NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

ISBN 0-315-56336-2

ANALYSIS AND DESIGN OF A GaAs FET FREQUENCY DOUBLER IN MICROSTRIP AND E-PLANE TECHNOLOGIES

by

Sara Meszaros, B.A., B.A.Sc.

A thesis submitted to the

School of Graduate Studies and Research

in partial fulfillment of the requirements

for the degree of

Master of Applied Science

Ottawa-Carleton Institute for Electrical Engineering

Department of Electrical Engineering

Faculty of Engineering

University of Ottawa

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

ACKNOWLEDGEMENTS

I am pleased to have the opportunity to express my appreciation to my colleagues at the Communications Research Centre (C.R.C.) for their encouragement and assistance in the course of this work. I would particularly like to thank Elizabeth Bala for her fine work on the fabrication of the circuits and Kees Verver for the benefit of his experience with finline measurements.

I am also most grateful to Dr. W.J.R. Hoefer for the advice and assistance received during the preparation of this thesis, and to faculty members of the Institute for their guidance and instruction over the years.

Finally, special thanks go to my husband Peter for his support and encouragement, which really made this effort possible.

ABSTRACT

This thesis presents the author's work on the analysis and design of FET frequency doublers, including the implementation of a FET model for large-signal applications and a nonlinear circuit analysis program based on the modified harmonic balance method.

In addition, the first reported E-plane frequency doublers are described and their performance data presented. The author's program for spectral domain analysis of unilateral finline in included in an appendix. The doubler was developed at the Communications Research Centre (C.R.C.) and was intended to be part of a local oscillator unit pumping a 44/7.5 GHz finline downconverter. It converts an input signal at f_o = 18.75 GHz to 37.5 GHz with a conversion loss of 5.8 dB, producing an output power of 8.5 dBm. The purpose of this design was to demonstrate the praticability of such planar technology for the development of low-cost, reliable components in the millimeter-wave range.

Contributions of this Thesis

There are two principal original contributions made in this thesis. Firstly, in the field of FET frequency multipliers, two examples have been demonstrated in planar technology, being the first designs reported in the literature to make use of finline for this application.

The second contribution, in the more general field of nonlinear device modeling, is the implementation of an integrated software package for the automated small-signal measurement and modeling of GaAsFETs for large-signal applications. This software, (PARFET,EXTPAR,INTPAR) provides the user with a simple and quick means of characterizing a device over its entire operating range. Since the bias-dependency of the resultant parameters is expressed in closed-form, the device model can be inserted into nonlinear analysis programs such as Libra™, providing a valuable supplement to built-in models, which are usually based on a single commercial device. As an illustration of this, the model is demonstrated in a harmonic balance program written by the author. This program was intended to illustrate various principles of operation of the GaAsFET frequency multiplier, including the effects of bias and harmonic terminations on operating efficiency and the relative contributions of the various sources of nonlinearity.

TABLE OF CONTENTS

5. DESIGN AND REALIZATION OF A GaAsFET FREQUENCY DOUBLER

6. MEASUREMENT AND PERFORMANCE OF THE FREQUENCY MULTIPLIERS

7. CONCLUSIONS

APPENDIX I: FET MODEL PROGRAMS

APPENDIX II: HARMONIC BALANCE PROGRAM

APPENDIX III: UNILATERAL FINLINE ANALYSIS PROGRAM

List of Figures

List of Figures (Continued)

1. INTRODUCTION

1.0 Introduction

This thesis describes the first reported effort to realize a GaAsFET frequency doubler in E-plane technology. During the course of this work, various aspects involved in the use of FETs in nonlinear applications were explored in some detail, including device modeling, nonlinear analysis and device-circuit interactions.

The Gallium Arsenide Field Effect Transistor (GaAsFET), continues to be the most widely-used active device available for solid-state microwave applications. Therefore, since its behaviour is relatively complex, there is a need for models (particularly for nonlinear operation) which are easily obtained, accurate and simple to implement during the design process. Chapter 2 reviews the principles of operation of this device, and presents a technique for deriving a large-signal FET model directly from S-parameter measurements made over a range of bias points in detail. Results for the NEC67300 GaAsFET used in the subsequent design are shown, as well as results for in-house devices.

A frequency multiplier is only one example of a nonlinear circuit. Analysis of such circuits (which include mixers, power amplifiers and oscillators) is fairly complex, particularly in the microwave range, and for practical design purposes requires the use of numerical simulations. The wide-spread interest in obtaining nonlinear design tools is reflected in the recent appearance of

commercially available packages such as Libra™ and Harmonica™. Both packages are based on the Modified Harmonic Balance analysis technique (MHB), a simulation method which represents the best compromise in terms of speed and flexibility. Chapter 3 presents a detailed description of the author's implementation of the MHB method for the frequency doubler problem.

Chapter 4 describes the theory of frequency multiplication in GaAsFETs. The relative contributions of the various sources of nonlinearity are described, as well as the conditions under which they may be optimized. Supporting results from the simulation programs outlined in Chapters 2 and 3 are also presented.

The frequency doublers described in Chapter 5 were developed as part of a study undertaken at the Center for Research in Communications (C.R.C., Department of Communications, Shirley's Bay, Ottawa), into the potential of E-plane components at millimeter-wave applications. Two versions were built and tested: one in unilateral finline and one in a combination of antipodal finline and microstrip. It was intended to form part of the local oscillator unit driving a 44/7.5 GHz finline downconverter. A comparison of the two approaches gave some insight into the usefulness of finline for applications involving 2-port devices.

Chapter 6 contains the measured results for both doublers: the microstrip/finline doubler achieved significantly better conversion efficiency, bandwidth and output power than the all-finline version.

Program listings for the FET model parameter extraction (Chapter 2),

modified harmonic balance analysis (Chapter 3) and unilateral finline analysis (Chapter 5) are included in the Appendices I, II and III respectively. These programs were written by the author for use on an IBM-PC AT or compatible and are implemented in Pascal (Borland Turbo-Pascal version 3.0).

In summary, in this thesis the author presents three contributions to the design and analysis of FET frequency doublers:

1) Two novel frequency doublers have been realized in E-plane technique.

2) A technique for measuring and characterizing FET nonlinearities been implemented using the HP8510 network analyzer, HP310 computer and IBM-PC microcomputer. The resulting model is applicable in many large-signal applications, and may be used in conjunction with other nonlinear software packages for the design of oscillators, mixers, power amplifiers, etc

3) A harmonic balance analysis program was written to simulate the behaviour of frequency doublers under various bias and load conditions. This provides a simple but useful means for understanding the operation of GaAsFET doublers.

2. MODELING AND NONLINEAR EFFECTS OF THE GALLIUM ARSENIDE FIELD EFFECT TRANSISTOR

2.0 Introduction

The Gallium Arsenide Field Effect Transistor, (GaAsFET), was introduced for microwave applications in the early seventies, [2.1 -2.3]. The significant advantages it offers over other solid-state devices in terms of gain, isolation and noise performance have resulted in the GaAsFET becoming the dominant two-port microwave device in use today. Improved fabrication and modeling techniques have encouraged its use in a variety of applications in digital and analog microwave signal processing, including amplifiers, mixers, oscillators and frequency multipliers and dividers. The good isolation, gain and power output capabilities of the FET, as compared to the diode, have resulted in it gradually replacing that device at up to millimeterwave frequencies, as fabrication improves.

This chapter provides a brief explanation of GaAsFET operation, describing in detail the particular large-signal model chosen for application in this thesis.. This model is obtained directly from measured S-parameters without need of optimization routines or lengthy DC characterization. The principal sources of nonlinearity in GaAsFETs are illustrated using results calculated for the NEC673 device using the programs listed in Appendix I.

2.1 Device Operation

The field-effect transistor, whether gallium arsenide or silicon, is a unipolar device in which the current flow between two of the terminals (source and drain) is controlled by a voltage applied to the third (gate). Figure 2.1 shows a schematic cross-section of a simplified device. Note the presence of a semi-insulating (SI) layer which restricts current flow to a well-defined channel region of maximum depth a. One of the advantages of GaAs as opposed to Si is that high-quality SI GaAs is easily grown, permitting use of the planar construction shown here, which is both convenient for discrete devices and integrable into monolithic integrated circuits (MMICs). If such a layer is not present, device isolation is significantly degraded, as it is difficult to control the active channel depth precisely.

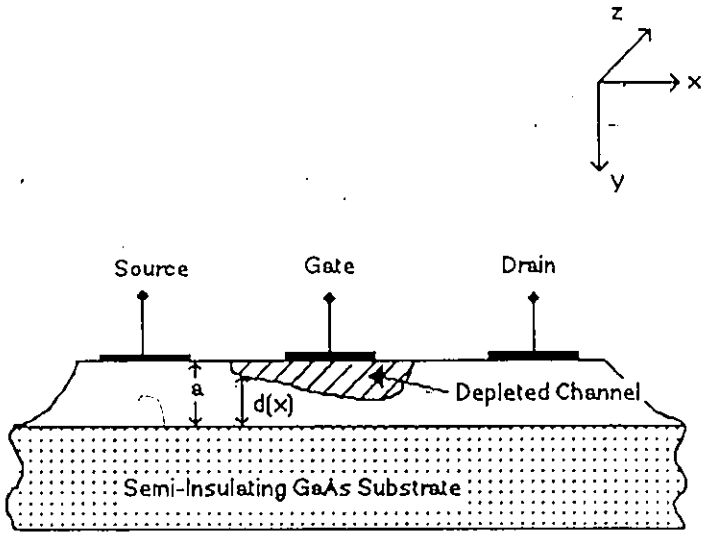


Figure 2.1: Schematic Cross-Section of GaAsFET

In order to reach a qualitative understanding of how this device operates, the following simplifying assumptions can be made:

i) channel doping is uniform, with carrier concentration N_D, and there are no free carriers in the SI layer

ii) the transition from depleted to active regions in the channel is abrupt

iii) the longitudinal electric field E_x is much less than the transverse component E_y in the depleted channel region and $E_x >> E_y$ in the active region. (This 'gradual channel approximation' is not strictly valid for short-gate devices.)

iv) the velocity-field characteristic for gallium arsenide is approximated by a piece-wise linear function, although in fact it is much more complex, as suggested in Figure 2.2.

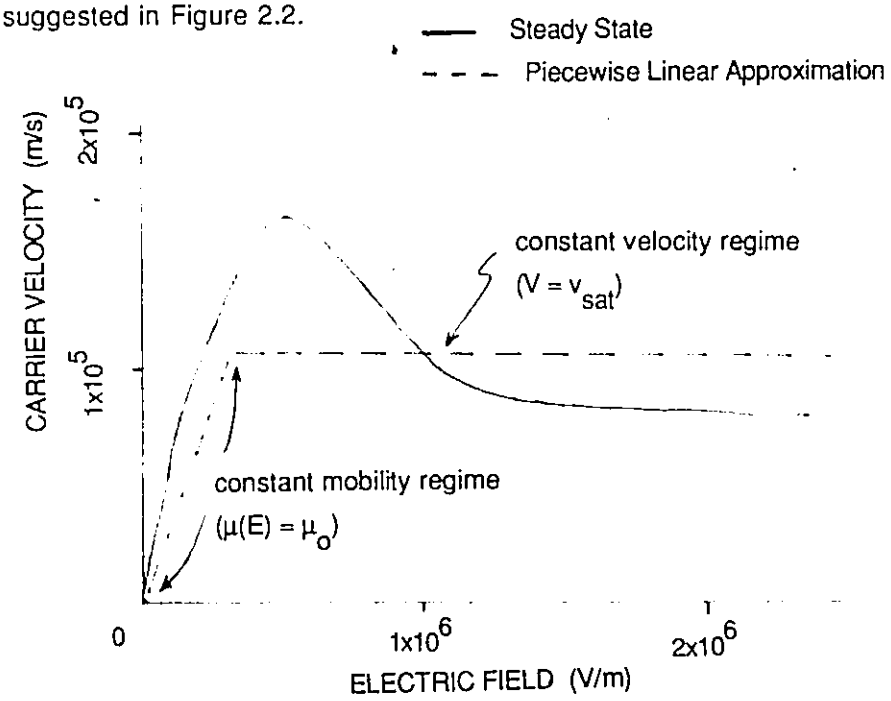


Figure 2.2 Velocity-Field Characteristic for Gallium Arsenide

If $V_{gs}=0$ and V_{ds} is some small value, a voltage drop is induced in the channel by the current flowing from drain to source, causing the gate to become increasingly negatively-biased with respect to the channel towards the drain end. Thus the carrier-depleted region is wider towards the drain end of the channel (i.e. d(x) decreases). Under these low-field conditions, electron mobility is assumed constant ($\mu(E) = \mu_o$), and the channel current may be described as for any standard FET (Shockley,[2.4]):

$$I_{DS} = q \cdot N_d \cdot Z \cdot d(x) \cdot \mu_o \cdot E(x) \qquad (2.1a)$$

where Z is the gate width. As the negative gate bias is increased and d(x) decreases, the field E(x) in the active channel increases beyond the critical value E_c and the carrier velocity attains its saturated value, V_{SAT}, so that:

$$I_{DS} = q \cdot N_d \cdot Z \cdot d(x) \cdot V_{SAT} \qquad (2.1b)$$

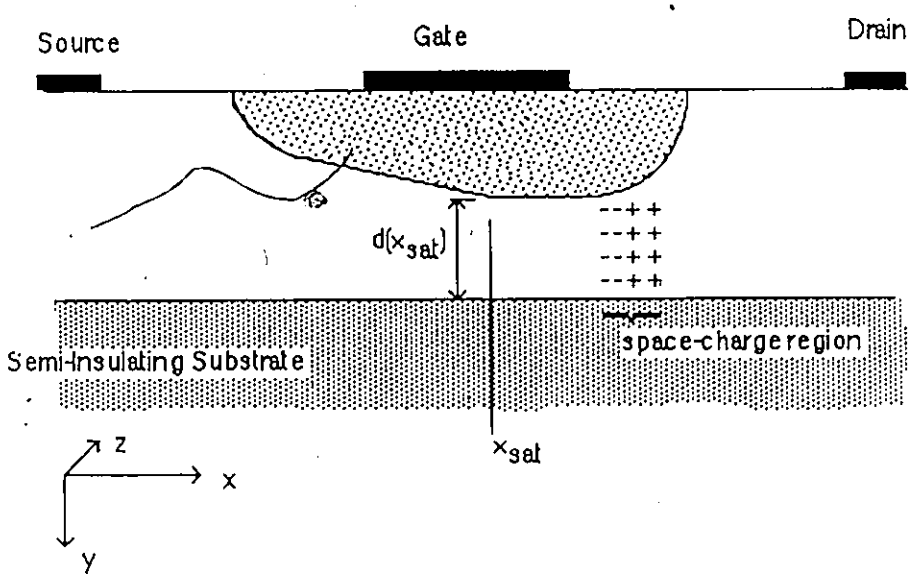


Figure 2.3: Formation of Space-Charge Dipole in GaAsFET

Once V_{SAT} has been reached, (2.1b) indicates that any further decrease in d(x) must be accompanied by an increase in carrier concentration, since I_{DS} is conserved. The abrupt widening of the channel seen at the drain end is then initially compensated for by a decrease in concentration and a space-charge layer is formed (Figure 2.3). Virtually all the voltage drop between drain and source occurs across this dipole layer when the device is in saturation. Thus, once the channel current saturates, it increases only slightly with larger drain voltages.

A typical I_{DS}/V_{DS} characteristic shown in Figure 2.4: a region of linear increase in current with voltage; a 'knee' where channel conductance falls off

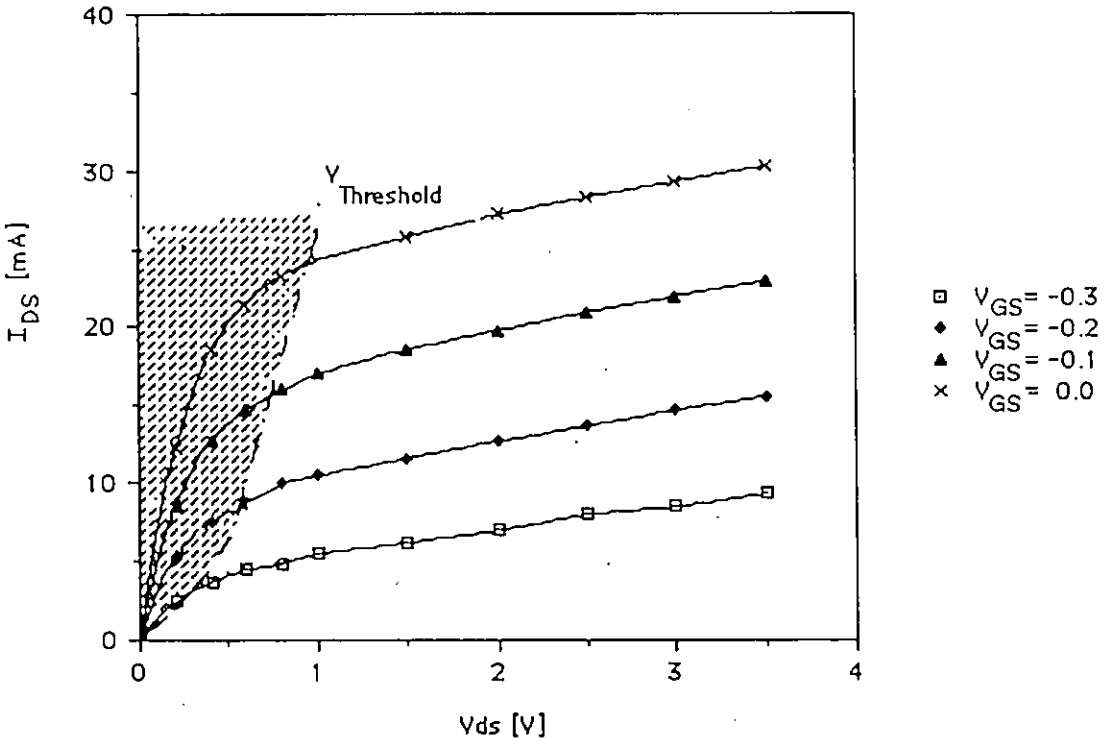


Figure 2.4: IDS-VDS Characteristic for GaAsFET (Measured Data for NEC673)

exponentially; and finally a saturated region where current is almost constant with increasing V_{DS}. The onset of saturation occurs at $V_{DS} = V_{TH}$, the threshold voltage for a given value of V_{GS}. Note that current is not completely constant in the saturated region, the slight increase being caused by a modification of the shape of the depleted channel, as discussed by Lehovec and Miller, [2.5]. As drain voltage is increased for a fixed V_{GS}, the higher electric field shifts the point at which carriers reach v_{sat} (x_{sat}) towards the source end of the active channel, which is less negatively biased with respect to the gate. Hence $d(x_{sat})$ is larger than before and more current is injected into the channel.

