES
*IF HP1000
      DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
      WRITE (KBUF,4020) NAMDOT,VDB,VGB,VSB,(NAMDOT,K=1,42)
      CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
      WRITE (LUWR,4020) NAMDOT,VDB,VGB,VSB,(NAMDOT,K=1,42)
*ENDIF
4020  FORMAT(A2,3(1PG10.2),"(VDB, VGB, VSB) ",41A2,A1)
#
#.....DOMAIN DEFINITIONS
*IF HP1000
      DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
      WRITE (KBUF,5010) Z1,Z2,(NAMDOT,K=1,39); CALL WRITF(KDCB,KERR,KBUF,LRECL)
*ELSE
      WRITE (LUWR,5010) Z1,Z2,(NAMDOT,K=1,39)
*ENDIF
5010  FORMAT(2X,2(1PG10.2),10X,"(Zmin,Zmax) estimated ",38A2,A1)
#
#.....9 COLUMNS PER PAGE
      NPAGE=(NX2-NX1)/NCPAGE
      IF ((NX2-NX1).NE.(NPAGE*NCPAGE)) NPAGE=NPAGE+1
      N2=NX1-1
      NROW=NY2-NY1+1
      DO KP=1,NPAGE [
            N1=N2+1
            N2=MIN0(N2+NCPAGE,NX2)
            NCOL=N2-N1+1
#
#.....BLANK LINE BETWEEN EACH PAGE
*IF HP1000
            DO K=1,LRECL; KBUF(K)=NAMBLK; CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
            WRITE (LUWR,5555)
5555        FORMAT(2X)
*ENDIF
#
#.....PRECEDE EACH BY COUNTS OF COLUMNS AND ROWS
*IF HP1000
            CALL CODE; WRITE (KBUF,6000) NCOL,NROW,KP,(NAMDOT,K=1,39)
            CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
            WRITE (LUWR,6000) NCOL,NROW,KP,(NAMDOT,K=1,39)
*ENDIF
6000        FORMAT(2X,I5,I10,9X,"(column,row) on Page ",I3,2X,39A2)
#
#.....FOLLOWED BY X-COORDINATES
*IF HP1000
            DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
            WRITE (KBUF,6005) (K,K=N1,N2)
            CALL WRITF (KDCB,KERR,KBUF,LRECL)
            DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
            WRITE (KBUF,6010) (XPOS(K),K=N1,N2)
            CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
            WRITE (LUWR,6005) (K,K=N1,N2)
            WRITE (LUWR,6010) (XPOS(K),K=N1,N2)
```



```
*ENDIF
6005        FORMAT(11X,9(I9,4X))
6010        FORMAT (2X,"Y(CM)X(CM)",4X,9(1PG9.2,4X))
#
#.....DUMP THE NUMBERS
            DO KY=NY1,NY2 [
*IF HP1000
                  DO K=1,LRECL; KBUF(K)=NAMBLK; CALL CODE
                  WRITE (KBUF,7000) KY,YPOS(KY),(Z(K,KY),K=N1,N2)
                  CALL WRITF (KDCB,KERR,KBUF,LRECL)
*ELSE
                  WRITE (KBUF,7000) KY,YPOS(KY),(Z(K,KY),K=N1,N2)
*ENDIF
7000              FORMAT(I2,1PG9.2,2X,9(1PG13.5))
            ]
      ]
#
#.....CLOSE THE FILE
900   CONTINUE
*IF HP1000
      CALL LOCF  (KDCB,KERR,JUNK,IRB,JUNK,JSEC)
      CALL CLOSE (KDCB,KERR,JSEC/2-IRB-1)
      IF (KERR.LT.0) CALL TFERR(2,KERR,NAMBUF)
*ENDIF
#
#.....DONE
      RETURN
      END
```

File: &SKEYD

```
*CALL TUDEFH
      SUBROUTINE SKEYD
*
*     DEFINE SOFT KEYS F1=DOPING F2=CARRIER F3=FIELD F4=POTENTIAL
*
*CALL TCOMMN
*
*IF HP1000
      DATA KESC/033B/
*ENDIF
*
      WRITE (KONSOL,1000) (KESC,K=1,4)
1000  FORMAT(R1,"&f1k2a6LDoping",^
             R1,"&f2k2a7LCarrier",^
             R1,"&f3k2a5LField",^
             R1,"&f4k2a9LPotential")
*
      RETURN
      END
```


File: &SKEYF

```
*CALL TUDEFH
      SUBROUTINE SKEYF
*
*     DEFINE SOFT KEYS   F1=X-COMPONENT  F2=Y-COMPONENT  F3=RATIO OF X/Y
*                        F4=MAGNITUDE
*
*CALL TCOMMN
*
*IF HP1000
      DATA KESC/033B/
*ENDIF
*
      WRITE (KONSOL,1000) (KESC,K=1,4)
1000  FORMAT(R1,"&f1k2a11LX-component",^
             R1,"&f2k2a11LY-component",^
             R1,"&f3k2a12LRatio of X/Y",^
             R1,"&f4k2a9LMagnitude_")
*
      RETURN
      END
```

File: &SKEYP

```
*CALL TWDEFN
      SUBROUTINE SKEYP
*
*     DEFINE SOFT KEYS F1=BORON  F2=ARSENIC  F3=PHOSPHORUS
*                      F4=BB     F5=+        F6=-
*
*CALL TCOMMN
*
*
*IF HP1000
      DATA KESC/033B/
*ENDIF
*
*.....DEFINE THE KEYS
      WRITE (KONSOL,2000) (KESC,K=1,6)
2000  FORMAT(R1,"&f1k2a5LBoron",^
             R1,"&f2k2a7LArsenic",^
             R1,"&f3k2a10LPhosphorus",^
             R1,"&f4k2a2LBb",^
             R1,"&f5k2a9L+(N-type)",^
             R1,"&f6k2a9L-(P-type)")
*
*.....DONE
      RETURN
      END
```


File: &SKEYT

```
*CALL TWDEFN
      SUBROUTINE SKEYT
*
*     DEFINE SOFT KEYS   F1=2-dimensional plot   F2=3-dimensional plot
*                        F3=SAVE INTO DISC FILE  F4=PRINT THE NUMBERS
*
*CALL TCOMMN
*
*IF HP1000
      DATA KESC/033B/
*ENDIF
*
      WRITE (KONSOL,1000) (KESC,K=1,4)
1000  FORMAT(R1,"&f1k2a7L2D-plot",^
             R1,"&f2k2a7L3D-plot",^
             R1,"&f3k2a12LSave on disc",^
             R1,"&f4k2a14LPrint on paper")
*
      RETURN
      END
```

```
*CALL TUDEFN
*IF HP1000
EMA(XYZ,0)
*ENDIF
      SUBROUTINE SOLVE (KSOLV)
$
$     POTENTIAL SOLUTION SEGMENT
$
$     KSOLV IS THE SECTION FLAG WHICH DENOTES THE RE-ENTRY POINT OF NEXT CALL
$
*CALL TEMA
*CALL TCOMMN
$
      DIMENSION NAMVLT(4)
$
*IF HP1000
      DATA KESC/033B/, KBELL/007B/
*ENDIF
      DATA KYES/2HY /, NAMSTR/2H**/, ITRMAX/10/,^
           NAMVLT /2HD ,2HG ,2HS ,2HB /
$
$.....INITIALIZATION
      KINIT=1
      KPITR=1
      IF ((KSOLV.EQ.0).AND.(KLOOP.EQ.1)) [ KPHIS=0; KBIAS=0 ]
      REPEAT [
$
$.....GET APPLIED VOLTAGES
          IF (KSOLV.EQ.0) [
              WRITE (KONSOL,1000)
1000          FORMAT(/"Applied voltages: VD, VG, VS, VB ? _")
              READ  (KEYBRD,*) VD,VG,VS,VB
*IF BATCH
              WRITE (KONSOL,1001) VD,VG,VS,VB
1001          FORMAT(1P4G13.3)
*ENDIF
              VDB=VD-VB
              VSB=VS-VB
              VGB=VG-VB
              IF (KPHIS.EQ.1) PHSREF=PHSREF-VB
$
$.....CHECK IF RE-INITIALIZE
              IF (KINIT.EQ.0) [
                  WRITE (KONSOL,1005)
1005              FORMAT("Re-initialize the potential (Yes=f7/No=f8) ? _")
                  READ  (KEYBRD,2000) KANSUR
*IF BATCH
                  WRITE (KONSOL,2000) KANSUR
*ENDIF
2000              FORMAT(A1)
                  IF (KANSUR.EQ.KYES) KINIT=1
              ]
$
$.....CHECK IF RE-DEFINE ITERATION PARAMETERS
              IF (KPITR.EQ.0) [
                  WRITE (KONSOL,1007)
1007              FORMAT("With same iteration parameters (Yes=f7/No=f8) ? _")
                  READ  (KEYBRD,2000) KANSUR
*IF BATCH
                  WRITE (KONSOL,2000) KANSUR
*ENDIF
                  IF (KANSUR.NE.KYES) KPITR=1
              ]
$
$.....GET 1-D ITERATION PARAMETERS
              KSOLV=1
              IF ((KINIT.EQ.1).AND.(KPITR.EQ.1)) [
                  WRITE (KONSOL,1011)
1011              FORMAT("Absolute resolution of 1-D iteration (mV's) ? _")
                  READ  (KEYBRD,*) ATOL1
*IF BATCH
                  WRITE (KONSOL,1001) ATOL1
*ENDIF
                  ATOL1=ABS(ATOL1*1.0(EXP)-3)
              ]
$
$....GET 2-D ITERATION PARAMETERS
              IF (KPITR.EQ.1) [
                  WRITE (KONSOL,1012)
1012              FORMAT("Absolute resolution of 2-D solution (mV's) ? _")
```