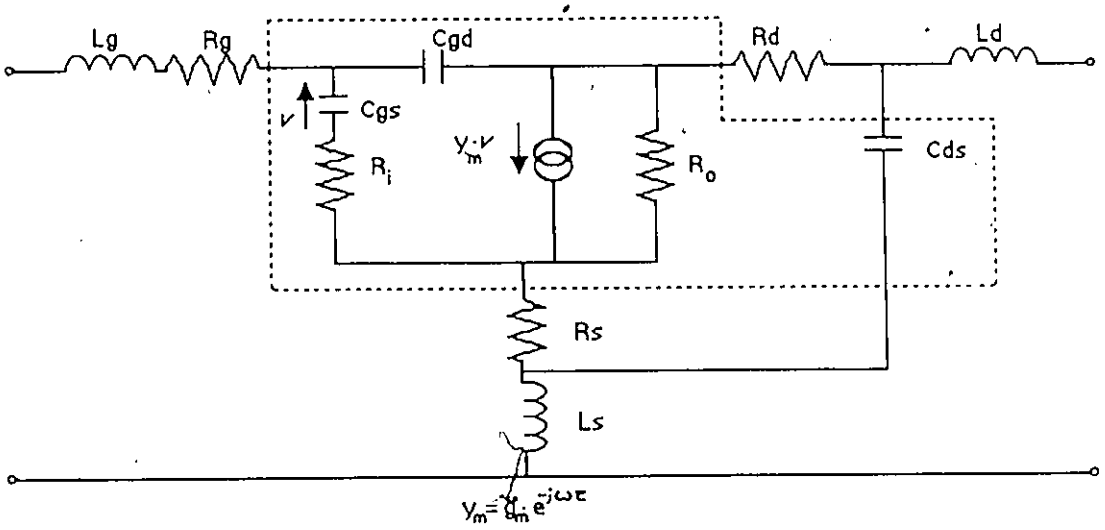


Figure 2.5: Lumped-Element Equivalent Circuit for GaASFET

Figure 2.5 shows a typical lumped-element equivalent circuit representation of a GaAsFET. It contains two different groups of elements: the extrinsic parameters R_S, R_d, R_g, L_s, L_d, and L_g, which are dependent on the device technological parameters only; and the intrinsic parameters (enclosed in the dashed box in the

figure), which are manifestations of the undepleted channel geometry and thus highly bias-dependent. The element values for this equivalent circuit are frequently obtained by a least-squares fit to measured S-parameters [2.6-2.8]. By calculating a series of such models over a range of operating points, one can observe that the principal nonlinear elements, i.e. those most sensitive to variations in operating point, are the gate-source capacitance C_{gs}, the gate-drain feedback capacitance C_{gd}, and particularly the transconductance g_m:

$$g_m \Delta \left| \frac{\partial I_{ds}}{\partial V_{gs}} \right|_{V_{ds} = \text{constant}} \tag{2.2}$$

and the output conductance g_o:

$$g_o \Delta \left| \frac{\partial I_{ds}}{\partial V_{ds}} \right|_{V_{gs} = \text{constant}} \tag{2.3}$$

2.2 FET Model Parameter Extraction

There have been a number of methods proposed for extracting the FET model parameters, including the least-squares fit optimization mentioned above and others [2.9-2.12]. These are satisfactory providing that reasonable starting values are known and only one or two bias points are required. However, for large-signal simulations, which require modeling over a wide range of operating conditions, such an approach is cumbersome. Considerable savings can be realized by selecting a FET model that can be directly related to the measured S-parameters, eliminating the time-consuming optimization step. In general, such models, [2.13-2.14], are again derived from measurements made at many bias points, but over a limited frequency range (usually low enough that the parasistics can be ignored). This approach only

analyzes the intrinsic FET and the access elements must be determined separately.

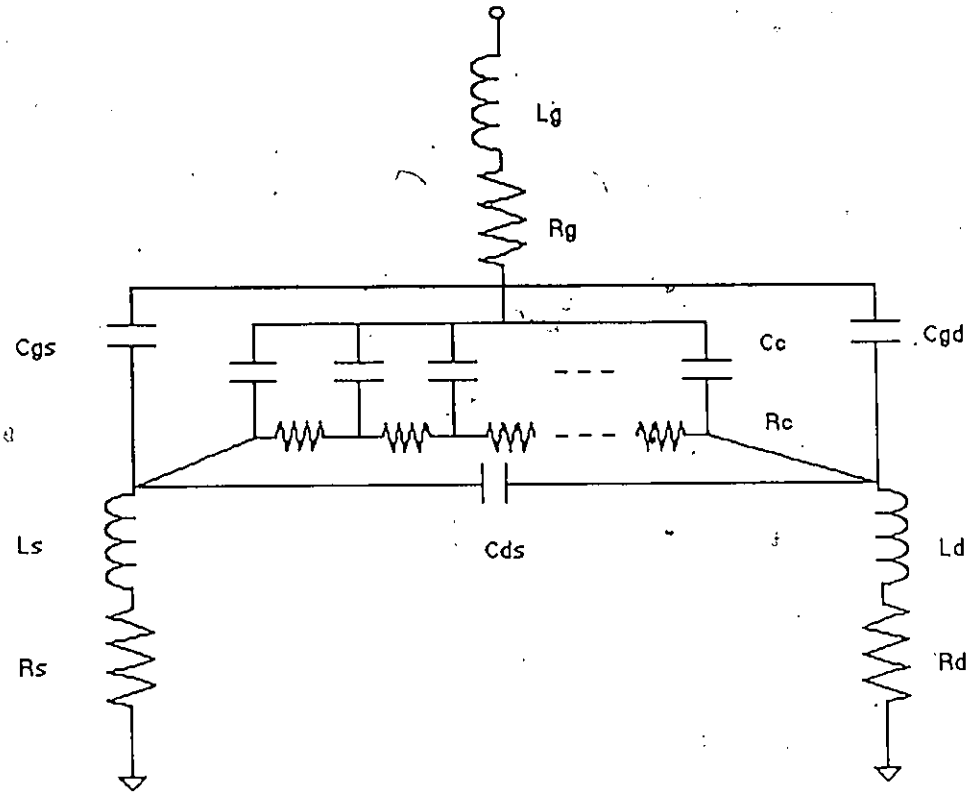


Figure 2.6: FET Model for Zero Drain Current Case

2.2.1 Extrinsic Parameters - Rs, Rd, Rg, Ls, Ld, Lg

One of the most widely-used techniques for calculating the resistive parasitics is the Fukui method, [2.15], however it requires an accurately calibrated DC test measurement system and does not determine the inductances. During the course of work carried out at C.N.E.T., (Centre National des Etudes des Télécommunications, LAB/MER/MLS, Lannion, France), the author had already implemented a program for the HP310 desktop computer which permits the automated measurement of S-parameters at up to 40 bias points, using the HP8510

network analyzer and either programmable or manual power supplies. Since this system was already available, a modeling approach was sought which would exploit its ease of use and accuracy.

The method used for determining the parasitics from RF measurements was first described by Diamand and Laviron, [2.16]. When drain current is zero, a FET may be represented by the simple model shown in Figure 2.6, with the channel region under the gate being described as a uniform transmission line of total resistance R_c and total capacitance C_c. A simplifying condition is imposed that this transmission line be electrically very short, i.e. $|\gamma l|^2 = \omega R_c C_c << 1$, where γ is the propagation constant in the channel. The channel resistance and capacitance are both sensitive functions of the active channel depth:

$$R_c = \frac{L}{q N_D Z \mu_o a} \tag{2.4}$$

$$C_c = \frac{L Z \varepsilon}{(a-d)}, \text{ (where } \varepsilon = \text{dielectric constant in the channel)} \tag{2.5}$$

The frequency of measurements must be kept low enough that the channel is short compared to one wavelength.. A channel opening greater than 20-25% is recommended, to ensure that C_c remains reasonably small, i.e:

$$\frac{d}{a} = 1 - \sqrt{\frac{-V_{GS} + \phi}{V_{GS} + V_p}} > 0.20 \tag{2.6}$$

where ϕ is the built-in gate-channel potential barrier and V_p is the pinch-off voltage, or value of V_{GS} for which d=0.

Under these conditions, the impedance matrix [Z] = [R] + j[X] for the equivalent circuit in Figure 2.6 in common-source configuration may be written:

$$[R] = \begin{bmatrix} Rs+Rg+\dfrac{Cc+3Cds+3Cdg}{3(Cc+Cgs+Cgd)}Rc & Rs+\dfrac{Cc+2Cgd}{3(Cc+Cgs+Cgd)}Rc \\ Rs+\dfrac{Cc+2Cgd}{3(Cc+Cgs+Cgd)}.Rc & Rs+Rd+Rc \end{bmatrix} \qquad (2.7a)$$

$$[X] = \begin{bmatrix} (Ls+Lg)\cdot\omega-\dfrac{1}{\omega\cdot(Cc+Cgs+Cgd)} & Ls\cdot\omega \\ Ls\cdot\omega & (Ls+Ld)\cdot\omega \end{bmatrix} \qquad (2.7b)$$

The inductances can be obtained directly from the reactance matrix [X] and should be independent of V_{GS}, although there is a sharp variation as pinch-off is approached. The resistances are calculated from [R] by noting that, for the flat doping profile assumed, (2.4) may be re-written [17]:

$$R_c = \frac{R_{co}}{1-\sqrt{\dfrac{-V_{GS}+\phi}{V_{GS}+V_p}}} \qquad (2.8)$$

Thus the intercepts of the straight-line graphs of $(R_{21}+R_{12})/2$, R_{22} and R_{11} vs. $\left[1-\sqrt{(-V_{GS}+\phi)/(V_{GS}+V_p)}\right]^{-1}$ may be calculated to give R_s, R_d and R_g, respectively, where:

$$[R] = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}$$

The program 'EXTPAR' listed in Appendix I was developed by the author for the IBM-PC. It extracts the extrinsic parameters from a series of S-parameters

measured at several frequencies over a range of gate biases, (V_{DS} fixed at zero), based on the above analysis. The only DC measurements required are for V_p and ϕ. This program was used to analyze several devices, including the NEC673 and FETs fabricated in-house.

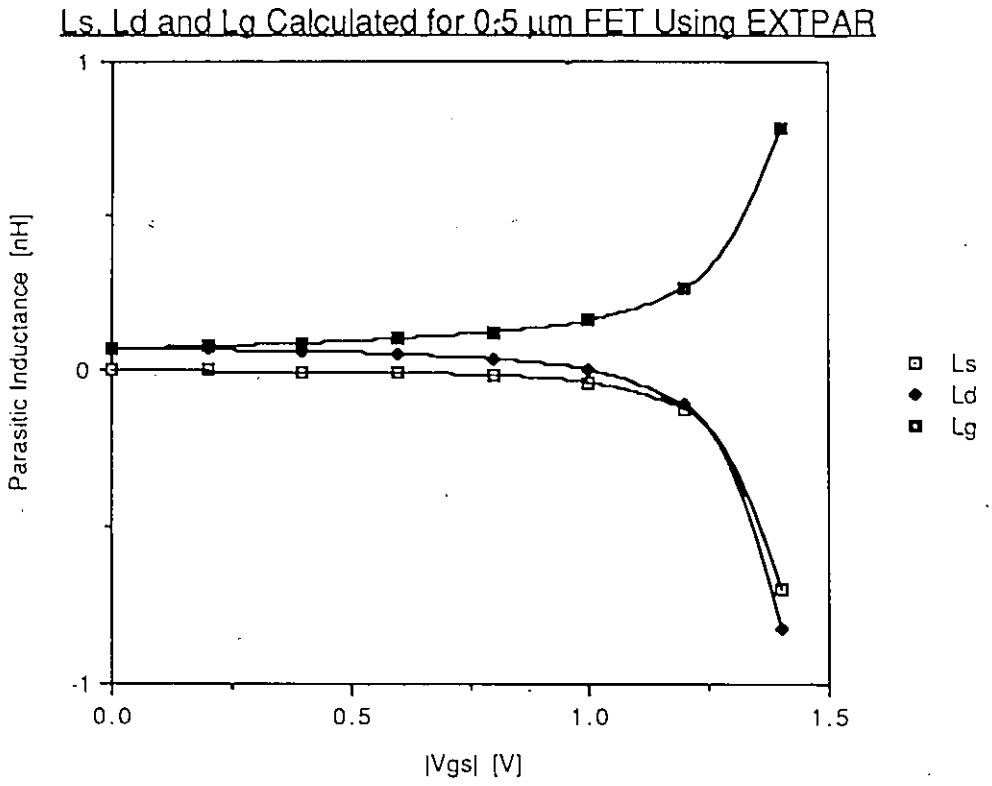


Figure 2.7: Variation of Parasitic Inductance with Gate Bias

Figures 2.7 and 2.8 show typical results obtained for an in-house 0.5 µm GaAsFET having V_p = 1.98 V and ϕ = 0.45 V. As expected, the inductances are seen to be independent of V_{GS} up to V_{GS} = -1.0, corresponding to a channel opening of 23%. (The inductances are extremely low in this case because the FET was probed on-

wafer using the Cascade RF-probing station, [2.18]). The plot of R_{11} vs. $\left[1 - \sqrt{(-V_{GS} + \phi)/(V_{GS}+V_P)}\right]^{-1}$ shows a similar bias limit: frequency dependence is small and the graphs are linear only for channel openings up to about 25% . The deviation from linearity as pinch-off is approached is caused partly by the additional bias-dependence of C_c, which appears in the expression for R_{11}, and partly by the increasing asymmetry of the channel.

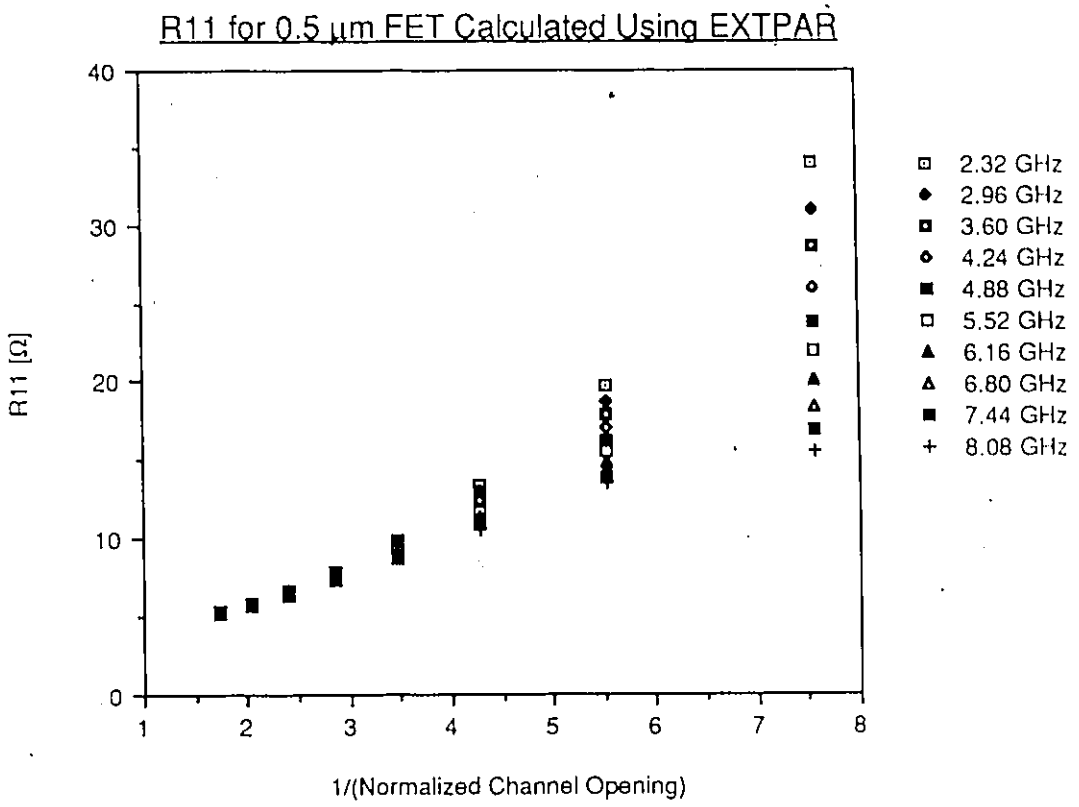


Figure 2.8: Variation of R11 with Gate Bias and Frequency

2.2.2 Intrinsic Parameter Extraction - g_m, C_{gd}, C_{gs}, C_{ds}, g_o, R_i

The characterization of the intrinsic FET parameters for large-signal

operation is done in two steps. First, small-signal equivalent circuits are derived from S-parameters. Once these discrete or 'incremental' parameters are known, global or 'instantaneous' expressions can be developed using the method described in Section 2.3. A relatively large number of incremental equivalent circuits are required for this synthesis, so it is especially desirable to have an efficient algorithm for extracting them. This is why the method proposed by Minasian, [2.19], was selected, since it does not use optimization and requires little CPU time.