```
                  READ  (KEYBRD,*) ATOL2
*IF BATCH
                  WRITE (KONSOL,1001) ATOL2
*ENDIF
                  ATOL2=ABS(ATOL2*1.0(EXP)-3)
                  WRITE (KONSOL,1020)
1020              FORMAT("Relaxation factor (1<=x<2, ~1.7) ? _")
                  READ  (KEYBRD,*) RELAX2
*IF BATCH
                  WRITE (KONSOL,1001) RELAX2
*ENDIF
                  RELAX1=RELAX2
                  WRITE (KONSOL,1040)
1040              FORMAT("Maximum count of 2-D iterations ? _")
                  READ  (KEYBRD,*) KMAX2
*IF BATCH
                  WRITE (KONSOL,1002) KMAX2
1002              FORMAT(I5)
*ENDIF
                  WRITE (KONSOL,1030)
1030              FORMAT("Convergence information per 2-D iteration",^
                    " (Yes=f7/No=f8) ? _")
                  READ  (KEYBRD,2000) KANSUR
*IF BATCH
                  WRITE (KONSOL,2000) KANSUR
*ENDIF
                  IF (KANSUR.EQ.KYES) KIMSG2=2
                  ELSE                KIMSG2=0
              ]
$
$.....CHECK IF TO SEARCH FOR A SPECIFIC SURFACE POTENTIAL
              IF (KPHIS.EQ.0) [
                  WRITE (KONSOL,1050)
1050              FORMAT("Search for specific surface potential ",^
                    "(Yes=f7/No=f8) ? _")
                  READ  (KEYBRD,2000) KANSUR
*IF BATCH
                  WRITE (KONSOL,2000) KANSUR
*ENDIF
                  IF (KANSUR.EQ.KYES) KPHIS=1
              ]
$
$.....GET SEARCHING PARAMETERS
              IF (KPHIS.EQ.1) [
                  IF (KBIAS.NE.0) [
                      WRITE (KONSOL,1055)
1055                  FORMAT("With same searching parameters ",^
                        "(Yes=f7/No=f8) ? _")
                      READ  (KEYBRD,2000) KANSUR
*IF BATCH
                      WRITE (KONSOL,2000) KANSUR
*ENDIF
                      IF (KANSUR.EQ.KYES) KNEW=0
                      ELSE                KNEW=1
                  ]
                  ELSE                    KNEW=1
                  IF (KNEW.EQ.1) [
                      WRITE (KONSOL,1060)
1060                  FORMAT("Target surface potential value ",^
                        "(reference to VS!) ? _")
                      READ  (KEYBRD,*) PHSREF
*IF BATCH
                      WRITE (KONSOL,1001) PHSREF
*ENDIF
                      PHSREF=PHSREF-VB
                      WRITE (KONSOL,1070)
1070                  FORMAT("Iterate which bias (D=Vd,G=Vg,S=Vs,B=Vb) ? _")
                      READ  (KEYBRD,2000) KANSUR
*IF BATCH
                      WRITE (KONSOL,2000) KANSUR
*ENDIF
                      DO K=1,4; IF (KANSUR.EQ.NAMVLT(K)) KBIAS=K
                      KBIAS=MIN0(4,MAX0(1,KBIAS))
$                      WRITE (KONSOL,1080)
$1080                  FORMAT("First try value ? _")
$                      READ  (KEYBRD,*) VTRY
$*IF BATCH
$                      WRITE (KONSOL,1001) VTRY
$*ENDIF
                      WRITE (KONSOL,1090)
```

```
1090            FORMAT("Searching tolerance (mV's) ? _")
                READ  (KEYBRD,*) ATOLS
*IF BATCH
                WRITE (KONSOL,1001) ATOLS
*ENDIF
                ATOLS=ABS(ATOLS*1.0(EXP)-3)
              ]
          ]
#
#.....INITIALIZE POTENTIAL BASED ON 1-D SOLUTION
          IF (KINIT.EQ.1) CALL INITL (1)
#          IF (KINIT.EQ.1) [
#               CALL INITL (1)
#               WRITE (KONSOL,2010) KBELL
#               READ  (KEYBRD,2000) KANSUR
#*IF BATCH
#               WRITE (KONSOL,2000) KANSUR
#*ENDIF
#               IF (KANSUR.EQ.KYES) RETURN
#2010           FORMAT(R1,^
#                "Check auto-initialization results (Yes=f7/No=f8) ? _")
#          ]
#      ]
#
#.....ALLOW USER TO MODIFY INITIALIZATION
        IF (KSOLV.EQ.1) KSOLV=2
#        IF (KSOLV.EQ.1) [
#             KSOLV=2
#             WRITE (KONSOL,2020)
#             READ  (KEYDRD,2000) KANSUR
#*IF BATCH
#             WRITE (KONSOL,2000) KANSUR
#*EKDIF
#2020         FORMAT("Modify initial solution (Yes=f7/No=f8) ? _")
#             IF (KANSUR.EQ.KYES) [
#                  CALL INMOD
#                  WRITE (KONSOL,2030)
#                  READ  (KEYDRD,2000) KANSUR
#                  IF (KANSUR.EQ.KYES) RETURN
#2030              FORMAT("Check modified initial solution (Yes=f7/No=f8) ? _")
#             ]
#        ]
#
#.....END OF INITIALIZATION
        IF (KSOLV.EQ.2) [
             KSOLV=3
             WRITE (KONSOL,2040) (NAMSTR,K=1,24)
2040         FORMAT(/,12A2,5X,"END OF INITIALIZATION",5X,12A2/)
        ]
#
#.....2-D ITERATION
        REPEAT [
             IF (KSOLV.EQ.3) [
                  IF (KMAX2.NE.0) [
                       CALL POSSN
#
#.....SEARCHING LOOP
                       IF (KPHIS.NE.0) [
                            CALL PHMIN (1,PHIS,LPHIS)
                            KOUNT=0
                            WHILE ((ABS(PHIS-PHSREF).GT.ATOL2).AND.^
                                   (KOUNT.LT.ITRMAX)) [
                                 IF (KOUNT.EQ.0) [
                                      IF (PHIS.GT.PHSREF) SGN=-1.0(EXP)0
                                      ELSE                SGN= 1.0(EXP)0
                                      DVTRY=SGN*VT300K
                                      IF ((KBIAS.EQ.1).OR.(KBIAS.EQ.4))
                                           DVTRY=-DVTRY
                                 ]
                                 IF (ABS(DVTRY).GT.ATOLS) [
                                      XPHS=XPOS(LPHIS)*1.0(EXP)4
                                      DPHS=PHIS-PHSREF
                                      WRITE (KONSOL,2050)PHIS,XPHS,LPHIS,DPHS,DVTRY
2050                                  FORMAT("** Phis =",F7.3,^
                                          " at X =",F7.3,"um (",I2,"), Dphs=",^
                                           F7.3,", Dv=",F7.3)
                                      IF(KBIAS.EQ.1)[  VDB=VDB+DVTRY; VBIAS=VDB+VB  ]
                                      IF(KBIAS.EQ.2)[  VGB=VGB+DVTRY; VBIAS=VGB+VB  ]
                                      IF(KBIAS.EQ.3)[  VSB=VSB+DVTRY; VBIAS=VSB+VB  ]
                                      IF(KBIAS.EQ.4)[
```



```
                                           VDB=VDB-DVTRY
                                           VGB=VGB-DVTRY
                                           VSB=VSB-DVTRY
                                           PHSREF=PHSREF-DVTRY
                                           VB=VB+DVTRY
                                           VBIAS=VB
                                      ]
                                      WRITE (KONSOL,8000) NAMVLT(KBIAS),VBIAS,^
                                                         (NAMSTR,K=1,31)
8000                                  FORMAT("Try V",A1," = ",F7.3,31A2)
                                      PHSOLD=PHIS
                                      KOUNT=KOUNT+1
                                      CALL INITL (0)
                                      CALL POSSN
                                      CALL PHMIN (1,PHIS,LPHIS)
                                      DVTRY=-DVTRY*(PHSREF-PHIS)/(PHSOLD-PHIS)
                                 ]
                                 ELSE  KOUNT=ITRMAX
                            ]
                       ]
#
#.....CHECK RESULTS
                       CALL CHECK
                       KSOLV=4
*IF IBATCH
                       WRITE (KONSOL,3030) KBELL
3030                   FORMAT (R1,"_")
*ENDIF
                       WRITE (KONSOL,3031)
3031                   FORMAT("Check results (Yes=f7/No=f8) ? _")
                       READ  (KEYBRD,2000) KANSUR
                       IF (KANSUR.EQ.KYES) RETURN
                  ]
                  ELSE [
                       KSOLV=5
                       KGO2=0
                  ]
             ]
#
#.....CHECK IF MORE 2-D ITERATION
             IF (KSOLV.EQ.4) [
                  KSOLV=3
                  WRITE (KONSOL,3040)
3040              FORMAT("More iterations (Yes=f7/No=f8) ? _")
                  READ  (KEYBRD,2000) KANSUR
*IF BATCH
                  WRITE (KONSOL,2000) KANSUR
*ENDIF
                  IF (KANSUR.EQ.KYES) KGO2=1
                  ELSE                KGO2=0
             ]
        ] UNTIL (KGO2.EQ.0)
        WRITE (KONSOL,3050) (NAMSTR,K=1,20)
3050    FORMAT(/,9A2,A1,5X,"END OF TWO DIMENSIONAL SOLUTION",5X,9A2,A1/)
#
#.....CHECK IF ANY MORE BIAS POINTS
        KSOLV=0
        KINIT=0
        KPITR=0
        IF (KPHIS.EQ.1) PHSREF=PHSREF+VB
        WRITE (KONSOL,4000)
4000    FORMAT("Another bias point (Yes=f7/No=f8) ? _")
        READ  (KEYBRD,2000) KANSUR
*IF BATCH
        WRITE (KONSOL,2000) KANSUR
*ENDIF
        IF (KANSUR.EQ.KYES) KGO=1
        ELSE                KGO=0
    ] UNTIL (KGO.EQ.0)
#
#.....DONE
    KSOLV=0
    KBIAS=0
    KPHIS=0
    RETURN
    END
```

File: &SETPA