By considering only the intrinsic parameters of the circuit shown in Figure 2.5, Minasian showed that it is possible to relate each element directly to the measured Y-parameters. Individual elements are calculated using:

$$g_m = \frac{Re|Y_{21}|}{1 - Re|Y_{21}| \cdot R_g} \tag{2.9}$$

$$g_o = \frac{1}{(1 + g_m \cdot R_s) \cdot Re|Y_{22}|} \tag{2.10}$$

$$C_{ds} = \frac{Im|Y_{12}| + Im|Y_{22}|}{\omega} \tag{2.11}$$

$$C_{gd} = \frac{-Im|Y_{12}|}{\omega} - \frac{R_s \cdot g_d \cdot C_{gs}}{(1 + g_m \cdot R_s)} \tag{2.12}$$

$$C_{gs} = (1 + g_m \cdot R_s) \frac{Im|Y_{12}| + Im|Y_{11}|}{\omega} \tag{2.13}$$

$$R_i = \left[\frac{Re|Y_{11}|}{(Im|Y_{12}| + Im|Y_{11}|)^2}\right] - R_g - R_s \left[1 - \frac{g_m \cdot \tau}{C_{gs}}\right] \tag{2.14}$$

$$\tau = \left[\frac{Im|Y_{12}| - Im|Y_{21}|}{\omega \cdot Re|Y_{21}|}\right] - \left[\frac{Re|Y_{11}|}{(Im|Y_{12}| + Im|Y_{11}|)^2}\right] \tag{2.15}$$

S-parameters must be measured at low enough frequencies that the inductances can be ignored. The author has found empirically that 4 GHz is a good upper limit.

The program 'INTPAR' listed in Appendix I uses data from the HP8510 automated measurement software mentioned earlier to calculated the intrinsic parameters for a given device at up to 40 bias points. Results of this analysis as applied to the NEC673 are shown in the next few pages, and they illustrate a number of important features of GaAsFET nonlinear behaviour.

The computed values of transconductance are shown in Figure 2.9. The slight decrease in g_m for small values of $|V_{GS}|$ has been reported by others [2.20,2.21] and is thought to be due to a reduction of charge-accumulation effects.

The output conductance g_o shown in Figure 2.10 displays a nearly exponential drop in the threshold region, after which point it remains virtually constant over V_{DS}. The drop is due to the accumulation of the space-charge dipole, which begins at the onset of saturation. Since any further increase in V_{DS} is compensated for by this dipole, the voltage drop along the rest of the channel remains constant. Hence V_{DS} has very little further effect on I_{DS} and g_o stabilizes at some small value.

The drain-source capacitance is frequently treated as one of the extrinsic parameters and considered to be constant. As can be seen from the results in Figure 2.11, this assumption is reasonable, as C_{ds} shows much less variation with V_{GS} than does C_{gs}, (Figure 2.12), provided the device is operating well into saturation. There is, however, a marked nonlinearity in C_{ds} as threshold is passed.

C_{gd}, shown in Figure 2.14, is much smaller than the other two capacitances, reflecting the greater distance between the gate and drain contacts. In addition, the extension of the depletion region towards the drain edge of the channel with increasing V_{DS} causes C_{gd} to drop sharply.

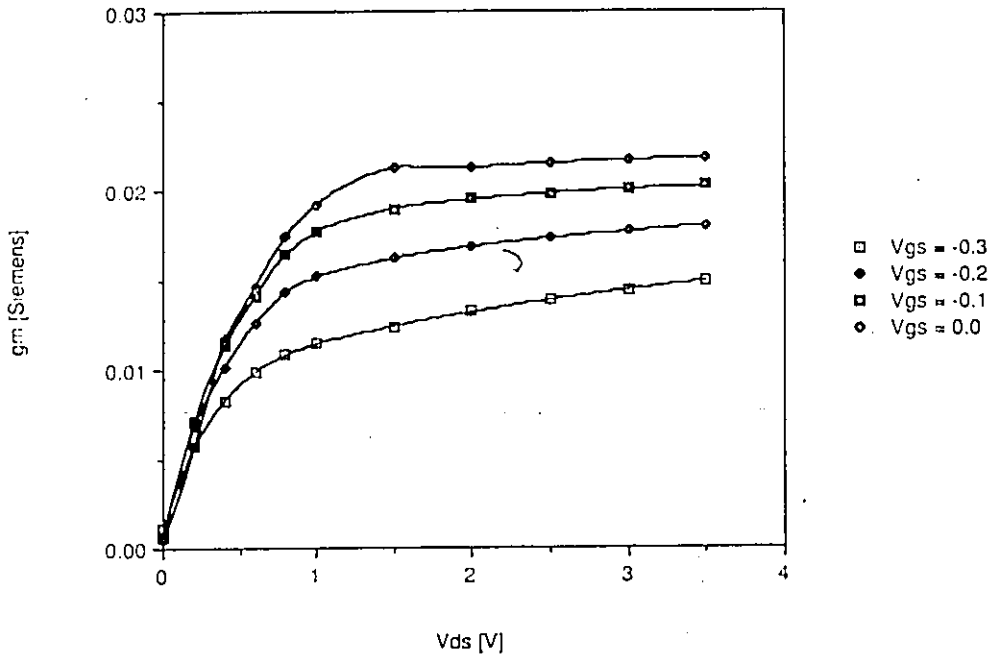


Figure 2.9: Computed Transconductance for NEC673

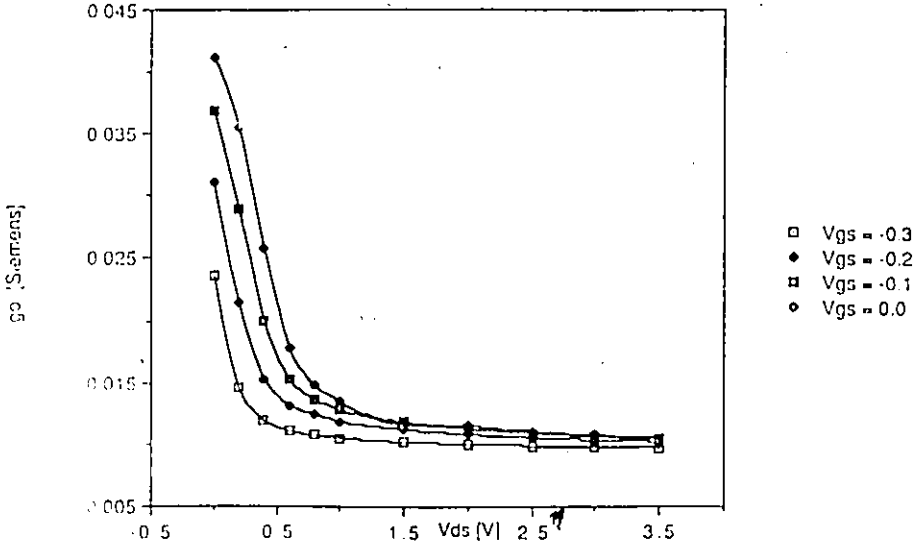


Figure 2.10: Computed Output Conductance for NEC673

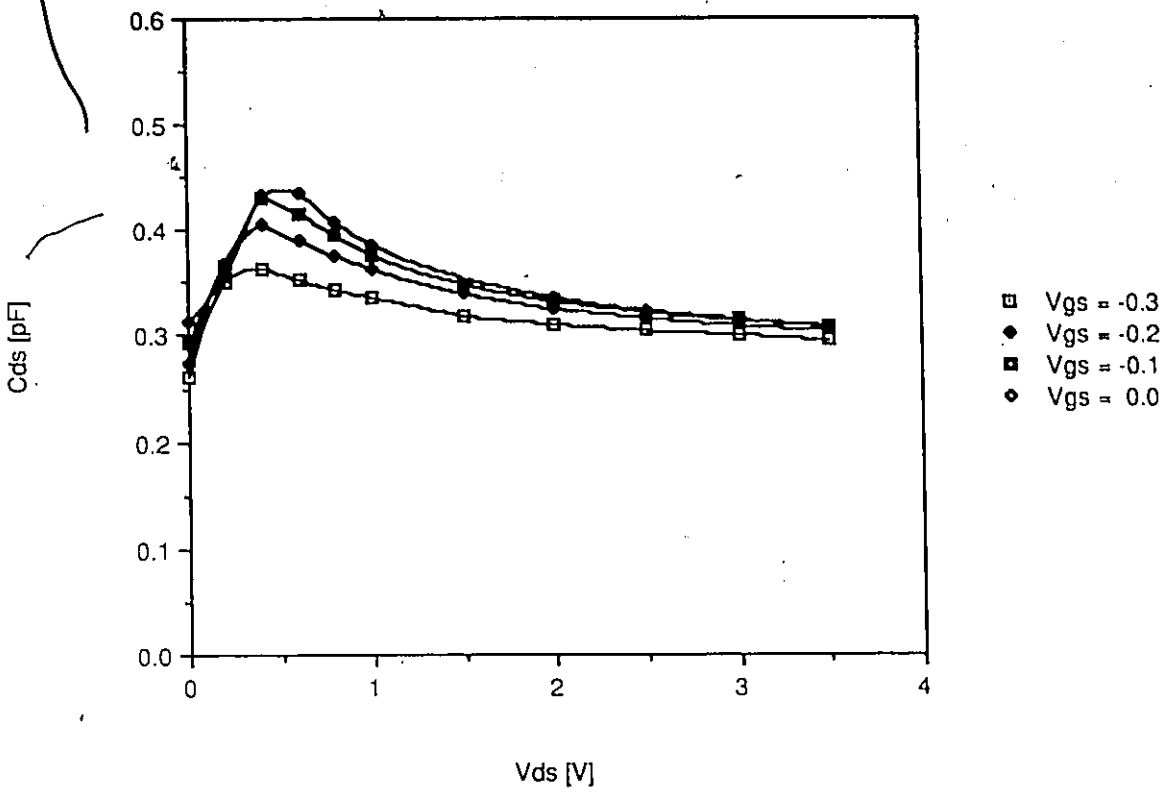


Figure 2.12: Drain-to-Source Capacitance for NEC673

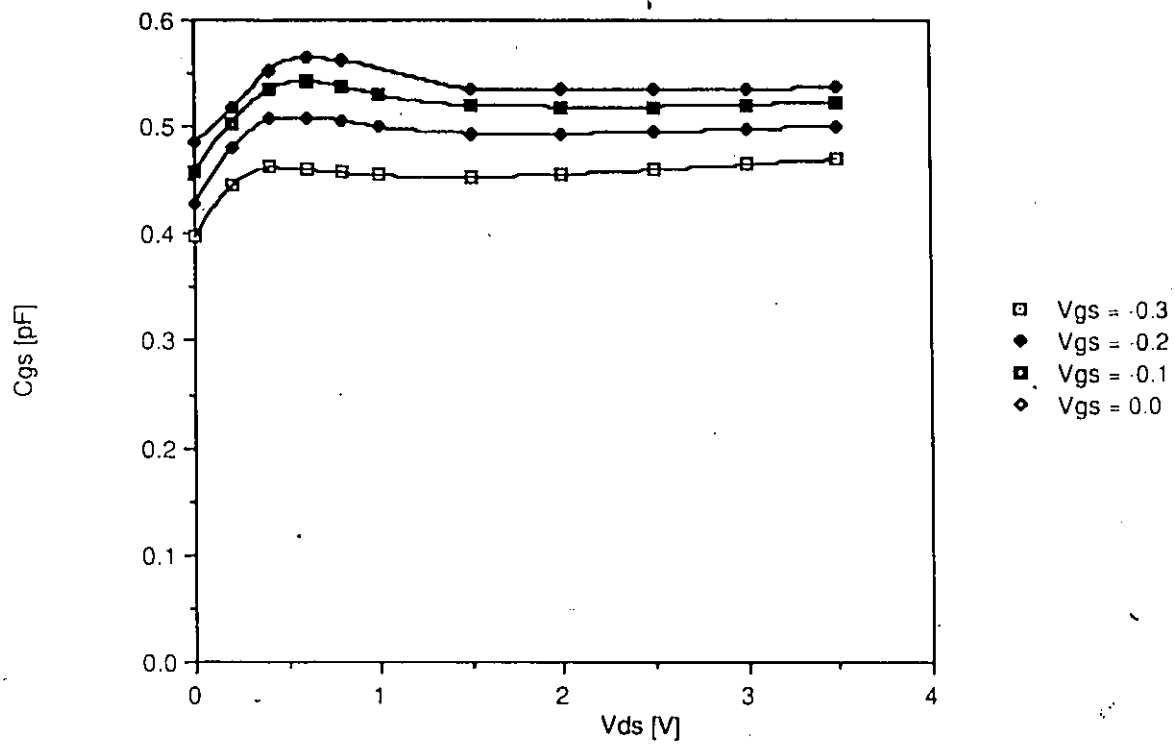


Figure 2.13: Gate-to-Source Capacitance for NEC673

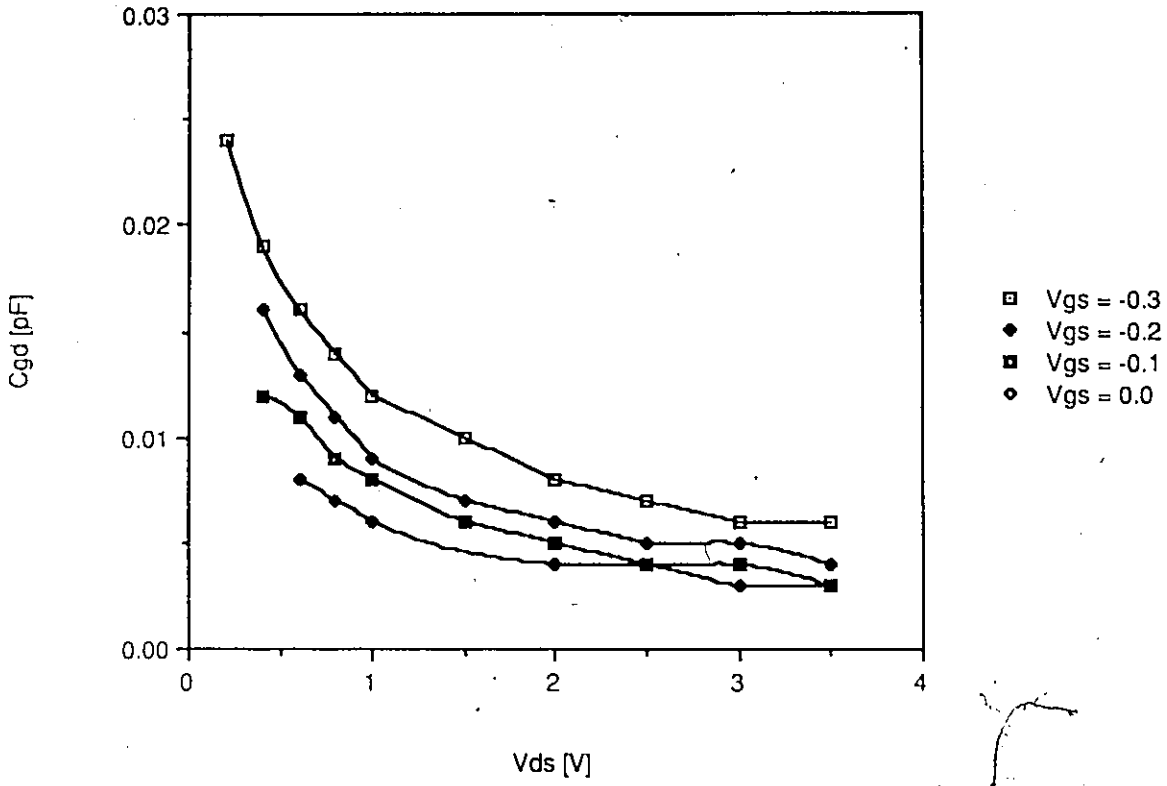


Figure 2.14: Gate-to-Drain Capacitance for NEC673

This model is derived at low frequencies (< 10 GHz), so the validity of extrapolating it was investigated, with the results shown in Figures 2.15-2.17. S-parameters calculated to 18.0 GHz using SUPERCOMPACT™ with the in-house device model for V_{DS} = 2.5 V, V_{GS} = 0 V are compared to actual measurements. The agreement is seen to be good enough to justify using this model for design purposes; at least up to 18 GHz.

In this technique, it is critical that the S-parameters used to characterize the device be properly de-embedded. Device bondwires are accounted for by the parasitic inductances, but connectors and any excess transmission line must be calibrated out using, for example, the Through-Reflect-Line calibration technique, [2.22].