```
*CALL TWDEFN
      SUBROUTINE SETPA
#
#     SETUP MESH AND DOPING PROFILES
#
*CALL TCOMMN
#
#.....SET UP MESH
      IF (NEWMSH.EQ.1) CALL TMESH
#
#.....GENERATE DOPING PROFILE IF MESH CHANGED OR NEW PROFILE DEFINED
      IF ((NEWDOP.EQ.1).OR.(NEWMSH.EQ.1)) [
          IF (KSUPRM.EQ.0) CALL DOPNG
          ELSE [
              IF (KREAD.EQ.0) CALL DOPS1
              ELSE            CALL REDF2
          ]
      ]
#
#.....DONE
      RETURN
      END
```


File: &TFERR

```
*CALL TWDEFN
      SUBROUTINE TFERR (KFLAG,KERR,NAMBUF,KRET)
#
#     HANDLE THE ERROROUS CASES IN FILE OPEN/CLOSE
#
*CALL TCOMMN
#
      DIMENSION NAMBUF(1)
#
      DATA KYES/2HY /
#
# ....IF KFLAG=1, ERROR IN OPEN FILE:
#...................ISSUE WARNING MESSAGE, SET REOPEN FLAG
      IF (KFLAG.EQ.1) [
          WRITE (KONSOL,1000) KERR,(NAMBUF(K),K=1,3),NAMBUF(5),NAMBUF(6)
          READ  (KEYBRD,2000) KANSWR
          IF (KANSWR.EQ.KYES) KRET=1
          ELSE                KRET=0
1000      FORMAT("Open error ",I3,", in file ",3A2,":",A2,":",A2,^
                 "! Another try (Yes=f7/No=f8) ? _")
2000      FORMAT(A1)
      ]
#
#.....ELSE, ERROR IN CLOSE FILE: ISSUE WARNING MESSAGE
      ELSE [
          WRITE (KONSOL,3000) KERR,(NAMBUF(K),K=1,3),NAMBUF(5),NAMBUF(6)
3000      FORMAT("Error ",I3," in closing file ",3A2,":",A2,":",A2)
      ]
#
#.....DONE
      RETURN
      END
```

File: &TFUNC

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      FUNCTION FUNC (X,Y)
#
#     INTER-POLATE THE FUNCTIONAL VALUE OF PLOTTING ARRAY
#
#     F(1)=Z(NX1,NY1)                   F(2)=Z(NX1+1,NY1)
#     F(3)=Z(NX1,NY1+1)                 F(4)=Z(NX1+1,NY1+1)
#
*CALL TEMA
*CALL TCOMMN
#
      DIMENSION F(4)
#
#.....STATEMENT FUNCTIONS
      ZUPPR(X)=(X-X1)*(F(2)-F(1))/DX+F(1)
      ZDOWN(X)=(X-X1)*(F(4)-F(3))/DX+F(3)
      ZLEFT(Y)=(Y-Y1)*(F(3)-F(1))/DY+F(1)
      ZRITE(Y)=(Y-Y1)*(F(4)-F(2))/DY+F(2)
#
#.....DEFINE FOUR CORNERS
      NX1=MAXO(LOCX(X),1);                        X1=XPOS(NX1)
      NY1=MAXO(LOCY(Y),1);                        Y1=YPOS(NY1)
      IF (NX1.LT NXMAX) DX=DELX(NX1)
      ELSE              DX=0.0(EXP)0
      X2=X1+DX
      IF (NY1.LT.NYMAX) DY=DELY(NY1)
      ELSE              DY=0.0(EXP)0
      Y2=Y1+DY
#
#.....GET FUNCTION VALUES AT THE CORNERS
      DO K=1,4; F(K)=Z(NX1,NY1)                   # INITIALIZATION
      IF (NX1.LT.NXMAX)                           F(2)=Z(NX1+1,NY1)
      IF (NY1.LT.NYMAX)                           F(3)=Z(NX1,NY1+1)
      IF((NX1.LT.NXMAX).AND.(NY1.LT.NYMAX))  F(4)=Z(NX1+1,NY1+1)
      ELSE [
          IF (NX1.LT.NXMAX)                       F(4)=F(2)
          IF (NY1.LT.NYMAX)                       F(4)=F(3)
      ]
#
#.....INTERPORLATE THE FUNCTION VALUE AT (X,Y)
      IF ((DX.NE.0.0(EXP)0).AND.(DY.NE.0.0(EXP)0)) [
          FUNC=0.5(EXP)0*((X-X1)*(ZRITE(Y)-ZLEFT(Y))/DX+ZLEFT(Y)+^
                     (Y-Y1)*(ZDOWN(X)-ZUPPR(X))/DY+ZUPPR(X))
      ]
      ELSE [
          IF (DX.NE.0.0(EXP)0) FUNC=ZUPPR(X)
          IF (DY.NE.0.0(EXP)0) FUNC=ZLEFT(Y)
      ]
#
#.....DONE
      RETURN
      END
```


File: &TMESH

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE TMESH
#
#     SET-UP MESH AND CALCULATE RELATED COEFFICIENTS
#
*CALL TEMA
*CALL TCOMMN
#
#.....EQUIVALENCE EMA ARRAYS INTO ONE-DIMENSIONAL ARRAYS
      DIMENSION ODCNSI(1),ODCSSI(1),ODCESI(1),ODCWSI(1),ODQONE(1),^
                ODCNOX(1),ODCSOX(1),ODCEOX(1),ODCWOX(1),^
                ODCNOI(1),ODCSOI(1)
      EQUIVALENCE (ODCNSI(1),CNORTH(1,1)),(ODCSSI(1),CSOUTH(1,1)),^
                  (ODCESI(1),CEAST(1,1)), (ODCWSI(1),CWEST(1,1)),^
                  (ODQONE(1),QONE(1,1)),^
                  (ODCNOX(1),CNOX(1,1)),(ODCSOX(1),CSOX(1,1)),^
                  (ODCEOX(1),CEOX(1,1)),(ODCWOX(1),CWOX(1,1)),^
                  (ODCNOI(1),CNOXI(1,1)),(ODCSOI(1),CSOXI(1,1))
#
#.....MOCK'S CONSTANT
      DATA DELTAX/0.1(EXP)0/,    DELTAY/0.05(EXP)0/
#
#.....SILICON MESH = 50X40
      NXMAX=50
      NYMAX=40
#
#.....OXIDE MESH = 50X(2+1) [ SHARES INTERFACE WITH SILICON MESH ]
      IF ((TOXO.NE.0.0(EXP)0).OR.(TOX1 NE.0.0(EXP)0)) NOXIDE=2
      ELSE                                            NOXIDE=0
#
#.....ALLOCATE SOURCE AND DRAIN DOMAINS
      XDIFF=XSOURC+XDRAIN
      NXMX1=NXMAX+1
      IF (XDIFF.NE.0.0(EXP)0) [
          NDIFF=10
          NS=NDIFF*(XSOURC/XDIFF);            ND=NDIFF-NS
          NSRCO=NS+1;                         NDRNO=NXMAX-ND
          NSRC1=NSRCO;                        NDRN1=NDRNO
      ]
      ELSE [
          NDIFF=0
          NSRCO=0;                            NDRNO=NXMX1
          NSRC1=1;                            NDRN1=NXMAX
      ]
#
#.....MOCK'S ALGORITHM APPLIED ONLY TO EVEN X-MESHES IN DRAWN CHANNEL REGION
      NCHANL=MINO(NDRN1-NSRC1,NXMAX)
      NHALF=NCHANL/2
      IF ((NHALF*2).NE.NCHANL) [
          NCHANL=NCHANL-1
          NDRN1=NDRN1-1
          NDRNO=NDRNO-1
          NXMAX=NXMAX-1
      ]
#
#.....DETERMINE X-MESH SIZES IN DRAWN CHANNEL REGION (SYMMETRICAL)
      CALL TMOCK (0.5(EXP)0*XCHANL,NHALF,NSRC1,1)
      NSTOP=NSRC1+NHALF-1
      IMAGE=NDRN1
      DO K=NSRC1,NSTOP [
          IMAGE=IMAGE-1
          DELX(IMAGE)=DELX(K)
      ]
#
#.....DETERMINE X-MESH SIZES IN SOURCE AND DRAIN DOMAINS
      IF (NSRCO.GT.1)      CALL TMOCK (XSOURC,NS,NSRCO-1,-1)
      IF (NDRNO.LT.NXMAX) CALL TMOCK (XDRAIN,ND,NDRNO,1)
#
#.....DEFINE ARRAY XPOS, X-ORGIN AT THE LEFT EDGE OF DRAWN CHANNEL REGION
      XPOS(NSRC1)=0.0(EXP)0
      NS1=NSRC1+1
      NS2=NSRC1-1
      ZXXX=0.0(EXP)0; DO K=NS1,NXMAX [ ZXXX=ZXXX+DELX(K-1); XPOS(K)=ZXXX ]
      IF (NSRCO.NE.0) [
          ZXXX=0.0(EXP)0
          FOR (K=NS2; K>=1; K=K-1) [ ZXXX=ZXXX-DELX(K); XPOS(K)=ZXXX ]
      ]
```

```
#
#.....GET ALL LOCATION INDICES
      NGATE0=LOCX(XGATE0)
      NGATE1=LOCX(XGATE1)
      IF (NGATE0.EQ.0) [    IF (NGATE1.NE.0)            NGATE0=1         ]
      ELSE             [    IF (XGATE0.NE.XPOS(NGATE0)) NGATE0=NGATE0+1  ]
      NOX0=LOCX(XOX0)
      NOX1=LOCX(XOX1)
      IF (NOX0.EQ.0)   [    IF (NOX1.NE.0)              NOX0=1           ]
      ELSE             [    IF (XOX0.NE.XPOS(NOX0))     NOX0=NOX0+1      ]
      IF (KSUPRN.EQ.0) [
          NIMPL0=LOCX(XIMPL0)
          NIMPL1=LOCX(XIMPL1)
          IF (NIMPL0.EQ.0) [    IF (NIMPL1.NE.0)            NIMPL0=1         ]
          ELSE             [    IF (XIMPL0.NE.XPOS(NIMPL0)) NIMPL0=NIMPL0+1  ]
      ]
#
#.....CALCULATE X-COEFFICIENT ARRAYS
      OVRDX=1.0(EXP)0/DELX(1)
      Z(1,1)=OVRDX*OVRDX
      Z(1,2)=0.0(EXP)0
      NXM1=NXMAX-1
      DO K=2,NXM1 [
          OVRDX1=OVRDX
          OVRDX=1.0(EXP)0/DELX(K)
          Z(K,1)=OVRDX*OVRDX
          Z(K,2)=OVRDX*OVRDX1
      ]
      Z(NXMAX,1)=0.0(EXP)0
      Z(NXMAX,2)=Z(NXM1,1)
#
#.....DETERMINE Y-MESH
      CALL TMOCK (YMAX,NYMAX-1,1,2)
      IF (KSUPRN.EQ.0) XJ=AMAX1(XJCT(1),XJCT(2),XJCT(3))
      ELSE             XJ=0.5(EXP)-4
      IF (XJ.GT.0.0(EXP)0) [
          Y10=DELY(1)
          KS=MIN0(10,NYMAX)
          DO K=2,KS;  Y10=Y10+DELY(K)
          IF (Y10.GT.XJ) [
              CALL TMOCK (XJ,10,1,2)
              NYM1=NYMAX-1
              NYMESH=NYM1-10
              YMESH=(YMAX-XJ)/NYMESH
              DO K=11,NYM1; DELY(K)=YMESH
          ]
      ]
#
#.....DEFINE ARRAY YPOS, Y-ORGIN AT INTERFACE
      YPOS(1)=0.0(EXP)0
      ZXXX=0.0(EXP)0; DO K=2,NYMAX [ ZXXX=ZXXX+DELY(K-1); YPOS(K)=ZXXX ]
#
#.....CALCULATE Y-COEFFICIENT ARRAYS
      OVRDY=1.0(EXP)0/DELY(1)
      Z(1,3)=OVRDY*OVRDY
      Z(1,4)=0.0(EXP)0
      NYM1=NYMAX-1
      DO K=2,NYM1 [
          OVRDY1=OVRDY
          OVRDY=1.0(EXP)0/DELY(K)
          Z(K,3)=OVRDY*OVRDY
          Z(K,4)=OVRDY*OVRDY1
      ]
      Z(NYMAX,3)=0.0(EXP)0
      Z(NYMAX,4)=Z(NYM1,3)
#
#.....DEFINE OXIDE MESH
      DO K=1,NXMAX [ DELOX(K,1)=0.0(EXP)0; DELOX(K,2)=0.0(EXP)0 ]  # INIT
      DELYOX=DELY(1)/(1.0(EXP)0-DELTAY)                             # MOCK'S FORMULA
      EOXESI=EPSIO2/EPSI                                            # DIELECTRIC
      TOXX0=TOX0*EOXESI;              TOXX1=TOX1*EOXESI
      DELS0=AMIN1(DELYOX,TOXX0);      DELS1=AMIN1(DELYOX,TOXX1)
      DELT0=TOXX0-DELS0;              DELT1=TOXX1-DELS1
      XSTOP=XOX0-XOXR                                     # SOURCE SIDE
      NSTOP=LOCX(XSTOP)
      IF (NSTOP.GE.1)
          DO K=1,NSTOP [
              DELOX(K,1)=DELS1
              DELOX(K,2)=DELT1
          ]
```



```
      IF ((NOX0.LE.NOX1).AND.(NOX1.NE.0))                         # MIDDLE PART
          DO K=NOX0,NOX1 [
              DELOX(K,1)=DELS0
              DELOX(K,2)=DELT0
          ]
      XSTRT=XOX1+XOXR                                             # DRAIN SIDE
      NSTRT=LOCX(XSTRT)
      IF      (NSTRT.EQ.0)           NSTRT=1
      ELSE IF (XSTRT.NE.LOCX(XSTRT)) NSTRT=NSTRT+1
      IF (NSTRT.LE.NXMAX) [
          DO K=NSTRT,NXMAX [
              DELOX(K,1)=DELS1
              DELOX(K,2)=DELT1
          ]
      ]
      IF (XOXR.NE.0.0(EXP)0) [                                      # THE RAMPS
          SLOPE=(TOXX1-TOXX0)/XOXR
          N1=NSTOP+1                                              # SOURCE RAMP
          N2=NOX0-1
          IF ((N2.GE.N1).AND.(N1.NE.0)) [
              XR=XSTOP
              DO K=N1,N2 [
                  TOXX=TOXX1-SLOPE*(XPOS(K)-XR)
                  DELS=AMIN1(TOXX,DELYOX)
                  DELOX(K,1)=DELS
                  DELOX(K,2)=TOXX-DELS
              ]
          ]
          N1=NOX1+1                                               # DRAIN RAMP
          N2=NSTRT-1
          IF ((N2.GE.N1).AND.(N1.NE.0)) [
              XR=XOX1
              DO K=N1,N2 [
                  TOXX=TOXX0+SLOPE*(XPOS(K)-XR)
                  DELS=AMIN1(TOXX,DELYOX)
                  DELOX(K,1)=DELS; DELOX(K,2)=TOXX-DELS
              ]
          ]
      ]
#
#.....CALCULATE Y-COEFFICIENT ARRAYS ASSOCIATED WITH OXIDE
      ESIEOX=EPSI/EPSIO2
      OVRDY=1.0(EXP)0/DELY(1)
      DO K=1,NXMAX [
          DARG=DELOX(K,1)
          IF (DARG.NE.0.0(EXP)0) OVRDY1=1.0(EXP)0/DARG
          ELSE                   OVRDY1=0.0(EXP)0
          DARG=DELOX(K,2)
          IF (DARG.NE.0.0(EXP)0) OVRDY2=1.0(EXP)0/DARG
          ELSE                   OVRDY2=0.0(EXP)0
          Z(K,5)=ESIEOX*OVRDY1*OVRDY1
          Z(K,6)=OVRDY1*OVRDY2
          Z(K,7)=OVRDY2*OVRDY2
          Z(K,8)=EOXESI*OVRDY1*OVRDY
      ]
#
#.....CALCULATE POISSON EQUATION COEFFICIENT ARRAYS
      QONEPS=Q/EPSI
      DO KX=1,NXMAX [
          CE=Z(KX,1)
          CW=Z(KX,2)
#
#..........INSIDE SILICON
          DO KY=1,NYMAX [
              CS=Z(KY,3)
              IF (KY.EQ.1) CN=Z(KX,8)
              ELSE         CN=Z(KY,4)
              OMEGA=1.0(EXP)0/(CE+CW+CS+CN)
#
# PERFORM SUBSCRIPT CALCULATION EXACTLY ONCE: array(I,J) -> array(K)
#         ONE-DIMENSIONAL OFFSET VALUE : K=(J-1)*50+I
#
              KXY=(KY-1)*50+KX
              ODQONE(KXY)=QONEPS*OMEGA
              ODCESI(KXY)=CE*OMEGA
              ODCWSI(KXY)=CW*OMEGA
              ODCSSI(KXY)=CS*OMEGA
              ODCNSI(KXY)=CN*OMEGA
              OMEGA=1.0(EXP)0/(CN+CS)
              IF (KY.EQ.1) [
```

File: &TMESH

```
                QONESI(KX)=QDHEPS*OMEGA
                CNSI(KX)=CN*OMEGA
                CSSI(KX)=CS*OMEGA
            ]
            ELSE [
                IF (KX.EQ.1) [
                    QONEI(KY-1)=QDHEPS*OMEGA
                    CNI(KY-1)=CN*OMEGA
                    CSI(KY-1)=CS*OMEGA
                ]
            ]
        ]
#
#.....INSIDE OXIDE
        DO KOX=1,2 [
            IF (KOX.EQ.1) [ CS=Z(KX,5); CN=Z(KX,6) ]
            ELSE          [ CS=Z(KX,7); CN=0.0(EXP)0    ]
            OMEGA=1.0(EXP)0/(CE+CU+CS+CN)
#
# PERFORM SUBSCRIPT CALCULATION EXACTLY ONCE: array(I,J) -> array(K)
#       ONE-DIMENSIONAL OFFSET VALUE : K=(J-1)*NXMAX+I
#
            KXY=NXMAX*(KOX-1)+KX
            ODCEOX(KXY)=CE*OMEGA
            ODCUOX(KXY)=CU*OMEGA
            ODCSOX(KXY)=CS*OMEGA
            ODCNOX(KXY)=CN*OMEGA
            IF (KOX.EQ.1) [
                OMEGA=1.0(EXP)0/(CN+CS)
                ODCNOI(KXY)=CN*OMEGA
                ODCSOI(KXY)=CS*OMEGA
            ]
            ELSE [
                ODCNOI(KXY)=0.0(EXP)0
                ODCSOI(KXY)=1.0(EXP)0
            ]
        ]
    ]
#
#.....DONE
      RETURN
      END
```


File: &TNOCK

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE TNOCK (WMESH,NMESH,NSTRT,KFLAG)
#
#     SETUP THE MESH SIZE ARRAY BASED UPON NOCK'S METHOD
#
*CALL TEMA
*CALL TCOMMN
#
#.....NOCK'S CONSTANTS
      DATA DELTAX/0.1(EXP)0/,    DELTAY/0.05(EXP)0/
#
      IF (IABS(KFLAG).EQ.1) DELTA=1.0(EXP)0/(1.0(EXP)0-DELTAX)
      ELSE                  DELTA=1.0(EXP)0/(1.0(EXP)0-DELTAY)
#
#.....DETERMINE THE MINIMUM MESH SIZE
      PROD=1
      SUM=1
      DO K=2,NMESH [
          PROD=PROD*DELTA
          SUM=SUM+PROD
      ]
      SIZMIN=WMESH/SUM
#
#.....LOAD THE ARRAY
      NI=NSTRT+ISIGN(1,KFLAG)
      NSTOP=NSTRT+ISIGN(NMESH-1,KFLAG)
      ND=ISIGN(1,KFLAG)
      IF (IABS(KFLAG).EQ.1) [
          DELX(NSTRT)=SIZMIN
          IF (ND.GT.0) [
              FOR (K=NI; K<=NSTOP; K=K+ND) DELX(K)=DELX(K-ND)*DELTA
          ]
          ELSE [
              FOR (K=NI; K>=NSTOP; K=K+ND) DELX(K)=DELX(K-ND)*DELTA
          ]
      ]
      ELSE [
          DELY(NSTRT)=SIZMIN
          IF (ND.GT.0) [
              FOR (K=NI; K<=NSTOP; K=K+ND) DELY(K)=DELY(K-ND)*DELTA
          ]
          ELSE [
              FOR (K=NI; K>=NSTOP; K=K+ND) DELY(K)=DELY(K-ND)*DELTA
          ]
      ]
#
#.....DONE
      RETURN
      END
```