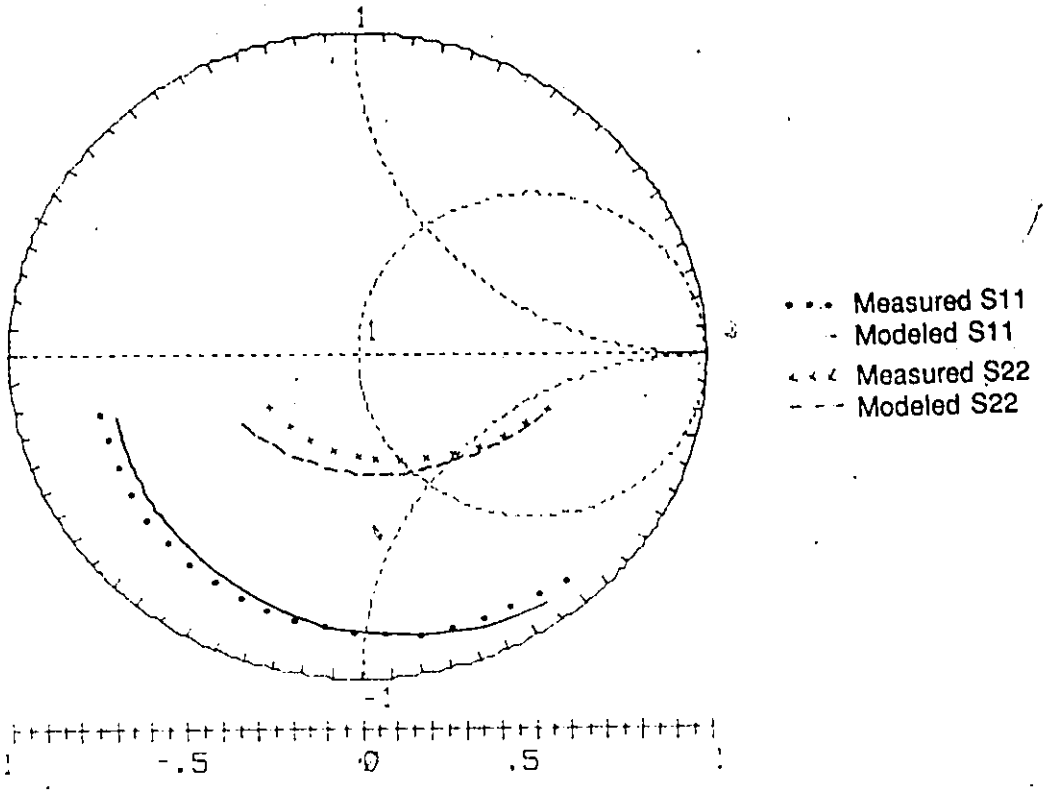


Figure 2.15: Measured vs. Modeled S11and S22 from 2 to18 GHz

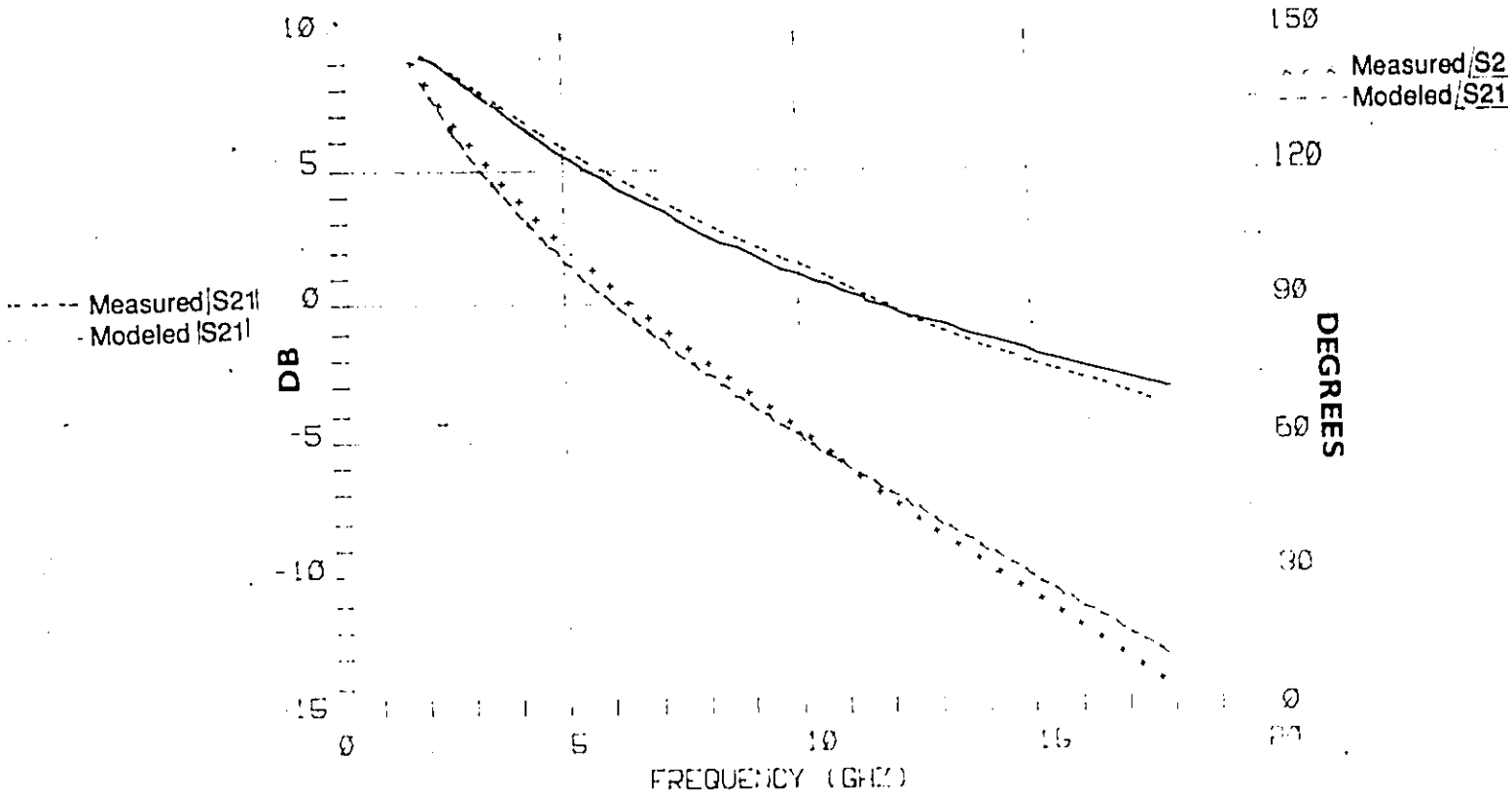


Figure 2.16: Measured vs. Modeled S21 From 2 - 18 GHz

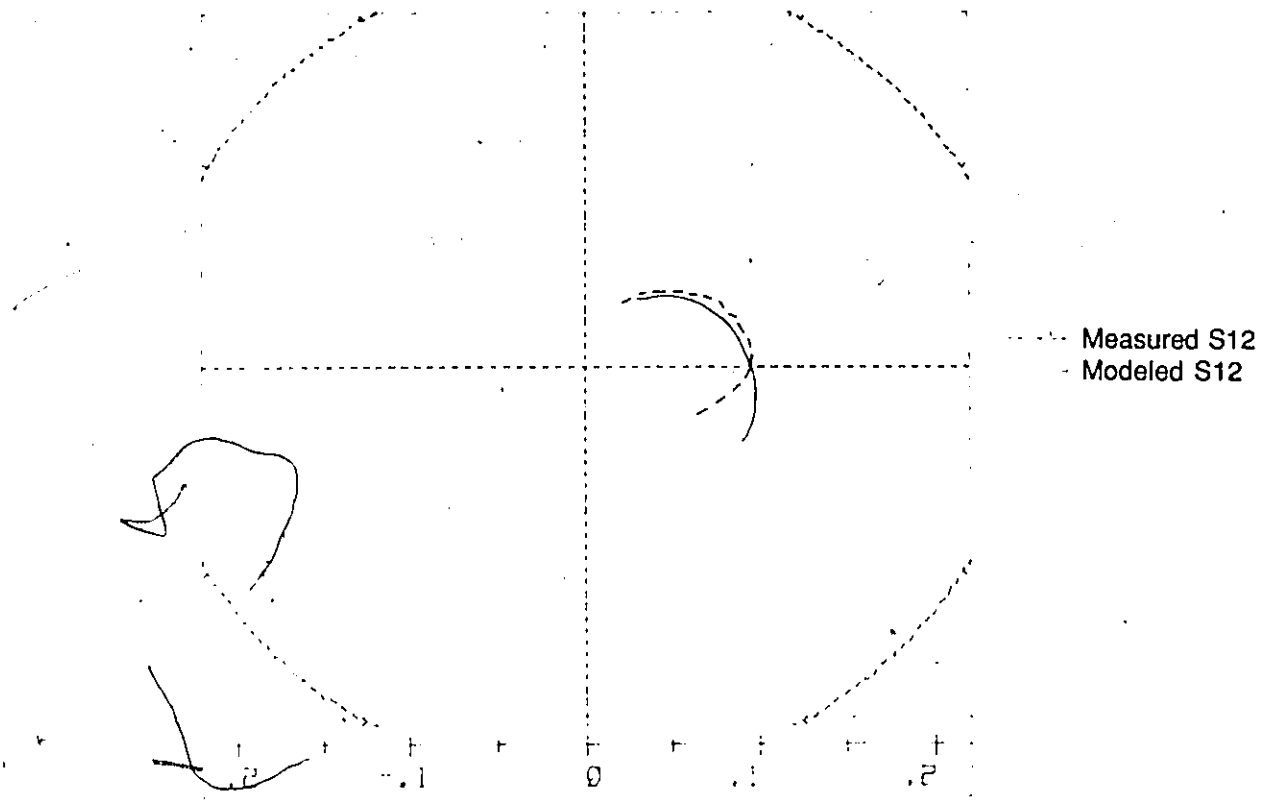


Figure 2.17: Measured vs. Modeled S12 from 2 to 18 GHz

2.3 Global Parameters - Instantaneous Expressions

A systematic procedure has been developed by Willing et al., [2.23], for deriving the global parameter expressions for a large-signal model from the experimentally determined bias-dependence of the various model elements. Although this work was originally based on an equivalent circuit derived using optimization techniques, it is generally applicable, and was used in by the author in this thesis for the model described in the previous sections. The instantaneous current through each nonlinear model element is described as the product of the instantaneous value of that element and the voltage (or time-derivative of the voltage) across it.

Referring to Figure 2.18, the following expressions can be written:

$$i_D(t) = g_M(v_G(t), v_D(t)) \cdot v_G(t) \tag{2.10}$$

$$i_O(t) = g_O(v_G(t), v_D(t)) \cdot v_D(t) \tag{2.11}$$

$$i_G(t) = c_{GS}(v_{GS}(t), v_D(t)) \cdot \frac{\partial v_G(t)}{\partial t} \tag{2.12}$$

where the use of small letters with capital subscripts indicates the instantaneous quantities.

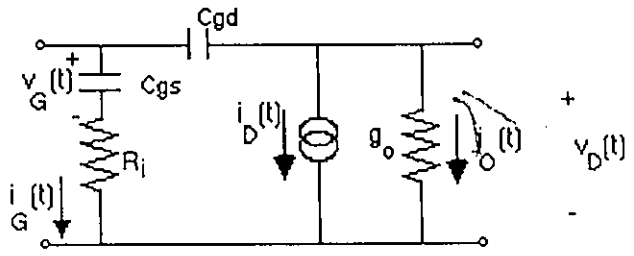


Figure 2.18: Instantaneous Currents and Voltages in Nonlinear FET

Note that the above equations are functions of the internal gate and drain voltages, $v_G(t)$ and $v_D(t)$ respectively, whereas the curves calculated in the previous section were functions of the external terminal voltages V_{GS} and V_{DS}. These static results are referred to the internal voltages V_G and V_D. It is assumed that quasi-static conditions are valid, i.e. that the static expressions may be extended to the time-varying case. The instantaneous voltages and currents may be written as the sum of their static and dynamic components:

$$i_D(t) = I_D + \tilde{i}_d(t) \tag{2.13}$$

$$i_G(t) = I_G + \tilde{i}_g(t) \tag{2.14}$$

$$v_G(t) = V_G + \tilde{v}_g(t) \tag{2.15}$$

$$v_D(t) = V_D + \tilde{v}_d(t) \tag{2.16}$$

where capital letters with capital subscripts denote static quantities and the tilde denotes the superimposed small-signal voltages and currents.

Under quasi-static conditions the expressions for the currents through the nonlinear elements are approximated as:

$$\tilde{i}_d(t) = G_M(v_G,v_D)\cdot\tilde{v}_g(t) \tag{2.17}$$

$$\tilde{i}_o(t) = G_O(v_G,v_D)\cdot(\tilde{v}_d(t)) \tag{2.18}$$

etc.

where $G_M(v_G,v_D)$, $G_D(v_G,v_D)$ are the incremental nonlinear parameters, and the time-dependence has been omitted for simplicity.

The transconductance will be used as an example to show how an instantaneous nonlinearity (g_M) is derived from its experimentally-determined incremental counterpart (G_M) for the resistive elements. Substituting the expression for instantaneous current and voltage into equation (2.10), one obtains:

$$I_D+\tilde{i}_d(t) = g_M(V_G+\tilde{v}_g(t),V_D+\tilde{v}_d(t))\cdot(V_G+\tilde{v}_g(t)) \tag{2.19}$$

Applying the Taylor Series expansion for functions of two variables in terms of $V_G+\tilde{v}_g(t)$ and $V_D+\tilde{v}_d(t)$, (2.19) can be rewritten:

$$I_D+\tilde{i}_d(t) = \left[g_M(V_G,V_D) + \left[\tilde{v}_g(t) \frac{\partial g_M(v_G,v_D)}{\partial v_G} + \tilde{v}_d(t) \frac{\partial g_M(v_G,v_D)}{\partial v_D} \right]_{\substack{v_G=V_G \\ v_D=V_D}} \right] \cdot (V_G+\tilde{v}_g(t)) \tag{2.20}$$

Once the terms in (2.20) have been multiplied out, retaining first order terms and equating the dynamic components, a substitution of (2.19) for the instantaneous parameter results in :

$$G_M(V_G,V_D)\,\tilde{v}_g(t) = \tilde{v}_g(t)\, g_M(V_G,V_D) + V_G \left[\tilde{v}_g(t) \frac{\partial g_M(v_G,v_D)}{\partial v_G} + \tilde{v}_d(t) \frac{\partial g_M(v_G,v_D)}{\partial v_D} \right]_{v_G=V_G} \tag{2.21}$$

or

$$G_M(V_G,V_D) = g_M(V_G,V_D) + \frac{V_G}{\tilde{v}_g(t)} \left[\tilde{v}_g(t) \frac{\partial g_M(v_G,v_D)}{\partial v_G} + \tilde{v}_d(t) \frac{\partial g_M(v_G,v_D)}{\partial v_D} \right]_{\substack{v_G=V_G \\ v_D=V_D}} \tag{2.22}$$

A similar procedure is used to find:

$$G_O(V_G,V_D) = g_O(V_G,V_D) + \frac{V_D}{\tilde{v}_d(t)} \left[\tilde{v}_g(t) \frac{\partial g_O(v_G,v_D)}{\partial v_G} + \tilde{v}_d(t) \frac{\partial g_O(v_G,v_D)}{\partial v_D} \right]_{\substack{v_G=V_G \\ v_D=V_D}} \tag{2.23}$$

For the capacitive elements:

$$I_G+\tilde{i}_g(t) = c_{GS}(v_G,v_D)\cdot\frac{d}{dt}(V_G+\tilde{v}_g(t))$$

$$\tilde{i}_g(t) = c_{GS}(v_G,v_D)\cdot\frac{d}{dt}(\tilde{v}_g(t))$$

$$= C_{gs}(v_G,v_D)\cdot\frac{d}{dt}(\tilde{v}_g(t))$$

or

$$C_{gs}(V_G,V_D) = \left| c_{GS}(v_G,v_D)\right|_{\substack{v_D=V_D\\ v_G=V_G}} \tag{2.24}$$

Similarly:

$$C_{gs}(V_G,V_D) = \left| c_{GS}(v_G,v_D)\right|_{\substack{v_D=V_D\\ v_G=V_G}} \tag{2.25}$$

For the charging resistance R_i:

$$R_i(V_G,V_D) = \left| r_I(v_G,v_D)\right|_{\substack{v_D=V_D\\ v_G=V_G}} \tag{2.26}$$

Closed-form expressions for the instantaneous capacitances and charging resistance can thus be obtained by fitting two-dimensional polynomials in $v_G(t)$ and $v_D(t)$ directly to the incremental parameters. The transconductance is derived by noting that, by definition (2.2) $v_D(t)$ must be constant, i.e. $\tilde{v}_d(t) = 0$. Imposing this condition on (2.22) and substituting the fitted polynomial for $G_M(v_G,v_D)$ results in a relatively simple expression for $g_M(v_G,v_D)$. If

$$G_M(v_G,v_D) = \sum_{m=0}^{n} f_n\,(v_D)\cdot v_G^m \tag{2.27}$$

then, based on (2.22) with $\tilde{v}_d(t)=0$:

$$g_M(v_G,v_D) = \sum_{m=0}^{n} \frac{f_n(v_D)}{m+1} \cdot v_G^m \qquad (2.28)$$

and, similarly:

$$g_O(v_G,v_D) = \sum_{m=0}^{n} \frac{h_n(v_G)}{m+1} \cdot v_D^m \qquad (2.29)$$

where:

$$G_O(v_G,v_D) = \sum_{m=0}^{n} h_n(v_G) \cdot v_D^m \qquad (2.30)$$

The resulting formulas for the instantaneous parameters may be easily inserted as subroutines in any nonlinear analysis software, and are applied in the Modified Harmonic Balance Program described in the next chapter.

Chapter 2 References

[2.1] J. Turner, A. Waller, R. Bennett and D. Parker, "An Electron Beam Fabricated GaAs Microwave Field-Effect Transistor," *1970 Symp. GaAs and Related Compounds*, 234-239.

[2.2] C. Liechti, E. Gowen, and J. Cohen, "GaAs Microwave Schottky-Barrier FET," *1972 Int. Solid-State Circuits Conf. Digest*, 158-159.

[2.3] W.W. Hooper et al., "An Epitaxial GaAs Field-Effect Transistor.", *Proc.IEEE*, 55, No. 7, 1237-1238, July 1967.

[2.4] W. Shockley, "A Unipolar 'Field-Effect' Transistor", *Proc.IRE*, **40**, No.11, 1365-1376, November 1952.

[2.5] K. Lehovec and R. Miller, "Field Distribution in Junction Field-Effect Transistors at Large Drain Voltages", *IEEE Trans. Electron Device*, **ED-22**,no. 5, 273-281, May 1875

[2.6] H. Willing et al., "A Technique for Predicting Large-Signal Performance of a GaAs MESFET", *IEEE Trans. on MTT*, **MTT-26**, No. 12, 1017-1023, December 1978.

[2.7] W. Curtice and R. Camisa, "Self-Consistent FET Models for Amplifier Design and Device Diagnostics", *1984 IEEE MTT Symposium*, 427-429.

[2.8] G. Vendelin and M. Omori, "Circuit Model for the GaAs MESFET Valid to 12 GHz", *Electronics Letters*, **11**, 60-61, February 1975.

[2.9] A. Madjar and F. Rosenbaum, "A Large-Signal Model for the GaAs MESFET", *IEEE Trans. on MTT*, **MTT-29**, No. 8, 781-788, August 1981.

[2.10] Y. Tajima et al. "GaAs FET Large-Signal Model and Its Application to Circuit Designs',*IEEE Trans. on Electron Devices*, **ED-28**, No. 2, 171-175, February 1981

[2.11] R. Engelmann. and C. Liechti, "Gunn Domain Formation in the Saturated Current Region of GaAs MESFETs",*1976 IEEE Int. Elect. Devices Conf. Digest*,351-354.