File: &TPRNT

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE TPRNT (NX1,NX2,NY1,NY2,X1,X2,Y1,Y2,Z1,Z2,KDISP,KELEC,KFELD)
#
#     PRINT THE RESULT
#
*CALL TEMA
*CALL TCOMMN
#
#
      DIMENSION MSGDSP(10,4), MSGEAN(4,2), MSGFLD(6,4)
#
      DATA LUPRT/6/, LENDSP/10/, NUMDSP/4/,^
           MSGDSP/2HDO,2HPI,2HNG,2H C,2HON,2HCE,2HNT,2HRA,2HTI,2HON,^
                  2HFR,2HEE,2H C,2HAR,2HRI,2HER,2H P,2HRO,2HFI,2HLE,^
                  2HFI,2HEL,2HD ,2HDI,2HSR,2HRI,2HBU,2HTI,2HON,2H  ,^
                  2HPO,2HTE,2HNT,2HIA,2HL ,2H P,2HRO,2HFI,2HLE,2H  /,^
           MSGEAN/2Hel,2Hec,2Htr,2Hon,^
                  2Hho,2Hle,2H  ,2H  /,^
           LENFLD/6/, NUNFLD/4/,^
           MSGFLD/2HX-,2Hco,2Hmp,2Hon,2Hen,2Ht ,^
                  2HY-,2Hco,2Hmp,2Hon,2Hen,2Ht ,^
                  2Hra,2Hti,2Ho ,2Hof,2H X,2H/Y,^
                  2Hme,2Hgn,2Hit,2Hud,2He ,2H  /
      DATA NAMBLK/2H  /, NAMSTR/2H**/, NAMDOT/2H../, NCPAGE/9/
#
#.....THE TITLE
      WRITE (LUPRT,3000) (NAMSTR,K=1,60)
3000  FORMAT(29A2,A1,4X,"TWIST",4X,29A2,A1)
#
#.....FUNCTION NAME
      WRITE (LUPRT,4000) NAMDOT,(MSGDSP(K,KDISP),K=1,LENDSP),^
                         (MSGEAN(K,KELEC),K=1,4),(NAMDOT,K=1,41)
4000  FORMAT(A2,2X,10A2,2X," referring to ",4A2,1X,41A2)
#
      IF (KDISP.EQ.3) [
           WRITE (LUPRT,4010) NAMDOT,(MSGFLD(K,KFELD),K=1,LENFLD)
4010       FORMAT(A2,34X,"(",6A2,")")
      ]
#
#.....BAIS VOLTAGES
      WRITE (LUPRT,4020) NAMDOT,VDB,VGB,VSB,(NAMDOT,K=1,42)
4020  FORMAT(A2,3(1PG10.2),"(VDB, VGB, VSB) ",41A2,A1)
#
#.....DOMAIN DEFINITIONS
      WRITE (LUPRT,5010) Z1,Z2,(NAMDOT,K=1,39)
5010  FORMAT(2X,2(1PG10.2),10X,"(Zmin,Zmax) estimated ",38A2,A1)
#
#.....9 COLUMNS PER PAGE
      NPAGE=(NX2-NX1)/NCPAGE
      IF ((NX2-NX1).NE.(NPAGE*NCPAGE)) NPAGE=NPAGE+1
      N2=NX1-1
      NROW=NY2-NY1+1
      DO KP=1,NPAGE [
           N1=N2+1
           N2=MIN0(N2+NCPAGE,NX2)
           NCOL=N2-N1+1
#
#.....BLANK LINE BETWEEN EACH PAGE, PRECEDE EACH BY COUNTS OF COLUMNS AND ROWS
           WRITE (LUPRT,6000) NCOL,NROW,KP,(NAMDOT,K=1,39)
6000       FORMAT(/,2X,I5,I10,9X,"(column,row) on Page ",I3,2X,39A2)
#
#.....FOLLOWED BY X-COORDINATES
           WRITE (LUPRT,6005) (K,K=N1,N2)
           WRITE (LUPRT,6010) (XPOS(K),K=N1,N2)
6005       FORMAT(11X,9(I9,4X))
6010       FORMAT (2X,"Y(CM)X(CM)",2X,9(1PG9.2,4X))
#
#.....DUMP THE NUMBERS
           DO KY=NY1,NY2 [
                WRITE (LUPRT,7000) KY,YPOS(KY),(Z(K,KY),K=N1,N2)
7000            FORMAT(I2,1PG9.2,2X,9(1PG13.5))
           ]
      ]
#
#.....DONE
      RETURN
      END
```


File: &TRACE

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE TRACE (KGCB,KDCB,N,M,L,PLINE)
#
#     TRACE OUT A CONTOUR LINE
#        KGCB -> GRAPHIC CONTROL BLOCK
#        N    -> STARTING X-INDEX
#        M    -> STARTING Y-INDEX
#        LL     -> LINE INDEX : >0 -> STARTING AT Y-DIRECTION
#                               <0 -> STARTING AT X-DIRECTION
#        PLINE-> LINE VALUE
#
#         (NX,MY) = STARTING POSITION  -> FINAL POSITION
#         KTRC    = TRACING  FLAG:   1 -> 1ST SEGMENT
#                                    2 -> NOT 1ST SEGMENT
#         KDIR    = COMING   DIRECTION -> GOING DIRECTION
#                   0 -> END OF LINE
#                    IF (POTSI(NX,MY).EQ.PLINE) :
#                     1 = BETWEEN (NX+1,MY  ) AND (NX+1,MY-1)
#                     2 =         (NX+1,MY-1)     (NX  ,MY-1)
#                     3           (NX  ,MY-1)     (NX-1,MY-1)
#                     4           (NX-1,MY-1)     (NX-1,MY  )
#                     5           (NX-1,MY  )     (NX-1,MY+1)
#                     6           (NX-1,MY+1)     (NX  ,MY+1)
#                     7           (NX  ,MY+1)     (NX+1,MY+1)
#                     8           (NX+1,MY+1)     (NX+1,MY  )
#                    IF (POTSI(NX,MY).NE.PLINE) :
#                     1 = BETWEEN (NX,MY) AND (NX+1,MY  ), FROM +MY TO -MY  (A)
#                     2           (NX,MY)     (NX  ,MY-1),      +NX    -NX  (B)
#                     3           (NX,MY)     (NX-1,MY  ),      -MY    +MY  (C)
#                     4           (NX,MY)     (NX  ,MY+1),      -NX    +NX  (D)
#                     5           (NX,MY)     (NX+1,MY  ),      -MY    +MY  (E)
#                     6           (NX,MY)     (NX  ,MY+1),      +NX    -NX  (F)
#                     7           (NX,MY)     (NX-1,MY  ),      +MY    -MY  (G)
#                     8           (NX,MY)     (NX  ,MY-1),      -NX    +NX  (H)
#         (XP,YP) = COORDINATE
#
#        POTSI(N,M) = DATA ARRAY
#        XPOS (N)   = X-POSITION ARRAY, DELX(N) = X-MESH SIZE ARRAY
#        YPOS (M)   = Y-POSITION ARRAY, DELY(M) = Y-MESH SIZE ARRAY
#
#        1 <= N <= NXMAX,  1 <= M <= NYMAX,  1 <= LL <= NLMAX
#
*CALL TEMA
*CALL TCOMMN
#
#.....PLOTTING PARAMETERS
#
      DIMENSION KGCB(1), KDCB(1), LBUF (3)
      DIMENSION KSRCH(8),KN(9),KM(9)
#
      DATA KN/1, 1, 0,-1,-1,-1,0,1,1/,  KM/0,-1,-1,-1, 0, 1,1,1,0/
      DATA NSEC/0/, NCR/2HGG/, NAMX/2HX /, NAMY/2HY /
#
#.....GET FIRST SET OF INDICES
      NX=N
      MY=M
      KTRC=1
      LL=L
      IF (LL.GT.0) [ N1=N;   M1=M+1; KDIR=4; CALL LNMRK (N,M,LL) ]
      ELSE         [ N1=N+1; M1=M;   KDIR=5; CALL LNMRK (N,M,LL) ]
#
#.....MOVE TO THE FIRST POINT
      POTO=POTSI(NX,MY)
      XP=XPOS(NX)
      YP=YPOS(MY)
      IF (PLINE.NE.POTO) [
           POT1=POTSI(N1,M1)
           IF (POTO.NE.POT1)
                IF (NX.NE.N1) XP=XP+DELX(NX)*(PLINE-POTO)/(POT1-POTO)
                ELSE          YP=YP+DELY(MY)*(PLINE-POTO)/(POT1-POTO)
           ELSE IF (NX.NE.N1) XP=XP+0.5(EXP)0*DELX(NX)
                ELSE          YP=YP+0.5(EXP)0*DELY(MY)
      ]
      CALL MOVE (KGCB,XP,-YP)
      XPM=XP
      YPM=YP
#
```

```
#.....SEGMENT LOOP, AT MOST TWO SEGMENTS
      REPEAT [
#
#.....TRACING LOOP, TERMINATING AT KEND>0
           NPTS=1
           KEND=0
           REPEAT [
#
#.....LOAD SEARCHING INDEX
              DO K=1,8; KSRCH(K)=1
              IF (PLINE.EQ.POT0) [ KTYP=1; KSRCH(KDIR)=0 ]
              ELSE [
                 KTYP=2
                 IF (KDIR.EQ.1) [ DO K=4,8; KSRCH(K)=0; KSRCH(3)=2 ]
                 IF (KDIR.EQ.2) [ DO K=6,8; KSRCH(K)=0
                                  DO K=1,2; KSRCH(K)=0; KSRCH(5)=2 ]
                 IF (KDIR.EQ.3) [ DO K=1,4; KSRCH(K)=0
                                            KSRCH(8)=0; KSRCH(7)=2 ]
                 IF (KDIR.EQ.4) [ DO K=2,6; KSRCH(K)=0; KSRCH(1)=2 ]
                 IF (KDIR.EQ.5) [ DO K=1,5; KSRCH(K)=0; KSRCH(6)=3 ]
                 IF (KDIR.EQ.6) [ DO K=1,3; KSRCH(K)=0
                                  DO K=7,8; KSRCH(K)=0; KSRCH(4)=3 ]
                 IF (KDIR.EQ.7) [ DO K=5,8; KSRCH(K)=0
                                            KSRCH(1)=0; KSRCH(2)=3 ]
                 IF (KDIR.EQ.8) [ DO K=3,7; KSRCH(K)=0; KSRCH(8)=3 ]
              ]
#
#.....SEARCH FOR INTERSECTION
              KNN=1
              KLINE=0
              REPEAT [
                 IF (KSRCH(KNN).NE.0) [
                    NO=NX+KN(KNN)
                    MO=MY+KM(KNN)
                    N1=NX+KN(KNN+1)
                    M1=MY+KM(KNN+1)
                    IF (KSRCH(KNN).EQ.2) [ N1=NX; M1=MY ]
                    IF (KSRCH(KNN).EQ.3) [ NO=NX; MO=MY ]
                    IF ((NO.GE.1).AND.(NO.LE.NXMAX).AND.^
                        (N1.GE.1).AND.(N1.LE.NXMAX).AND.^
                        (MO.GE.1).AND.(MO.LE.NYMAX).AND.^
                        (M1.GE.1).AND.(M1.LE.NYMAX))     [
                        POT0=POTSI(NO,MO)
                        POT1=POTSI(N1,M1)
                        SIGN0=SIGN(1.0(EXP)0,POT0-PLINE)
                        SIGN1=SIGN(1.0(EXP)0,POT1-PLINE)
                        IF ((SIGN0.NE.SIGN1).OR.(PLINE.EQ.POT0)) KLINE=1
                        ELSE                                     KNN=KNN+1
                    ]
                    ELSE                                         KNN=KNN+1
                    IF (KLINE.EQ.0)                              KNN1=KNN-1
                    ELSE                                         KNN1=KNN
                 ]
                 ELSE                                            KNN=KNN+1
              ] UNTIL ((KNN.GT.8).OR.(KLINE.EQ.1))
#
#.....INTERPOLATE COORDINATES
              IF (KLINE.EQ.1) [
                 XP=XPOS(NO)
                 YP=YPOS(MO)
                 XP1=XPOS(N1)
                 YP1=YPOS(M1)
                 IF (POT0.NE.POT1) [
                    IF (NO.NE.N1) XP=XP+(XP1-XP)*(PLINE-POT0)/(POT1-POT0)
                    IF (MO.NE.M1) YP=YP+(YP1-YP)*(PLINE-POT0)/(POT1-POT0)
                 ]
                 ELSE IF (PLINE.NE.POT0) [
                    XP=0.5(EXP)0*(XP+XP1)
                    YP=0.5(EXP)0*(YP+YP1)
                 ]
#
#.....UPDATE LINE INDEX AND CHECK THE END OF LINE
                 IF ((NO.EQ.NX).AND.(MO.EQ.MY)) [
                    NT=N1; MT=M1; PT=P1; N1=NO; M1=MO; POT1=POT0
                                         NO=NT; MO=MT; POT0=PT
                 ]
                 KLX=-IABS(LL)
                 KLY=-KLX
                 IF (NO.EQ.N1)
                    IF (MO.LT.M1) [
```



```
                       KBITY=LNCHK(NO,MO,KLY)
                       IF (KBITY.EQ.0) CALL LNMRK (NO,MO,KLY)
                       ELSE                 KEND=1
                    ]
                    ELSE [
                       KBITY=LNCHK(N1,M1,KLY)
                       IF (KBITY.EQ.0) CALL LNMRK (N1,M1,KLY)
                       ELSE                 KEND=1
                    ]
                 ELSE
                    IF (NO.LT.N1) [
                       KBITX=LNCHK(NO,MO,KLX)
                       IF (KBITX.EQ.0) CALL LNMRK (NO,MO,KLX)
                       ELSE                 KEND=1
                    ]
                    ELSE [
                       KBITX=LNCHK(N1,M1,KLX)
                       IF (KBITX EQ.0) CALL LNMRK (N1,M1,KLX)
                       ELSE                 KEND=1
                    ]
#
#.....UPDATE DIRECTION INDEX
                 IF (KEND.EQ.0) [
                    KDIR0=KDIR
                    IF (POT0.NE.PLINE)
                       IF ((N1.NE.NX).OR.(M1.NE.MY))
                          IF (NO.EQ.N1) IF (MO.GT.M1) IF (NO.GT.NX) KDIR=8
                                                      ELSE          KDIR=2
                                        ELSE          IF (NO.GT.NX) KDIR=4
                                                      ELSE          KDIR=6
                          ELSE          IF (NO.GT.N1) IF (MO.GT.MY) KDIR=3
                                                      ELSE          KDIR=7
                                        ELSE          IF (MO.GT.MY) KDIR=5
                                                      ELSE          KDIR=1
                       ELSE [
                          IF (KDIR0.EQ.1) KDIR=6
                          IF (KDIR0.EQ.6) KDIR=1
                          IF (KDIR0.EQ.2) KDIR=5
                          IF (KDIR0.EQ.5) KDIR=2
                          IF (KDIR0.EQ.3) KDIR=8
                          IF (KDIR0.EQ.8) KDIR=3
                          IF (KDIR0.EQ.4) KDIR=7
                          IF (KDIR0.EQ.7) KDIR=4
                       ]
#
#.........CASE A
                    ELSE [
                       IF (KTYP.EQ.1)
                          IF      (NX.EQ.NO) IF (MO.GT.MY) KDIR=2
                                             ELSE          KDIR=6
                          ELSE IF (NX.GT.NO) IF (MO.GT.MY) KDIR=1
                                             ELSE          KDIR=8
                               ELSE          IF (MO.GT.MY) KDIR=4
                                             ELSE          KDIR=5
#
#.........CASE B
                       ELSE IF (NO.GT.NX) [
                            IF      (MO.GT.MY) IF (KDIR0.EQ.4) KDIR=4
                                               ELSE            KDIR=3
                            ELSE IF (MO.EQ.MY) IF (KDIR0.EQ.4) KDIR=5
                                               ELSE            KDIR=4
                                 ELSE          IF (KDIR0.EQ.1) KDIR=6
                                               ELSE            KDIR=5
                          ]
                          ELSE IF (NO.EQ.NX) [
                               IF (MO.GT.MY) IF (KDIR0.EQ.3) KDIR=3
                                             ELSE            KDIR=2
                               ELSE          IF (KDIR0.EQ.1) KDIR=7
                                             ELSE            KDIR=6
                          ]
                            ELSE [
                             IF      (MO.GT.MY) IF (KDIR0.EQ.3) KDIR=2
                                                ELSE            KDIR=1
                             ELSE IF (MO.EQ.MY) IF (KDIR0.EQ.2) KDIR=1
                                                ELSE            KDIR=8
                                  ELSE          IF (KDIR0.EQ.2) KDIR=8
                                                ELSE            KDIR=7
                            ]
                    ]
#
```