[2.12] V. Hwang. and T. Itoh, "An Efficient Approach for Large-Signal Modeling and Analysis of the GaAs MESFET", *IEEE Trans on MTT*, **MTT-35**, No. 4, 396-402, April 1987.

[2.13] R. Minasian, "Simplified GaAs MESFET Model to 10 GHz", *Electronics Letters*, **13**, 549-551, September 1977.

[2.14] M. Sango et al., "A GaAs MESFET Large-Signal Circuit Model for Nonlinear Analysis", *1988 IEEE MTT Symposium, 1053-1056.*

[2.15] H. Fukui, "Determination of the Basic Parameters of a GaAs MESFET", *Bell Syst. Tech. J.*, **58**, No. 3, March 1979.

[2.16] F. Diamand and M. Laviron, "Measurement of the Extrinsic Series Elements of a Microwave MESFET Under Zero Current Conditions", *12th European Micr. Conference*, September 1982.

[2.17] A. Grebene and Gandhi, "General Theory for Pinched Operation of the Junction-Gate FET", *Solid Stae Electron.*, **21**, 573-589, July 1969.

[2.18] Hewlett-Packard Product Note 8510-6, 'On-Wafer Measurements Using the HP 8510 Network Analyzer and Cascade Microtech Wafer Probes', May 1986.

[2.19] Minasian, *op.cit.*

[2.20] D. Morgan and M. Howes, eds., Gallium Arsenide: Materials, Devices and Circuits.,Chapter 10, John Wiley and Sons, 1985.

[2.21] Engelmann and Liechti, *op.cit.*

[2.22] Hewlett-Packard Product Note 8510-8, 'Applying the HP 8510B TRL Calibration, October 1987.

[2.23] Willing et. al., *op.cit.*

3. NONLINEAR ANALYSIS : THE MODIFIED HARMONIC BALANCE METHOD

3.0 Introduction

Nonlinear analysis has an extremely broad range of applications in circuit design, since all transistors and diodes have inherent nonlinear characteristics which can affect circuit performance to a greater or lesser degree, depending on operating conditions. The resulting circuit response may be undesirable as, for example, in the case of large-signal amplifiers, where it causes intermodulation and distortion products, AM-to-PM conversion and other unwanted effects. The same behaviour is, however, exploited in the operation of such components as mixers, harmonic generators and microwave detectors.

For purposes of circuit analysis, nonlinear systems may be considered in two groups: slightly nonlinear systems, in which response gradually becomes more nonlinear as operating conditions vary; and highly nonlinear systems, which depend upon the nonlinearity for their operation. The first group, (which includes generation of intermodulation and spurious products and microwave detection), exhibits a continuum of responses, so it may be appropriate to analyze such problems using 'piecewise linear' or perturbation techniques. Analysis of the second group, however, must use true nonlinear methods, since attempts to linearize such systems eliminates the behaviour one is trying to characterize. Harmonic generation

and mixing are instances of processes which require full nonlinear treatment and, since closed-form analytical solutions are rare for this class of problems, it is almost always necessary to invoke numerical techniques to obtain solutions.

3.1 Nonlinear Analysis Techniques[3.1]

The term 'nonlinear' in this context actually covers several different families of differential equations (DE's) and, while the Laplace Transform provides a general procedure for the solution of linear homogeneous DE's, there is no analogous approach for handling the nonlinear case. Solutions must be developed according to the specific characteristics of the system under consideration. Typical examples include: parametric systems, which are linear but have time-varying coefficients of the form $\ddot{y} + [a + bf(t)] \cdot y = 0$; 'true' nonlinear differential equations, where coefficients are themselves functions of the dependent variable, (e.g. $\ddot{y} + f(y) \cdot \dot{y} + a \cdot y = 0$); and partial DE's for distributed circuit analysis, i.e. those in which spatial effects are included. The problem of harmonic generation in GaAsFET's falls in the second group.

The steady state solutions for the above systems may, in theory, be obtained simply by numerical integration, but this is not usually practical, due to the amount of computation time required. This is particularly true for microwave systems, where time constants may range from picoseconds for parasitic elements to seconds for the bias circuits. Analysis would require both small step sizes to retain

all information and a long period to arrive at the steady-state solution. In addition, convergence of such mathematically 'stiff' systems cannot be guaranteed.

Prior to the advent of widely-available digital computers, many analytical tools were developed for application to nonlinear problems. Perturbation and averaging techniques and graphical methods such as phase-plane diagrams are all examples. The extra analysis required to implement these techniques results in greatly reduced computation times as compared to 'brute-force' programming, so they are still generally used as the basis of nonlinear analysis software. In particular, the principle of 'harmonic balance' is the foundation of the averaging or 'residual' techniques, when applied to periodic systems. The solution assumed for such systems can always be expressed as a Fourier series of sinusoids, and the principle of harmonic balance is that, by adjusting successive harmonic terms in the assumed solution series to satisfy the original equation, one can reduce the residual to an arbitrarily small quantity.

The efficiency of this approach is improved by minimizing the number of variables to be optimized. The Modified Harmonic Balance (MHB) method proposed by Nakhla and Vlach, [3.2], exploits the fact that the circuits to be analyzed are typically composed of a small nonlinear subcircuit (e.g. a diode junction capacitance), embedded in a much larger linear subcircuit (such as matching and biasing networks). Since the linear portion can be efficiently analyzed in the frequency domain, a great improvement in speed and convergence can be obtained by

analyzing only the nonlinear portion in the time-domain and interfacing between the two domains via the Fourier transform, as suggested in Figure 3.1. The currents and voltages at the ports of each subcircuit must be equal, so in this case the residual function $\varepsilon(t)$ is defined as a measure of the difference between them. The process of harmonic balance provides a means of self-consistent solution between the sections by minimizing the residual at each harmonic. The function $\varepsilon(t)$ can be appropriately chosen so as to optimize the circuit response even as it is being calculated, as illustrated by Guo *et. al.* in [3.3]. This would entail simultaneous modification of linear circuit parameters as well as the circuit response. An additional benefit of carrying out the analysis of the embedding circuitry in the frequency domain is that the effects of dispersion in media such as finline may be readily included.

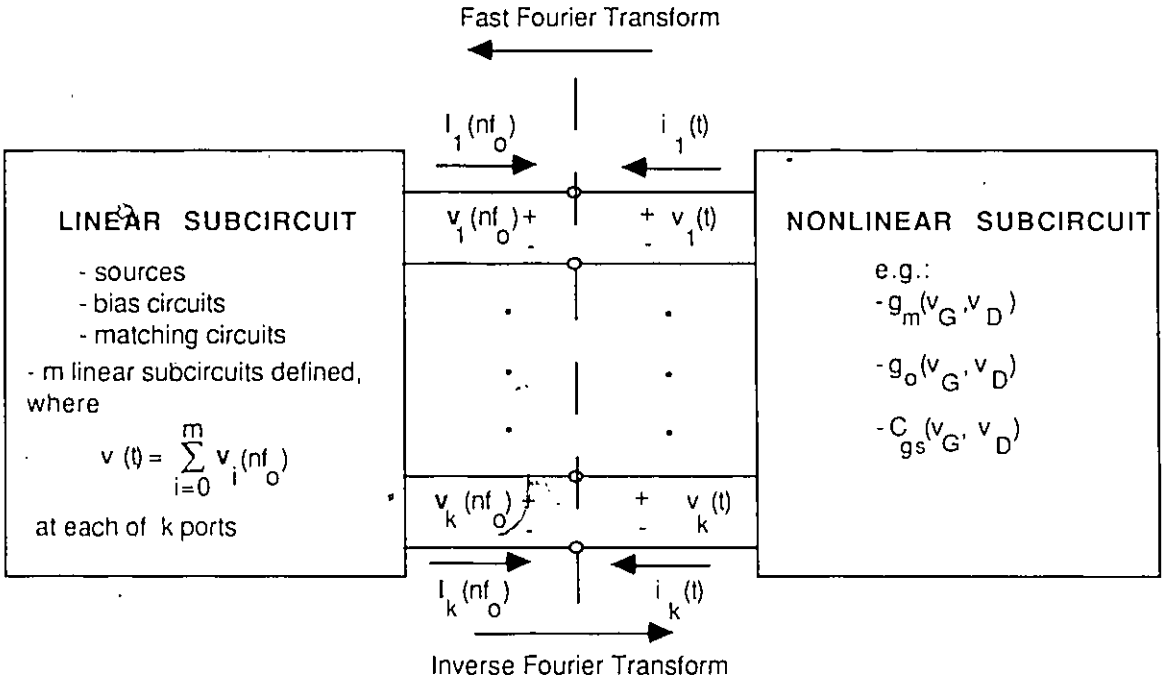


Figure 3.1: Partitioning of Subcircuits for MHB Method

This method has become one of the most popular techniques for analyzing nonlinear circuits, and has been applied successfully to the analysis of many GaAsFET circuits, including mixers with gain (Maas, [3.4]), image-reject mixers (Holmes *et.al.*, [3.5]), amplifiers (Rhyne and Steer, [3.6]), and distributed amplifers (Curtice, [3.7]), as well as multipliers (Gilmore, [3.8]). In this thesis, it is applied to GaAsFET frequency multipliers to provide insight into the effects of various parameters such as bias level and output fundamental termination on operating efficiency.

3.2 The Modified Harmonic Balance Method

The large-signal model used in this work is the one described in Chapter 2, however the actual details of the FET operation are not important in the implementation of the MHB method, provided that the voltage dependencies of the nonlinear device elements are adequately described.

Figure 3.2 shows the equivalent FET multiplier circuit, consisting of the FET and its matching and bias circuitry. Linear FET elements such as the parasitics introduced by bondwires and contact resistances R_d and R_g are absorbed into the embedding circuits, leaving the principal nonlinear effects of $g_M(v_G,v_D)$, $g_o(v_G,v_D)$, $C_{gs}(v_G,v_D)$and $C_{gd}(v_G,v_D)$ to characterize the 'intrinsic' FET.

The goal of the analysis is to determine the output voltage waveforms, once the excitation at the fundamental ($\tilde{v}_g(t)$) and the bias conditions are known. The

nonlinear differential equations describing the intrinsic FET are solved using numerical integration. The linear portions, consisting of input and output embedding impedances seen at DC, the fundamental and its harmonics, are solved in the frequency domain. The application of this algorithm is valid only for stable FET operation. If the device is potentially unstable, its actual operating point will depend on initial conditions and will not be unique. The analysis can be carried out by extension to include any number of harmonics, thus being equally adaptable to xN frequency multiplication.

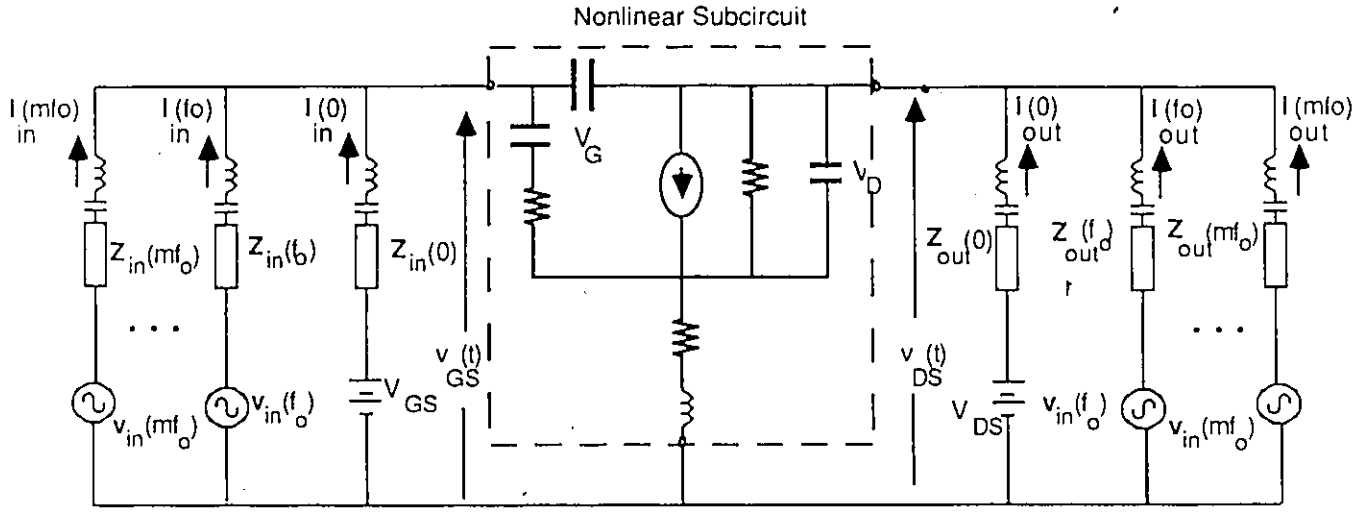


Figure 3.2: Equivalent FET Multiplier Circuit (includes m harmonics)

Note that the linear networks shown in Figure 3.2 include ideal LC filters in series with the embedding impedance for each harmonic being considered. This is equivalent to saying that each spectral component is treated independently during the

frequency-domain calculations. The loop equations for these networks are:

$$\mathbf{v}_{in}(nf_o) = \mathbf{Z}_{in}(nf_o)\cdot\mathbf{I}_{in}(nf_o) + \mathbf{v}_{GS}(nf_o) \tag{3.1}$$

$$\mathbf{v}_{out}(nf_o) = \mathbf{Z}_{out}(nf_o)\cdot\mathbf{I}_{out}(nf_o) + \mathbf{v}_{DS}(nf_o) \tag{3.2}$$

$$n = 0,1,2,3,...$$

$v_{in}(nf_o)$ is the nth harmonic of the input signal, so for the case of pure sinusoidal excitation $v_{in}(nf_o)=0,\ n\neq 0,1$. $v_{GS}(nf_o)$ represents the nth harmonic of the gate-source voltage waveform at the device terminals, and is generally complex. $Z_{in}(nf_o)$ and $i_{in}(nf_o)$ are the impedance and current, respectively, seen in the nf_o branch of the input network. Similar notation applies to the quantities in the output network, where initially $v_{out}(nf_o) = 0$ for $n\neq 0$, since only DC bias is applied at the drain. (Boldface type is used to signify frequency-domain quantities).

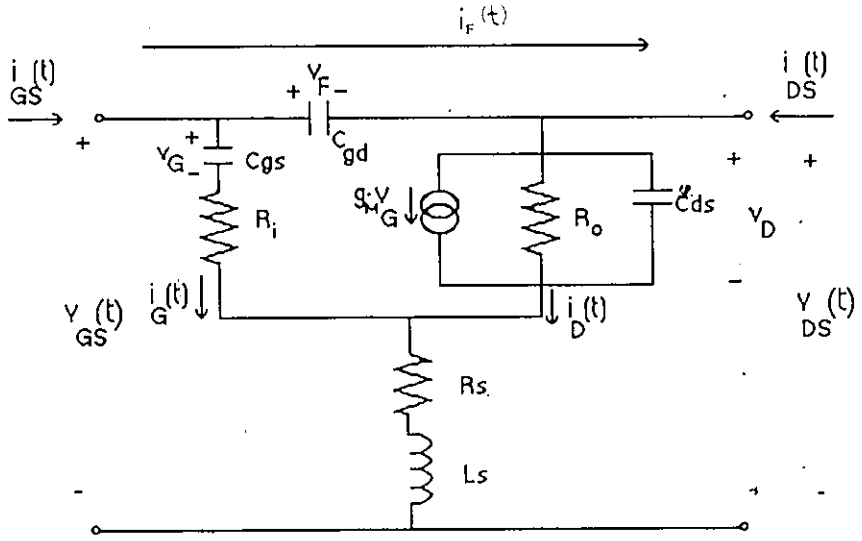


Figure 3.3: Nonlinear FET Loop Equations

Figure 3.3 shows a more detailed schematic of the intrinsic FET, illustrating the loops used to derive the differential equations:

$$-v_{GS}(t)+v_G(t)+\left(i_G(t)\cdot R_i\right)+[i_D(t)+i_G(t)]\cdot R_s+L_s\frac{d}{dt}[i_D(t)+i_G(t)] = 0 \tag{3.3}$$

$$-v_{DS}(t)+v_D(t)+[i_D(t)+i_G(t)]\cdot R_s+L_s\frac{d}{dt}[i_D(t)+i_G(t)] = 0 \tag{3.4}$$

$$-v_{GS}(t) + v_f(t) + v_{DS}(t)= 0 \tag{3.5}$$

Equations 3.1-3.5 form a set of equations in two unknowns, namely the terminal voltages $v_{GS}(t) = V_{GS} + \tilde{v}_{gs}(t)$ and $v_{DS}(t) = V_{DS} + \tilde{v}_{ds}(t)$. Given the initial conditions one can proceed iteratively, alternating between the two sets of equations until the voltage waveforms produced by both are the same. Various optimization techniques may be used to arrive at the correct series of coefficients for the Fourier series representation of these waves.

The reflection algorithm used in this work was developed by Kerr, [3.9], and is one solution which avoids some of the problems inherent in the usual optimization techniques: in particular, it does not require assumption of an initial trial function, which if improperly chosen can drastically affect convergence. Lengths of imaginary ideal lossless transmission line are inserted between the linear and nonlinear portions of the circuit. These lines are an integral number of wavelengths long and are of characteristic impedance Z_{cin} at the input and Z_{cout} at the output. Transient analysis is then used to find the steady-state operating conditions, starting only from the initial excitation.

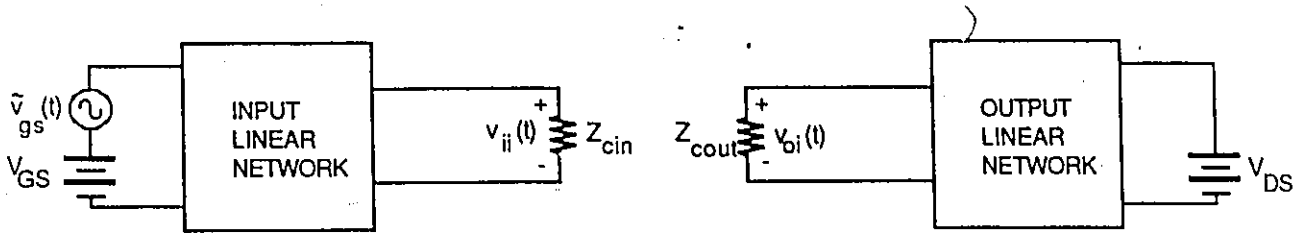


Figure 3.4: Multiplier Equivalent Circuit at t=0

When input signals are applied to the transmission lines from the linear circuits at t=0, the equivalent circuit is as shown in figure 3.4, prior to the signals reaching the device terminals. The initial voltage waves incident on the FET (assuming only $\tilde{v}_{gs}(t)$, V_{GS} at the input and V_{DS} at the output) are:

$$v_{oi}(t) = \frac{V_{DS} \cdot Z_{cout}}{Z_{cout} + Z_{out}(0)} \tag{3.6}$$

$$v_{ii}(t) = \left(\frac{V_{GS} \cdot Z_{cin}}{Z_{cin}+Z_{in}(0)}\right) + \frac{Z_{cin}}{\sqrt{[Z_{cin}+\mathrm{Re}|Z_{in}(nf_o)|]^2+\mathrm{Im}|Z_{in}(nf_o)|^2}} \cdot \cos\left(\omega t - \tan^{-1}\left(\frac{\mathrm{Im}|Z_{in}(f_o)|}{Z_{cin}+\mathrm{Re}|Z_{in}(nf_o)|}\right)\right) \tag{3.7}$$

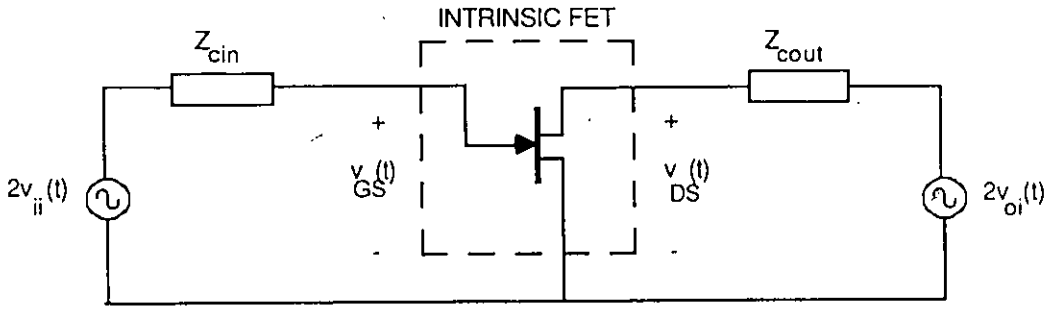


Figure 3.5: Incident Waves Reach FET

On reaching the FET, these incidental waves are used in Equations (3.3)-(3.5) to solve for the terminal voltages and currents, with the transmission line being represented as a source of internal impedance Z_c, (Figure 3.5). Note that the differential equations must be integrated over a sufficient number of cycles for steady state to be reached. These solutions are in turn propagated back along the transmission lines as the reflected waveforms $v_{or}(t)$ and $v_{ir}(t)$ (Figure 3.6). At the linear input and output networks, these are transformed into the frequency domain using the Fast Fourier Transform (FFT). The new incident wave ($v_{ii}(t)$, $v_{oi}(t)$) is found by multiplying each harmonic of $v_{ir}(t)$ and $v_{or}(t)$ by its reflection coefficient, performing an inverse FFT and adding the result to the previous incident wave. The reflection coefficient at the input network is:

$$\rho_{IN}(nf_o) = \frac{Z_{IN}(nf_o) - Z_{cin}}{Z_{IN}(nf_o) + Z_{cin}} \tag{3.8}$$

The output $\rho_{out}(nf_o)$ is calculated the same way.

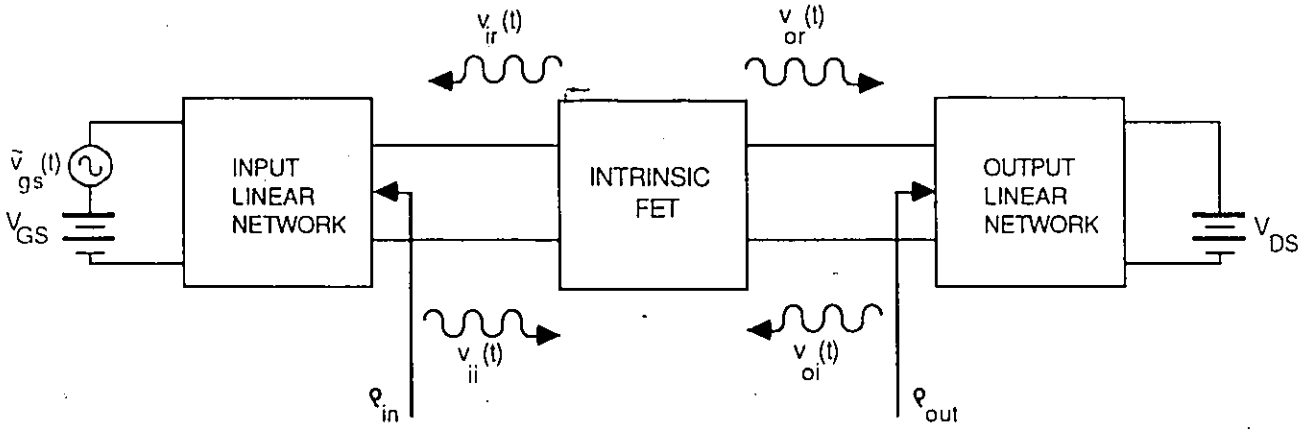


Figure 3.6: Reflected and Incident Waves in Circuit

Convergence may be determined in a number of ways in either the frequency or time-domain. For example, in his analysis of FET mixers, Maas chooses a minimal change in FET terminal voltages and currents as the convergence criterion. A similar test is used in [3.9], only it is applied in the frequency domain:

$$\frac{Z^{i}(nf_o)}{Z^{i+1}(nf_o)} \leq 1+\varepsilon \tag{3.9}$$

where the superscript indicates the iteration number, ε the specified tolerance and

$$Z_x(nf_o) = \frac{v_x(nf_o)}{I_x(nf_o)} \quad , \quad x=ii,oi \tag{3.10}$$

The two ports may converge at the same time, but this is not always the case, so the test must be applied separately. The characteristic impedance of the transmission lines, (which need not be the same), can affect how rapidly the equations converge. Choosing Z_{cin} and Z_{cout} so as to minimize $\rho_{in}(f_o)$ and $\rho_{out}(f_o)$ will improve convergence speed.

A listing of the FET frequency multiplier analysis program written by the author based on the MHB method is provided in Appendix II. Initial conditions for v_G, v_D and i_D and $^{di_G}/_{dt}$ are estimated, based on the expected DC components of the solution. The internal drain current $i_D(0)$ is then calculated using the instantaneous expressions as described in Chapter 2:

$$i_D(t) = g_m(v_G,v_D)\cdot v_G(t) + g_o(v_G,v_D)\cdot v_D(t) + C_{ds}\frac{dv_D}{dt} \tag{3.11}$$

The fundamental period is divided into 128 steps ($\Delta t = {}^{1}/_{128\cdot f_o}$), (the

stepsize was chosen empirically, and depends on the number of harmonics to be included in the analysis). The voltages and currents are computed using:

$$v_F = \frac{1}{C_{gd}} \int i_F dt \tag{3.12}$$

$$v_G = \frac{1}{C_{gs}} \int i_G dt \tag{3.13}$$

$$i_G = \int \frac{di_G}{dt} \tag{3.14}$$

v_D, i_F and di_G/dt are then calculated:

$$v_D = v_G - v_F + R_i \cdot i_G \tag{3.15}$$

$$i_F = C_{gd} \frac{d}{dt} (v_{GS} - v_{DS}) \tag{3.16}$$

$$\frac{di_G}{dt} = \frac{v_{GS} - v_{DS} - (i_G + i_D) \cdot R_s}{L_s} \tag{3.17}$$

$i_D(t)$ is found using (3.11) and di_D/dt is:

$$\frac{di_D}{dt} = \frac{i_D(\Delta t) - i_D(0)}{\Delta t}$$

These steps are repeated over several cycles of the fundamental, until the initial points in $i_D(t)$ are the same for subsequent cycles, indicating that steady state operation has been reached, (this normally requires 4-5 periods). Figures 3.7 and

3.8 show results of a sample calculation for a FET operating near pinch-off (V_{DS} = 1.95V, V_{GS} = -0.5V). In this case convergence at both ports, (defined according to (3.9) with ε=0.05), is reached after only 3 iterations. The choice of input drive level, transmission line impedance and terminating impedances all affect the stability of the circuit. In general, it is best to start with a low drive level and run the simulation several times, increasing the drive level and modifying the terminations as desired. Program execution time is typically not more than 3-4 minutes, so this approach is the most practical way to avoid run-time errors.

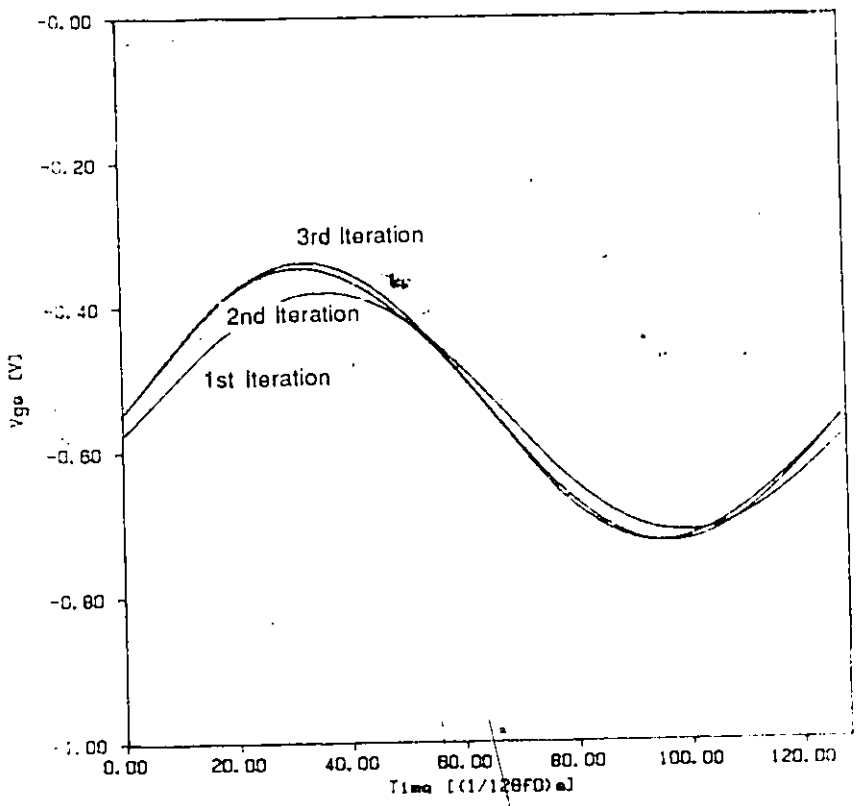


Figure 3.7: Iterative Solution for vGS(t) Using Harmonic Balance

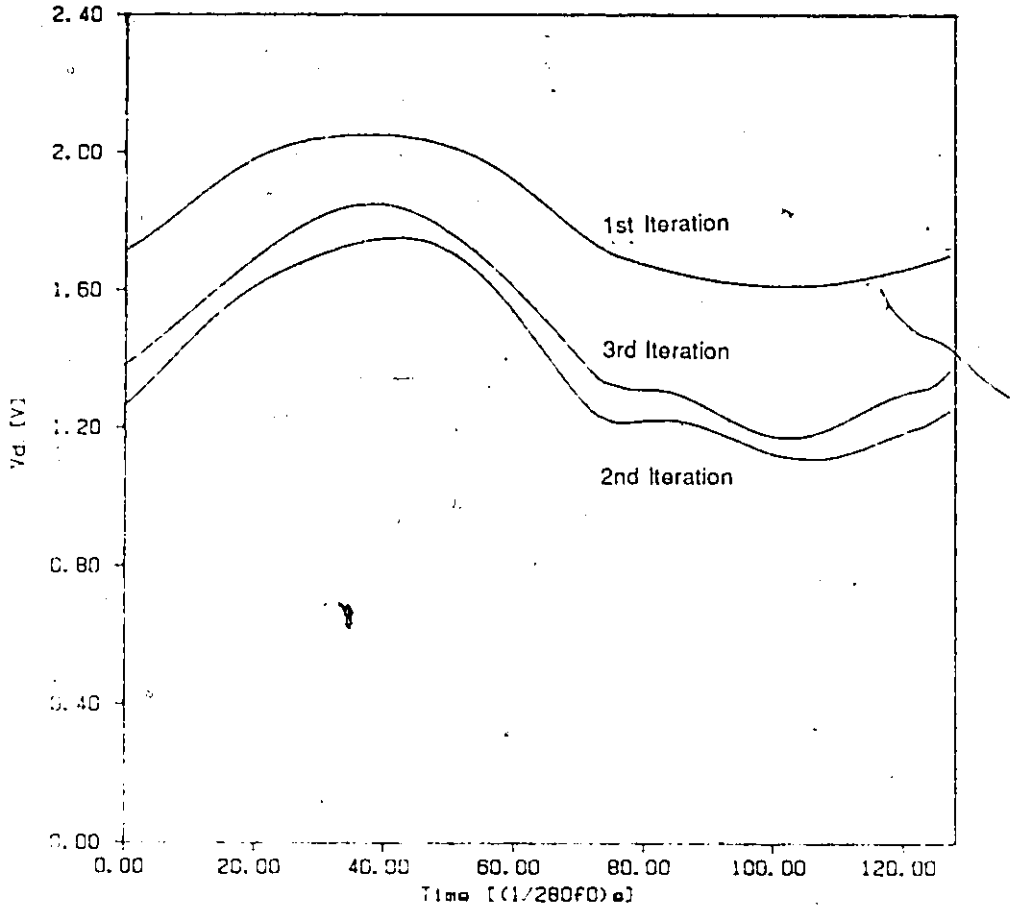


Figure 3.8: Iterative Solution for $v_{DS}(t)$ Using Harmonic Balance

Further examples of outputs from the MHB program are presented in the following chapter to illustrate the effects of teminations and bias point on operation.

Chapter 3 References

[3.1] W. Cunningham, Introduction to Nonlinear Analysis, McGraw-Hill, 1958.

[3.2] M. Nakhla and J. Vlach, "A Piecewise Harmonic Balance Technique for Determination of Periodic Response of Nonlinear Systems," *IEEE Trans. on Circuits and Systems*,**CAS-23**, No.2, .158-159, February 1976.

[3.3] C. Guo et al., "Optimal CAD of MESFETS Frequency Multipliers With and Without Feedback", *1988 IEEE MTT Symposium*, 1115-1118.

[3.4] S. Maas, "Theory and Analysis of GaAs MESFET Mixers", *IEEE Trans. on MTT*, **MTT-32**, No.10, 1402-1411, October 1984.

[3.5] C. Homes, O. Pitzalis and Y. Yuan, "Harmonic Balance Simulates Nonlinear Circuits", *Microwaves & RF*, **Vol. 26**,No. 4, 141-149, October 1987

[3.6] G. Rhyne, M. Steer, "A New Frequency Domain Approach to the Analysis of Nonlinear Microwave Circuits", *1985 IEEE MTT Symposium*, 401-404.

[3.7] W. Curtice, "Nonlinear Analysis of GaAs MESFET Amplifiers, Mixers, and Distributed Amplifiers Using the Harmonic Balance Technique", *IEEE Trans. on MTT*, **MTT-35**, No. 4, 441-447, April 1987.

[3.8] R. Gilmore, "Design of a Novel FET Frequency Doubler Using a Harmonic Balance Algorithm", *1986 IEEE MTT-S Digest*, 585-588.

[3.9] A. Kerr ,"A Technique for Determining the Local Oscillator Waveforms in a Microwave Mixer", *IEEE Trans. on MTT*, **MTT-24**, No. 10, 828-831, October 1975.

4. GaAsFET FREQUENCY DOUBLERS

4.0 Introduction

The expanding use of the millimeter-wave range for radar, communications and EW systems has placed increasingly stringent requirements on stable, low phase-noise frequency sources. At frequencies above 18 GHz, fundamental frequency power generation rapidly becomes more difficult, as traditional solid state sources such as Gunn and IMPATT oscillators have relatively low power and poor stability and tubes (BWO's, klystrons, etc.) are often physically impractical and unreliable. Harmonic generation is an alternative in which power is generated by a stable low frequency source and then multiplied up to the required frequency in one or more stages. In addition, frequency multipliers have a number of applications in synthesizers, phase-locked loops and tracking systems.

Any nonlinear device can be used to produce harmonics. Diodes (abrupt-junction varactors, Snap-Recovery Diodes, etc.) have long been used, although they always introduce some loss. Bipolar junction transistors are preferred at lower frequencies for their stability and conversion efficiency, but their poor reverse isolation restricts their use at frequencies above 4 GHz. The GaAsFET, with its increased isolation and high gain, is the most promising solid-state candidate for frequency multiplication. The first reported FET frequency doubler, (Pan, [4.1]), illustrated the advantages to be gained with this device, producing +13 dBm of power

at 8 GHz with 1 dB gain. Since that time, a number of papers have reported improvements and variations on this application, including octave-bandwidth (Gilmore,[4.2]) and self-oscillating doublers (Rauscher, [4.3]) and operation at 45 GHz using standard FETs (Saito, [4.4]).