File: &TRACE

```
#.....LOAD INDICES AND PLOT THE POINT
                    NX=NO
                    NY=NO
                 ]
                 NPTS=NPTS+1
                 CALL DRAW (KGCB,XP,-YP)
              ]
#
#.....END OF LINE
              ELSE KEND=2
          ] UNTIL (KEND.NE.0)
#
#.....LABLE
          IF (KTRC.EQ.1) [
              CALL CODE; WRITE (LBUF,1000) PLINE
1000          FORMAT (F6.2)
              CALL LORG (KGCB,9)
              CALL GFONT (KGCB,6HFONT2 ,NSEC,NCR,KDCB)
              NS=1
              NC=6
              CALL GTEXT (KGCB,LBUF,NS,NC,KDCB)
              CALL GFONT (KGCB,0,0,0,KDCB)
          ]
#
#.....END OF 1ST SEGMENT
          IF ((KTRC.EQ.1).AND.(KEND.EQ.2).AND.(L.LT.0)) [
              CALL MOVE (KGCB,XPN,-YPN)
              NX=N
              NY=N
              LL=-L
              KDIR=1
          ]
          ELSE KTRC=KTRC+1
#
#.....END OF 2ND SEGMENT
          KTRC=KTRC+1
      ] UNTIL (KTRC.GT.2)
      CALL PENUP (KGCB)
#
#.....DONE
      RETURN
      END
```


File: &TSCAL

```
*CALL TWDEFN
      SUBROUTINE TSCAL (KGCB,XONY,CHARN,CHITE,XLENG,YLENG)
#
#     SCALE CHARACTER SIZE
#
*CALL TCOMNN
      DIMENSION KGCB(192)
#
      DATA ASPEC/0.7<EXP>0/, SLANT/0.0<EXP>0/
#
#.... CHECK PLOTTING SURFACE
      IF (XONY.GT.1.0<EXP>0) [
          UMRGN=100.0<EXP>0/6.0<EXP>0
          DMRGN=UMRGN/0.6<EXP>0
          BMRGN=XONY*20.0<EXP>0
          FMRGN=BMRGN
          YLENG=100.0<EXP>0-UMRGN-DMRGN
          XLENG=XONY*100.0<EXP>0-BMRGN-FMRGN
      ]
      ELSE [
          FMRGN=100.0<EXP>0/6.0<EXP>0
          BMRGN=FMRGN/0.6<EXP>0
          UMRGN=20.0<EXP>0/XONY
          DMRGN=UMRGN
          YLENG=100.0<EXP>0/XONY-UMRGN-DMRGN
          XLENG=100.0<EXP>0-BMRGN-FMRGN
      ]
#
#.....ESTIMATE CHARACTER SIZE
      CX=XLENG/CHARN
      CY=YLENG/CHARN
      CHITE=AMIN1(CX,CY)
      CWIDTH=ASPEC*CHITE
      CWIDTH=AMIN1(CWIDTH,UMRGN*0.25<EXP>0,BMRGN*0.25<EXP>0)
      CHITE=CWIDTH/ASPEC
      CALL CSIZE (KGCB,CHITE,ASPEC,SLANT,0)
#
#.....DONE
      RETURN
      END
```

File: &WBASE

```
*CALL TWDEFN
*IF HP1000
EMA (XYZ,0)
*ENDIF
      SUBROUTINE WBASE (N,M,NB1,NB2)
#
#     DETERMINE BARRIER WIDTH
#
*CALL TEMA
*CALL TCOMMN
#
      POTB=POTSI(N,M)
      TWOVT=VT300K+VT300K
#
#.....FIND LEFT LIMIT OF THE BARRIER
      POTX=POTSI(N-1,M)
      NB1=N
      WHILE ((NB1.GT.NSOURC).AND.((POTX-POTB).LE.TWOVT)) [
          NB1=NB1-1
          POTX=POTSI(NB1-1,M)
      ]
#
#.....FIND RIGHT LIMIT OF THE BARRIER
      POTX=POTSI(N+1,M)
      NB2=N
      WHILE ((NB2.LT.NDRAIN).AND.((POTX-POTB).LE.TWOVT)) [
          NB2=NB2+1
          POTX=POTSI(NB2+1,M)
      ]
#
#.....DONE
      RETURN
      END
```


File: &WDEPL

```
*CALL TWDEFN
      SUBROUTINE WDEPL (WS0,WS1,WD0,WD1)
#
#     DETERMINE LATERAL SPANS OF SURFACE DEPLETION REGIONS
#
*CALL TCOMMN
#
      DATA A0/0.0631353(EXP)0/, A1/0.8013292(EXP)0/, A2/-0.01110777(EXP)0/
#
#.....GET JUNCTION DEPTH
      IF (ABS(DOSE(3)).GT.1.0(EXP)0) [
          XJ=WSURF(3)
          ALPHAX=1.0(EXP)0/(ALPHB+ALPHA(3))
          PHIJ0=PHIJ
#
#.....DEPLETION MOSFET
          KDEPL=0
          IF ((ABS(CSURF(1)).GT.0.0(EXP)0).AND.^
              (SIGN(1.0(EXP)0,CSURF(1)).NE.TYPE)) [
#              IF ((ABS(CSURF(1)).GT.1.0(EXP)0).AND.^
#                 (ABS(CSURF(3)).GT.1.0(EXP)0))
              IF (ABS(CSURF(3)).GT.1.0(EXP)0)
                  PHIJ0=VT300K*ABS(ALOG(ABS((SIGN(CSUB,TYPE)+CSURF(3))/^
                                            (SIGN(CSUB,TYPE)+CSURF(1)))))
              KDEPL=1
              PION2=2.0(EXP)0*ATAN2(1.0(EXP)0,1.0(EXP)0)
              TOXSI=TOX0*EPSI/EPSIO2
              TOXSI2=TOXSI*TOXSI
              TOX2=TOX0*TOX0
              W0=PION2*(SQRT(XJ*XJ+TOX2)-TOX0)
          ]
#
#.....DETERMINE WIDTH OF SURFACE DEPLETION REGION
          WS1=SQRT(ALPHAX*ABS(VSB+PHIJ))/XJ
          WS1=XJ*(A0+A1*WS1+A2*WS1*WS1)
          WD1=SQRT(ALPHAX*ABS(VDB+PHIJ))/XJ
          WD1=XJ*(A0+A1*WD1+A2*WD1*WD1)
          WS0=WS1*ABS((SIGN(CSUB,TYPE)+CSURF(1))/(SIGN(CSUB,TYPE)+CSURF(3)))
          WD0=WD1*ABS((SIGN(CSUB,TYPE)+CSURF(1))/(SIGN(CSUB,TYPE)+CSURF(3)))
          IF (KDEPL.EQ.1) [
              IF ((VSB+PHIJ0-VGB).GT.0.0(EXP)0) [
                  WL=AMIN1(W0,SQRT(ALPHAX*(VSB+PHIJ0-VGB)+TOXSI2)-TOXSI)
                  VC=WL*WL/ALPHAX
                  VSCO=WL*(TOXSI+TOXSI)/ALPHAX
                  VSCOT=VC+VSCO
                  EOX=((VSB+PHIJ0-VGB)-VSCOT)/(TOX0+WL*EPSIO2/EPSI)
                  PHIS=VGB+EOX*TOX0+VSCO
                  W2=WL/PION2+TOX0
                  WS1=SQRT(W2*W2-TOX2)
                  WS0=EPSIO2*EOX/(Q*ABS(CSTEP(3)))
              ]
              IF ((VDB+PHIJ0-VGB).GT.0.0(EXP)0) [
                  WL=AMIN1(W0,SQRT(ALPHAX*(VDB+PHIJ0-VGB)+TOXSI2)-TOXSI)
                  VC=WL*WL/ALPHAX
                  VSCO=WL*(TOXSI+TOXSI)/ALPHAX
                  VSCOT=VC+VSCO
                  EOX=((VDB+PHIJ0-VGB)-VSCOT)/(TOX0+WL*EPSIO2/EPSI)
                  PHID=VGB+EOX*TOX0+VSCO
                  W2=WL/PION2+TOX0
                  WD1=SQRT(W2*W2-TOX2)
                  WD0=EPSIO2*EOX/(Q*ABS(CSTEP(3)))
              ]
          ]
      ]
      ELSE [
          WS1=SQRT(ALPHB*ABS(VSB+PHIB+PHIB))
          WD1=SQRT(ALPHB*ABS(VDB+PHIB+PHIB))
          WS0=WS1*ABS((SIGN(CSUB,TYPE)+CSURF(1))/(SIGN(CSUB,TYPE)+CSURF(3)))
          WD0=WD1*ABS((SIGN(CSUB,TYPE)+CSURF(1))/(SIGN(CSUB,TYPE)+CSURF(3)))
      ]
#
#.....DONE
      WS0=AMAX1(WS0,0.0(EXP)0)
      WS1=AMAX1(WS1,0.0(EXP)0)
      WD0=AMAX1(WD0,0.0(EXP)0)
      WD1=AMAX1(WD1,0.0(EXP)0)
      RETURN
      END
```

REFERENCES

[1] J.E.Lilienfeld, U.S. Patent No. 1,745,175, 1930.

[2] O.Heil, British Patent No. 439,457, 1935.

[3] W.Shockley and G.L.Pearson, "Modulation of Conductance of Thin Films of Semiconductors by Surface Charges," *Physical Review,* Vol.74, 1948, pp.232-233.

[4] D.Kahng and M.M.Atalla, "Silicon-Silicon Dioxide Field Induced Surface Devices," IRE Solid-State Device Research Conference, Carnegie Inst. of Tech., Pittsburgh, Penn., 1960.

[5] P.R.Gray and R.G.Meyer, *Analysis and Design of Analog Integrated Circuits,* Chapter 2,John Wiley & Sons, 1977.

[6] E.S.Schlegel, "A Bibliography of Metal-Insulator-Semiconductor Studies," *IEEE Trans. Electron Devices,* Vol.ED-14, No.11, Nov. 1967, pp.728-749.

[7] E.S.Schlegel, "Additional Bibliography of Metal-Insulator-Semiconductor Studies," *IEEE Trans. Electron Devices,* Vol.ED-15, No.12, Dec. 1968, pp.951-953.

[8] W.Shockley, U.S. Patent No. 2,787,564, 1954.

[9] J.F.Gibbons, "Ion Implantation in Semiconductors-Part I, Range Distribution Theory and Experiments," *Proc. IEEE,* Vol.56, No.3, March 1968, pp.295-319.

[10] J.F.Gibbons, "Ion Implantation in Semiconductors-Part II, Damage, Production and Annealing," *Proc. IEEE*, Vol.60, No.9, Sept. 1972, pp.1062-1095.

[11] L.W.Nagel, "SPICE2, A Computer Program to Simulate Semiconductor Circuits," *ERL Memorandum No.ERL-M520*, University of California, Berkeley, May 1975.

[12] S.P.Fan, M.Y.Hsueh, A.R.Newton and D.O.Pederson, "MOTIS-C: A New Circuit Simulation for MOS LSI Circuits," *Proceedings of IEEE International Symposium, Circuits and Systems*, April 1977, pp.700-703.

[13] A.R.Newton, "The Simulation of Large-Scale Integrated Circuits," *ERL Memorandum No.UCB/ERL-M78/52*, University of California, Berkeley, July 1978.

[14] E.Khalily, "T.E.C.A.P.: A Automated Characterization System," *Technical Report No.5017-1*, Stanford Electronics Laboratories, Stanford University, March 1979.

[15] A.Vladimirescu and S.Liu, "The Simulation of MOS Integrated Circuits Using SPICE2," *ERL Memorandum No. UCB/ERL M80/7*, University of California, Berkeley, Feb. 1980 (rev. Oct. 1980).

[16] H.Shichman and D.A.Hodges, "Modeling and Simulation of Insulated-Gate Field-Effect Transistor Switching Circuits," *IEEE J. Solid-State Circuits*, Vol.SC-3, No.5, Sept. 1968, pp.285-289.

[17] G.Baccarani, M.Rudan and G.Spadini, "Analytical IGFET Model Including Drift and Diffusion Currents," *Solid-State and Electron Devices*, Vol.2, No.2, March 1978, pp.62-68.

[18] M.B.Barron, "Computer Aided Analysis of Insulated Gate Field Effect Transistors," *Technical Report No.5501-1,* Stanford Electronics Laboratories, Stanford University, Nov. 1969.

[19] M.S.Mock, "A Two-Dimensional Mathematical Model of the Insulated-Gate Field-Effect Transistors," *Solid-State Electronics,* Vol.16, 1973, pp.601-609.