4.1 Harmonic Generation

Any device which exhibits a nonlinear transfer characteristic will generate harmonics of an applied signal according to the type of nonlinearity. For example, if the output current, i_o, is related to the input voltage v_i by some arbitrary transfer function:

$$i_o(t) = f(v_i(t)) \tag{4.1}$$

then a Taylor series expansion of (4.1) gives:

$$i_o(t) = f(0) + f'(0)\cdot v_i(t) + f''(0)\cdot v_i^2 + \ldots + \frac{1}{n!} f^n(0)\cdot v_i^n + \ldots\ldots \tag{4.2}$$

where $f'(v_i)=\partial f/\partial v_i$, and the time-dependency has been omitted for simplicity. If $f(v_i)$ is a linear function, all but the first two terms in (4.2) will be zero but, in any other case, higher powers of v_i will appear at the output. Therefore, considering the simplest case, in which a tone $v_i(t) = V\cos\omega t$ is applied, the series expansion of i_o becomes:

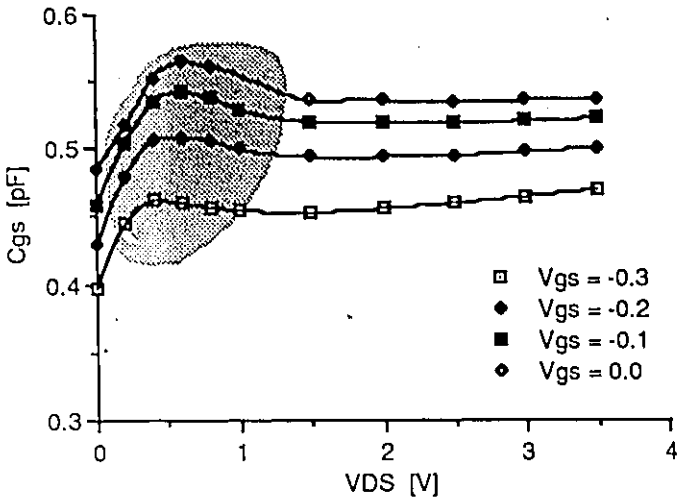
$$i_o = f(0) + f'(0)V\cos(\omega t) + f''(0)V^2\cos^2(\omega t) + \ldots + \frac{1}{n!}f^n(0)V^n\cos^n(\omega t) + \ldots \tag{4.3a}$$

$$i_o = f(0) + f'(0)V\cos(\omega t) + f''(0)\frac{V^2}{2}\cos(2\omega t) + \ldots \tag{4.3b}$$

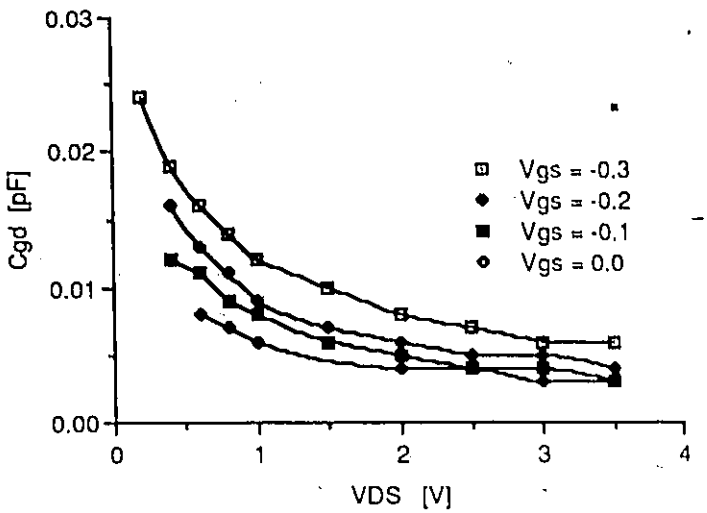
In general, for every non-zero derivative, $\partial^n f / \partial v_i^n$, an nth harmonic will appear at the output. Thus, the relative magnitude of each component depends not only on the form of the input, but on the derivatives of the transfer function as well. The ideal transfer function for xn multiplication would have a strong nth derivative and zero or minimal derivatives of other orders. A square-law device has $\partial^n f / \partial v_i^n = 0$ for n>2 and so is particularly suitable for frequency doubling, but not for generating odd harmonics. Transfer functions with exponential nonlinearities such as diodes have an infinite number of non-zero derivatives, which is why they can be used to realize any degree of multiplication. The drawback is that very careful design is required to enhance the desired harmonic without interference from other relatively strong components. In addition, unwanted harmonics can mix with the operating frequencies at input as well as output, interfering destructively and degrading circuit performance.

4.2 Harmonic Generation with GaAsFETs

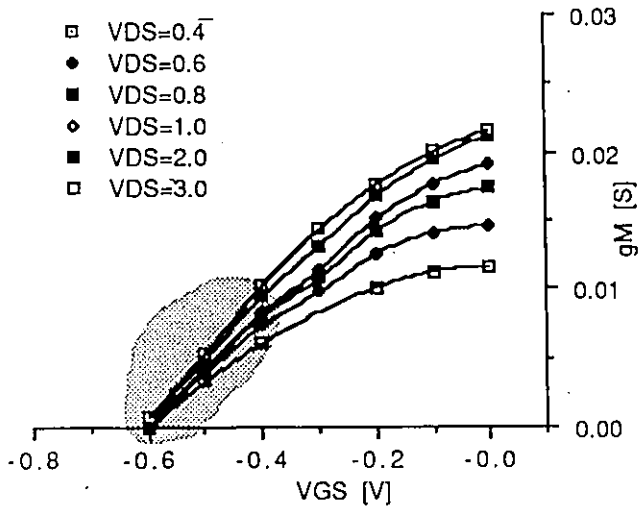
The FET has four principal nonlinearities which can be exploited for harmonic generation: 1) the drain output clipping effect caused by either channel pinchoff or forward conduction through the gate; 2) the transconductance, $g_M(v_G,v_D)$; 3) the output conductance, $g_o(v_G,v_D)$); 4) the gate-channel depletion capacitance, $C_{gs}(v_G,v_D)$) and $C_{gd}(v_G,v_D)$). All these effects are dependent on the intrinsic voltages V_g and V_d across the internal gate-source and gate-drain junctions. Figure 4.1 shows the variations of C_{gs}, C_{gd}, g_M and g_o under various bias



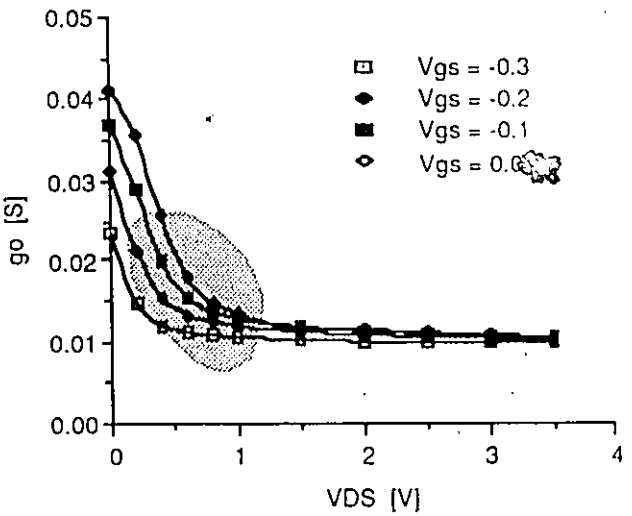
4.1A: Gate-Source Capacitance, Cgs



4.1B: Gate-Drain Capacitance, Cgd



4.1C: Transconductance



4.1D: Output Conductance

Figure 4.1: Principal Nonlinearities in GaAsFET

conditions as calculated for a typical GaAsFET (the NEC673) using the program described in Chapter 2, with the regions of greatest nonlinearity shaded. The designer can enhance or minimize these elements by altering the device bias point and the output load line, so the relative contributions of each effect should be assessed to determine the optimum operating conditions.

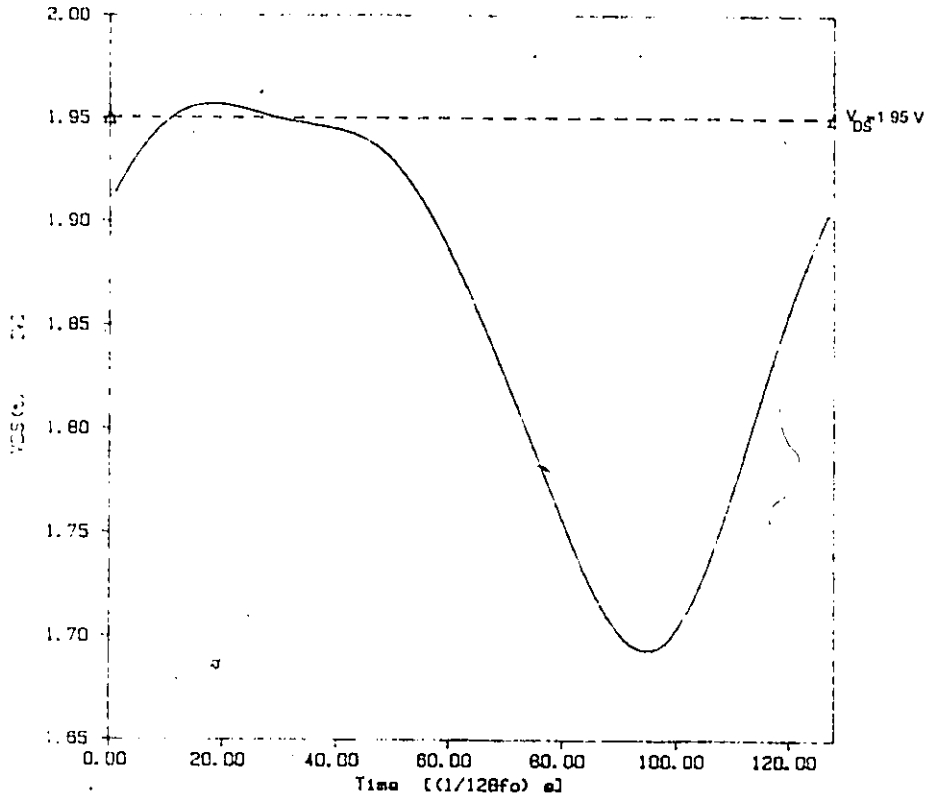


Figure 4.2: Output Clipping

(MHB Simulation for NEC673 Biased at V_{GS}=-0.1, V_{DS}=1.95)

4.2.1 Drain Current Clipping

Biasing the device close to pinchoff ($V_{gs} \approx V_p$) or forward conduction ($V_{gs} \approx 0$) causes the output signal to be clipped in the manner suggested in Figure 4:2. The

Fourier series representation of an ideal reactified sine wave is:

$$f(t) = \frac{1}{\pi} + \frac{1}{2}\sin\omega t - \frac{2}{\pi}\sum_{n}^{\infty} \frac{1}{(n^2-1)}\cos n\omega t \ , \ n=2,4,6,..... \qquad (4.4)$$

which contains only even-order harmonics. Neglecting all other effects, the second harmonic is seen to be $20\log(4/3\pi)$ or 7.4 dB below the fundamental at the output. In fact, the actual conversion gain will depend on both the amplification available from the device at f_o and $2f_o$, and the input voltage developed across C_{gs}, as well as the device-circuit interaction discussed in section 4.4. Using $V_{gs}=V_p$ is usually favoured, as it may improve DC-RF efficiency and avoids the possibility of burning out the FET. As Figure 4.1 shows, biasing near $V_{gs}=0$ provides greater transconductance, but biasing near $V_{gs}=V_p$ provides a smaller value of C_{gs}, hence a larger input voltage. Which effect predominates depends on both the device and the operating frequency, so it is not surprising that various studies, [4.5-4.7], have produced different results as to the best operating regime.

4.2.2 Tranconductance, g_m and Output Conductance, g_o

The contribution of the transconductance g_m can be estimated by assuming a square-law I_{DS}-V_{GS} transfer function as shown in Figure 4.3, (which is based on measured measured results for a NEC673). As an illustration, the drain current can be approximated using Curtice's model, [4.8]:

$$I_{DS} = (V_{GS}-V_p)^2 \cdot \left[\beta\tanh(\mu V_{DS} - \frac{G_D V_{DS}}{V_p^2}\right] \ , |V_{GS}| < V_p$$

$$= 0 \qquad , |V_{GS}| > V_p \qquad (4.6)$$

G_D is the measured output conductance and β and μ are curve-fitting parameters. Assuming a constant G_D (which is reasonable at large V_{ds}), and an input signal $v_{GS}(t) = (V_o + V\cos\omega_o t)$, (4.6) may be rewritten:

$$I_{DS} = \left[(V_o - V_p) + 2(V_o - V_p)V\cos\omega_o t + \frac{V^2}{2}(1+\cos 2\omega_0 t)\right] \cdot \left[\beta\tanh(\mu V) + \frac{G_D V_{DS}}{V_p^2}\right] \qquad (4.7)$$

from which the relative magnitude of the second harmonic is:

$$\frac{V^2}{4(V_o - V_p)}$$

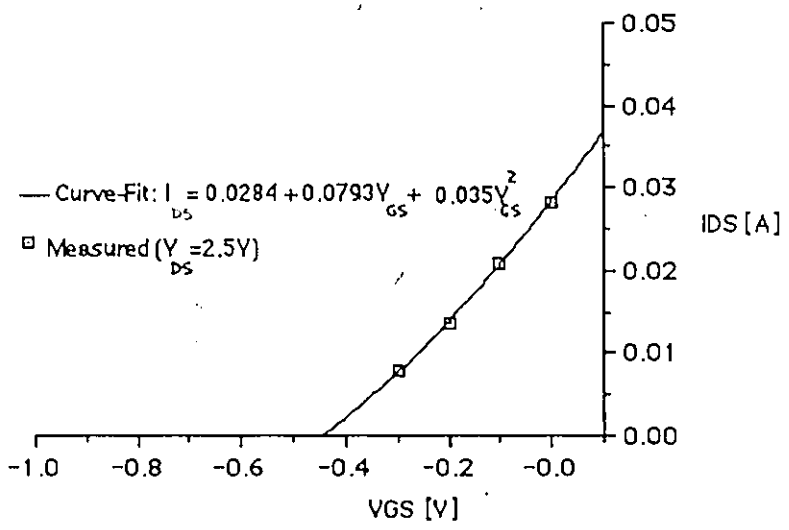


FIGURE 4.3: Square Law I_{DS}-V_{GS} Characteristic

Equation (4.7) indicates that operating near pinchoff favors doubling with g_m, although the exact contribution of this nonlinearity depends on the magnitudes of the input, pinchoff and bias voltages. The assumption that G_D is constant for a fixed

drain voltage is not strictly valid, so (4.7) serves only as a guideline.

By biasing at $V_{GS} \approx 0$, the second harmonic with respect to the fundamental at the output becomes $V/4V_p$. In this mode of operation, clipping is the dominant nonlinear effect. The transconductance is significant only for the amplication it provides. The output conductance makes some contribution when the drain is biased near the 'knee' of the characteristic ($V_{DS} \approx 0.9$ V in Figure 4.1D). However, g_o on its own has not been found to be an important factor in multiplication, due to the presence of parasitics, (Willing *et.al.*,[4.9]).

4.2.3 Gate-Channel Depletion Capacitance, C_{gs} and C_{gd}

The input depletion capacitance C_{gs}, together with the channel resistance R_i, can act as a lossy varactor. While application of the Manley-Rowe equations, [4.10], shows that varactor multipliers can theoretically achieve 100% conversion efficiency, using the results developed by Penfield and Rafuse,[4.11], for varactor multipliers, it can be estimated that a typical submicron FET would produce a $P_{out}(2f_o)$ 15 to 20 dB below $P_{out}(f_o)$, (Gopinath and Rankin, [4.12]). These simulations suggest that the varactor effect could decrease conversion loss by up to 2 dB, but it is generally regarded as a secondary effect during design. The simulated $C_{gs}(v_G, v_D)$ shown in Figure 4.1A confirms that GaAsFET's do not make good varactors, as the capacitance does not display the strong nonlinearity required for efficient harmonic generation. The gate-drain capacitance, $C_{gd}(v_G, v_D)$ shows somewhat weaker nonlinear behaviour for the particular device shown here. It has been suggested by Curtice, [4.13], that this is probably due to the greater separation

of the gate electrode from the drain as compared to the source.

4.3 Design Considerations

The effect of internal feedback can be pronounced, even in submicron-gate FETs, particularly at millimeter-wave frequencies. This, together with the forward transmission gain, means that circuit interactions at both $2f_0$ and f_0 at input and output must be considered. The input termination at f_0 and the output termination at $2f_0$ are important for the same reason they are in amplifer design: achieving maximum power transfer through a conjugate match. However the input termination at $2f_0$ and the output at f_0 are at least as crucial to the design. Any portion of the second harmonic which is fed back through the device may be amplified and reflected back to the output, where it interferes destructively or constructively with the signal. At high frequencies, the device may not have sufficient gain for this to be significant. However, when enough gain is available, proper choice of input termination can significantly improve performance, (Rauscher, [4.14]). Self-oscillating doublers have been realized by reinforcing internal parasitics with an external feedback path. Studies such as [4.14] have shown that device-circuit interactions at the two frequencies are essentially independent of each other, which permits relatively straightforward design. The main problem lies in realizing terminations.

At low frequencies, harmonic separation may be done with high-Q resonant traps, but there are practical difficulties in achieving this using microstrip or finline techniques. One solution is the balanced topology shown in Figure 4.4. Since

the combination point represents a virtual ground at f_o and all odd harmonics, any purely reactive termination required can be presented at the drain simply by adjusting a length of transmission line. (A method of matching which is transparent to the even harmonics.) Output matching at $2f_o$ placed beyond this point will effectively be screened from the fundamental. A FET doubler based on this sort of approach has been reported by Stancliff, [4.15].