[20] A.D.Sutherland, "A Two-Dimensional Computer Model for the Steady State Operation of MOSFET's," *Research and Development Technical Report, ECOM-75-1344-F Supplement,* Advanced Research Project Agency, Sept. 1977.

[21] Y.A.El-Mansy and A.R.Boothroyd, "A New Approach to the Theory and Modeling of Insulated-Gate Field-Effect Transistors," *IEEE Trans. Electron Devices,* Vol.ED-24, No.3, March 1977, pp.241-253.

[22] Y.A.El-Mansy and A.R.Boothroyd, "A Simple Two-dimensional Model for IGFET Operation in the Saturation Region," ibid, pp.254-262.

[23] J.R.Brews, "Comments on 'A New Approach to the Theory and Modeling of IGFET's'," *IEEE Trans. Electron Devices,* Vol.ED-24, No.12, Dec. 1977, pp.1369-1370.

[24] Y.A.El-Mansy, *Modelling of Insulated-Gate Field-Effect Transistors,* Ph.D. Dissertation, Carleton University, Ottawa, Canada, Nov. 1974.

[25] Y.A.El-Mansy and A.R.Boothroyd, "MOS Device Modelling," preprint.

[26] H.C.Pao and C.T.Sah, "Effects of Diffusion Current on Characteristics of Metal-Oxide (Insulator) Semiconductor Transistors," *Solid-State Electronics,* Vol.9, 1966, pp.927-937.

[27] A.P.Gnädinger and H.E.Talley, "Quantum Mechanical Calculation of the Carrier Distribution and the Thickness of the Inversion Layer of a MOS Field-Effect Transistor," *Solid-State Electronics,* Vol.13, 1970, pp.1301-1309.

[28] F.Stern, "Self-Consistent Results for N-type Si Inversion Layers," *Physical Review B,* Vol.5, No.12, 15 June 1972, pp.4891-4899.

[29] J.A.Pals, "Experimental Verification of the Surface Quantization of an N-Type Inversion Layer of Silicon at 300 and $77°K$," *Physical Review B,* Vol.5, No.10, 15 May 1972, pp.4208-4210.

[30] K.von Klitzing, G.Landwehr and G.Dorda, "Shubnikov de Haas Oscillations in P-Type Inversion Layers on N-Type Silicon," *Solid-State Communications,* Vol.14, No.5, 1974, pp.387-393.

[31] S.M.Sze, *Physics of Semiconductor Devices,* John Wiley & Sons, 1969.

[32] R.Seiwatz and M.Green, "Space Charge Calculations for Semiconductors," *J. Applied Physics,* Vol.29, No.7, July 1958, pp.1034-1040.

[33] T.I.Kamins and R.S.Muller, "Statistical Considerations in MOSFET Calculations," *Solid-State Electronics,* Vol.10, 1967, pp.423-431.

[34] J.McDougall and E.C.Stoner, "The Computation of Fermi-Dirac Functions," *Trans. Royal Society,* Vol.A237, 7 Feb. 1938, pp.67-104.

[35] M.J.McNutt and C.T.Sah, "High Frequency Space Charge Layer Capacitance of Strongly Inverted Semiconductor Surfaces," *Solid-State Electronics,* Vol.17, 1974, pp.377-385.

[36] R.M.Swanson and J.D.Meindl, "Ion-Implanted Complementary MOS Transistors in Low-Voltage Circuits," *IEEE J. Solid-State Circuits,*

Vol.SC-7, No.2, April 1972, pp.146-153.

[37] M.C.Tobey Jr. and N.Gordon, "Concerning the Onset of Heavy Inversion in MIS Devices," *IEEE Trans. Electron Devices,* Vol.ED-21, No.10, Oct. 1974, pp.649-650.

[38] R.J.van Overstraeten, G.Declerck and G.L.Broux, "Inadequacy of the Classical Theory of the MOS Transistor Operating in Weak Inversion," *IEEE Trans. Electron Devices,* Vol.ED-20, No.12, Dec. 1973, pp.1150-1153.

[39] S.G.Davison and J.D.Levine, "Surface States," *Solid-State Physics, Advances in Research and Applications,* Ed. H.Ehrenreich, F.Seitz and D.Turnbull, Academic Press, Vol.25, 1970.

[40] J.Koomen, "Investigation of the MOST Channel Conductance in Weak Inversion," *Solid-State Electronics,* Vol.16, 1973, pp.801-810.

[41] D.A.Antoniadis, S.E.Hansen and R.W.Dutton, "SUPREM-II: A Program for IC process modelling and simulation," *Technical Report SEL 78-020,* Stanford Electronics Laboratories, Stanford University, June 1978.

[42] W.G.Oldham, S.N.Nandgaonkar, A.R.Neureuther and M.O'Toole, "A General Simulator for VLSI Lithography and Etching Process: Part I-Application to Projection Lithography," *IEEE Trans. Electron Devices,* Vol.ED-26, No.4, April 1979, pp.717-722.

[43] T.Toyabe, K.Yamaguchi, S.Asai and M.S.Mock, "A Numerical Model of Avalanche Breakdown in MOSFET's," *IEEE Trans. Electron Devices,* Vol.ED-25, No.7, July 1978, pp.825-832.

[44] D.P.Kennedy and R.R.O'Brien, "Analysis of the Impurity Atom Distribution Near the Diffusion Mask for a Planar p-n Junction," *IBM*

Journal, Vol.9, May 1965, pp.179-186.

[45] H.G.Lee, J.D.Sansbury, R.W.Dutton and J.L.Moll, "Modeling and Measurement of Surface Impurity Profiles of Laterally Diffused Regions," *IEEE J. Solid-State Circuits,* Vol.SC-13, No.4, August 1978, pp.455-461.

[46] C.Hastings, Jr., *Approximation for Digital Computers,* Princeton University Press, Princeton, N.J., 1955.

[47] S.Liu, B.Hoefflinger and D.O.Pederson, "Interactive Two-Dimensional Design of Barrier-Controlled MOS Transistors," *IEEE Trans. Electron Devices,* Vol.ED-27, No.8, August 1980, pp.1550-1558.

[48] L.M.Dang, "A Simple Current Model for Short-Channel IGFET and Its Application to Circuit Simulation," *IEEE J. Solid-State Circuits,* Vol.SC-14, No.2, April 1979, pp.358-367.

[49] R.H.Dennard, F.H.Gaensslen, H.N.Yu, V.L.Rideout, E.B.Bassous and A.R.LeBlanc, "Design of Ion-Implanted MOSFET's with Very Small Physical Dimensions," *IEEE J. Solid-State Circuits,* Vol.SC-9, No.5, Oct. 1974, pp.256-268.

[50] T.Nakamura, M.Yamamoto, H.Ishikawa and M.Shinoda, "Submicron Channel MOSFET's Logic Under Punchthrough," *IEEE J. Solid-State Circuits,* Vol.SC-13, No.5, October 1978, pp.572-577.

[51] L.D.Yau, "A Simple Theory to Predict the Threshold Voltage of Short-Channel IGFET's," *Solid-State Electronics,* Vol.17, 1974, pp.1059-1063.

[52] H.S.Lee, "An Analysis of the Threshold Voltage for Short Channel IGFET's," *Solid-State Electronics,* Vol.16, 1973, pp.1407-1417.

[53] D.E.Ward and R.W.Dutton, "A Charge-Oriented Model for MOS Transistor Capacitances," *IEEE J. Solid-State Circuits,* Vol.SC-13, No.5, Oct. 1978, pp.703-708.

[54] H.Masuda, M.Nakai and M.Kubo, "Characteristics and Limitation of Scaled-Down MOSFET's Due to Two-Dimensional Field Effect," *IEEE Trans. Electron Devices,* Vol.ED-26, No.6, June 1979, pp.980-986.

[55] G.Merckel, J.Borei and N.Z.Cupcea, "An Accurate Large-Signal MOS Transistor Model for Use in Computer-Aided Design," *IEEE Trans. Electron Devices,* Vol.ED-19, No.5, May 1972, pp.681-690.

[56] R.S.Muller and T.I.Kamins, *Device Electronics for Integrated Circuits,* John Wiley & Sons, 1977.

[57] J.Bardeen and W.Shockley, "Deformation Potentials and Mobilities in Non-polar Crystal," *Physical Review,* Vol.80, 1950, pp.72-80.

[58] E.Conwell and V.F.Weisskopf, "Theory of Impurity Scattering in Semiconductors," *Physical Review,* Vol.77, 1950, pp.388-390.

[59] Y.C.Cheng and E.A.Sullivan, "On the Role of Scattering by Surface Roughness in Silicon Inversion Layers," *Surface Science,* Vol.34, 1973, pp.717-731.

[60] A.R.Nelson and E.Brown, "Conduction in Quantized Silicon Inversion Layers with Many Occupied Subbands," *Physical Review B,* Vol.9, Feb. 1974, pp.1664-1668.

[61] D.R.Alexander, R.J.Antinone and G.W.Brown, *SPICE2 MOS Modeling Handbook,* The BDM Corporation, BDM/A-77-071-TR, May 1977.

[62] P.E.Cottrell, R.R.Troutman and T.H.Ning, "Hot Electron Emission in N-Channel IGFET's," *IEEE J. Solid-State Circuits,* Vol.SC-14, April 1979, pp.442-455.

[63] G.T.Wright, "Current/Voltage characteristics, Channel Pinchoff, and Field-Effect Transistor," *Electronics Letters,* Vol.6, Feb. 1970, pp.107-109.

[64] B.T.Murphy, "Unified Field Effect Transistor Theory Including Velocity Saturation," *IEEE J. Solid-State Circuits,* Vol.SC-15, June 1980, pp.325-328.

[65] R.C.Foss, R.Hartland and J.Roberts, "An MOS Transistor Model for a Micro-Mini Computer Based Circuit Analysis System," *IEE Fifth European Solid-State Circuits Conference,* 1979, pp.56-57.

[66] G.Baum and H.Beneking, "Drift Velocity Saturation in MOS Transistors," *IEEE Trans. Electron Devices,* Vol.ED-17, No.6, June 1972, pp.481-482.