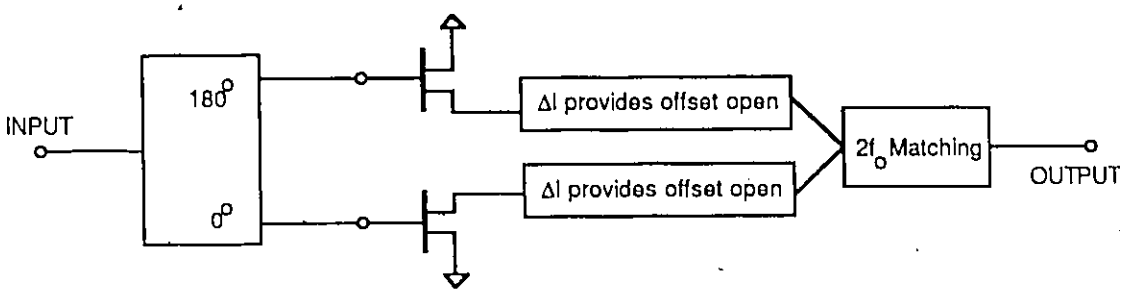


Figure 4.4: Balanced Doubler Configuration

The optimum output termination for the fundamental depends on which nonlinearity is to be maximized. In virtually all cases, output clipping is the predominant effect, i.e. the FET is acting as a half-wave rectifier. Maximum conversion efficiency is obtained by presenting an open circuit to f_o at the output, (offset by the conjugate of the device output reactance), resulting in the largest possible voltage swing. On the other hand, if g_m is being used as the main source of nonlinearity, a short circuit at the fundamental provides the best termination, as suggested by (4.7). This is supported by simulations run using the MHB program with a modified FET model. Figure 4.5 shows the effects of output termination at the

fundamental on two types of multipliers. In the first case, the only nonlinearity was provided by g_m and, in the second, output clipping. All other factors (input power level, other harmonic terminations at input and output, etc.) were the same for both runs.

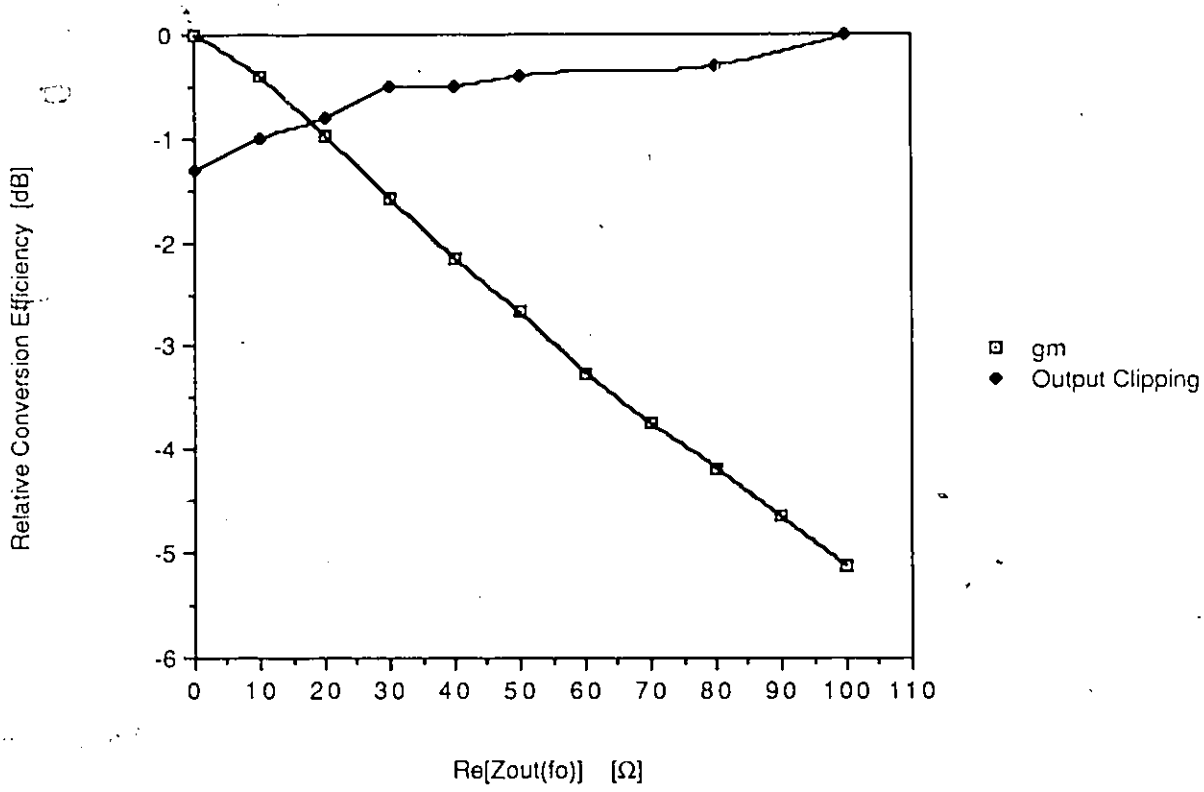


Figure 4.5: Effect of Fundamental Output Termination on DoublerPerformance
(MHB Simulation on Modified FET)

Chapter 4 References

[4.1] J.J. Pan, "Wideband MESFET Frequency Multiplier", *1978 IEEE MTT Symposium Digest*, 306-308

[4.2] R. Gilmore, "Concepts in the Design of Frequency Multipliers," *Microwave Journal*,**30**, No.3, 129-139, March 1987.

[4.3] C. Rauscher, "Frequency Doublers with GaAs MESFETs", *1982 IEEE MTT Symposium*, 280-282.

[4.4] T. Saito *et al.*, "A 45 GHz GaAsFET MIC Oscillator-Doubler", *1982 IEEE MTT Symposium*, 283 -285.

[4.5] C. Rauscher, "High-Frequency Doubler Operation of GaAs Field-Effect Transistors", *IEEE Trans. on MTT*, **MTT-31**,No. 6, 462-473, June 1983.

[4.6] E. Camargo *et al.*, "Sources of Non-Linearity in GaAs MESFET Frequency Multipliers", *1983 IEEE MTT Symposium Digest*, 343-345.

[4.7] M. Gupta, R. Laton, T. Lee, "Performance and Design of Microwave FET Harmonic Generators", *IEEE Trans. on MTT*, **MTT-29**,No. 3, 261-263, March 1981.

[4.8] W. Curtice, "A MESFET Model for Use in the Design of GaAs Integrated Circuits", *IEEE Trans. on MTT*, **MTT-28**,No. 5, 448-456, May 1980.

[4.9] H. Willing et al., "A Technique for Predicting Large-Signal Performance of a GaAs MESFET", *IEEE Trans. on MTT*, **MTT-26**, No. 12, 1017-1023, December 1978.

[4.10] J. Manley, H. Rowe, :Some General Properties of Nonlinear Elements: Part I - General Energy Relations", *Proc. IRE*, **44**, 904, 1956.

[4.11] P. Penfield, R. Rafuse, Varactor Applications, MIT Press, Cambridge, MA., 1962.

[4.12] A. Gopinath and J.B. Rankin, "Single-Gate MESFET Frequency Doublers", *IEEE Trans. on MTT*, **MTT-30**,No. 6, 869-875, June 1982.

[4.13] W. Curtice, 1980, *op. cit.*

[4.14]C. Rauscher, 1983, *op. cit.*

[4.15] R. Stancliff, "Balanced Dual-Gate GaAsFET Frequency Doublers". *1981 IEEE MTT Symposium*, 143-145.

5. DESIGN AND REALIZATION OF A GaAsFET FREQUENCY MULTIPLIER

5.0 Introduction

The 18.75/37.5 GHz frequency doublers designed and tested by the author are described in this chapter. A number of GaAsFET multipliers are reported in the literature [5.1-5.4], although most of these operate at lower frequencies and none report using finline (see also Chapter 4 references). As this work was undertaken as part of a study into the potential of E-plane media for millimeter-wave applications, two versions were developed and compared: the first was an all-finline design, using a FET mounting technique developed at C.R.C.; the second combined suspended microstrip for the matching circuits with antipodal finline for the waveguide transitions. (The results and relative merits of the two approaches are presented in Chapter 6).

5.1 Finline Doubler

A method of achieving stable FET operation in finline developed at CRC by Jean L'Ecuyer [5.5],was used in this design. Two rectangular waveguides (WR42 input, WR22 output) are broadwall-coupled via a small (0.030" x 0.040") opening in which the FET is mounted. A unilateral finline taper is used to transition from the input port, concentrating the field in a 0.006" gap at the gate of the device. The drain is connected across a similar gap which in turn is tapered out to the WR22 port. The source leads of the FET are grounded to the septum dividing the two

waveguides, which has plated-through holes to maintain electrical continuity. Figure 5.1 shows the circuit layout and Figure 5.2 show a cross-section of the circuit and housing at the point where the FET is mounted.

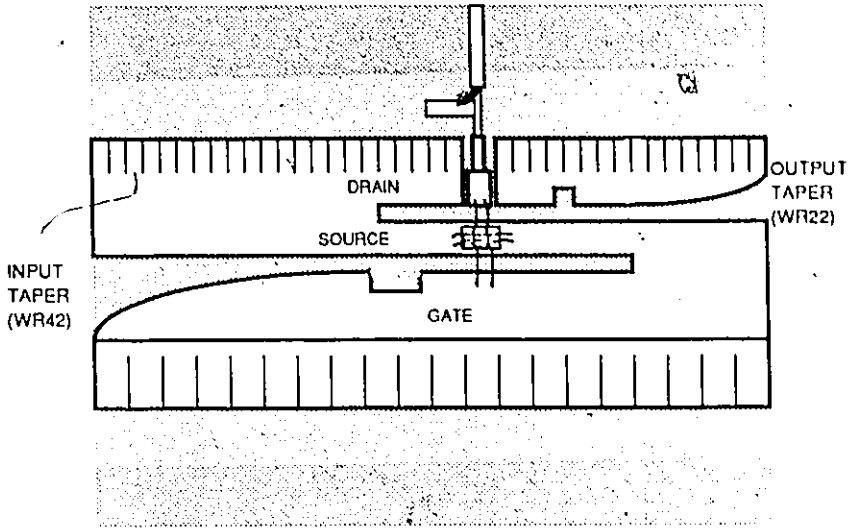


Figure 5.1: Finline Doubler Layout (Not to scale)

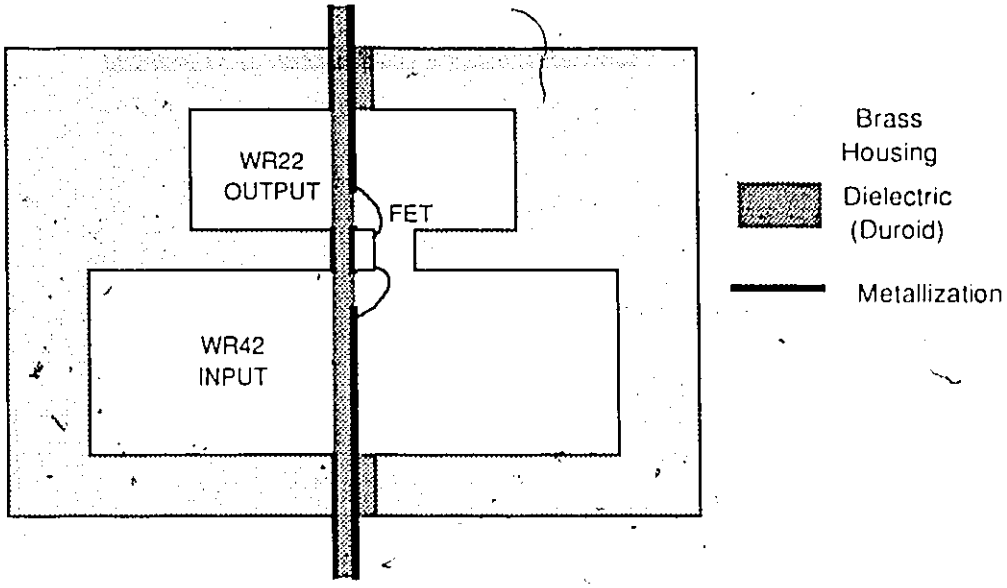


Figure 5.2: Cross-Sectional View of Housing and Circuit

5.1.1 Finline Doubler Circuit Configuration

Since it was not possible to characterize the FET directly in the finline medium, due to the lack of reliable calibration standards, the preliminary matching was based on a small-signal model developed from the manufacturer's S-parameters.

Analysis of the unilateral finline elements was based on a simplified spectral domain analysis using 'UNIFIN'. This is a program written by the author to calculate the cutoff frequency and propagation constant of a symmetrical finline structure using the standard spectral domain technique. The subroutines used in determining the power-voltage impedance were developed by another student at the University of Ottawa, Solange Monnier. The results are applicable to the asymmetrical case, as shown below in figure 5.3, providing the result for Z_o is divided by 2. A listing of UNIFIN is provided in Appendix III.

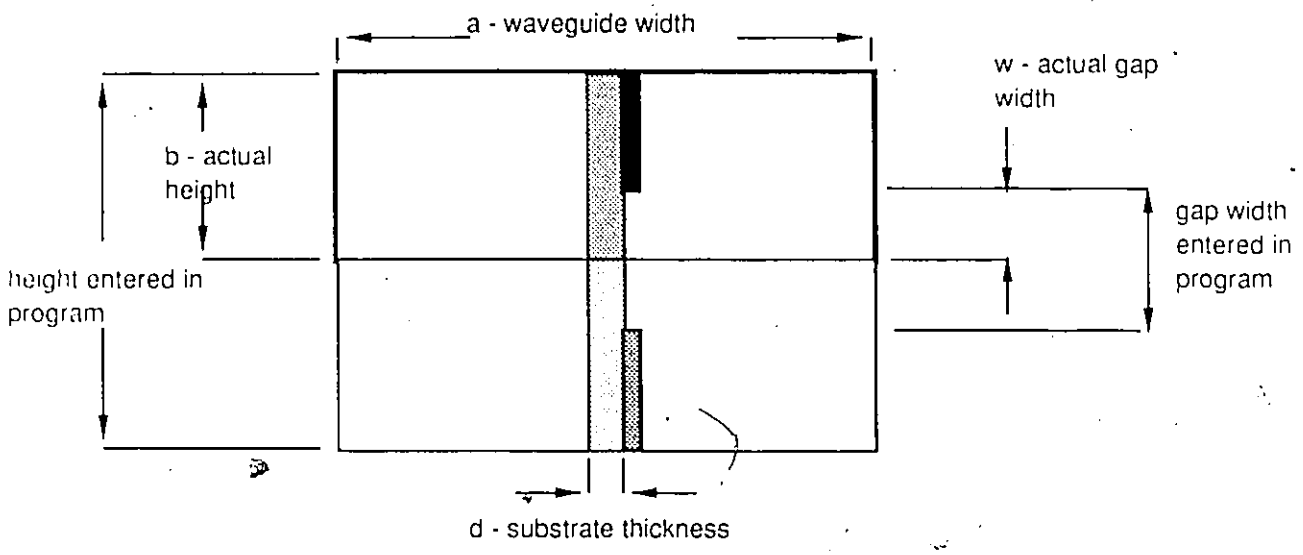


Figure 5.3: Adapting UNIFIN to Asymmetrical Finline Structure

It seemed advantageous to keep the input and output gaps as narrow as possible, since they had to be spanned by the device bondwires. A width of 0.006" was chosen for the input, corresponding to an impedance of 77Ω at f_o and 82Ω at $2f_o$. The output gap (in WR22) was 0.005", or 90Ω at f_o and 84Ω at $2f_o$. Lower impedance levels would have been desirable, but required impractically narrow gaps.

The design is particularly sensitive to bias circuit response: the bias network should provide loss at lower frequencies to damp any instability in the FET (the NEC673 is potentially unstable up to about 14 GHz), but also be transparent at the design frequencies so as not to degrade performance. A preliminary doubler circuit was measured over a variety of bias conditions and showed that, in these conditions, V_{GS}=0 provided the best operating point. The gate bias circuit was thus excluded from the final design, and the input was left DC-coupled to ground.

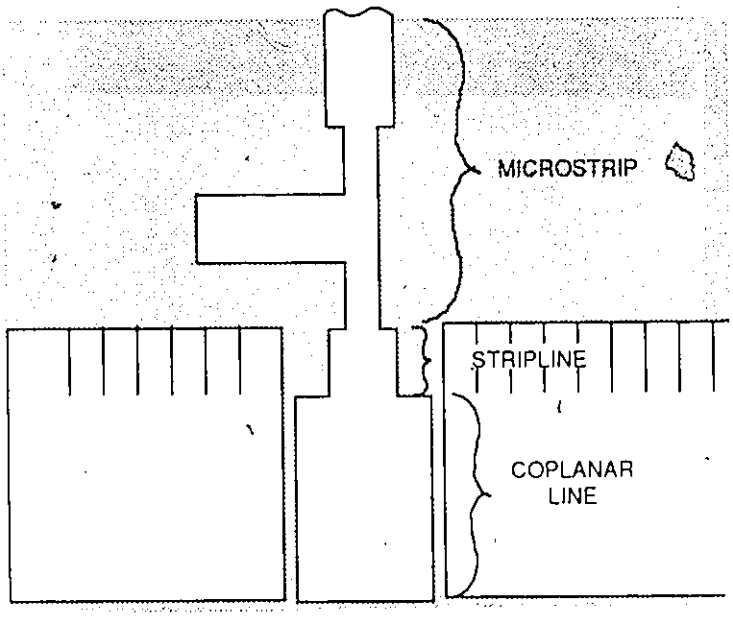


Figure 5.4: Close-Up of Bias Circuit

The drain bias circuit uses the standard quarterwave open-circuit stub to present an open in shunt with the device. However in this case the stub, which is microstrip, must be transformed through slotline and coplanar line to maintain DC isolation from the fin, as seen in figure 5.4. The coplanar section is designed to be λ/2 long, so the short seen at the waveguide wall is translated to the input of the drain circuit, providing continuity along the fin gap. The electrical characteristics for each line type were optimized on SUPERCOMPACT™ and then translated into physical parameters.

The topology of the finline circuit restricted the type and placement of matching elements available. Figure 5.5 illustrates some of the possibilites: terminating the gap across which the device is connected creates a shunt short-circuit stub; Burton and Hoefer, [5.6], have shown that a a series short-circuit stub is created by an abrupt, large increase in gap width, provided the step width is narrow (≤ 0.1λ); if the step is a significant fraction of a wavelength long or is not much greater than the original gap width, the effect is just of an impedance step, and can be used for quarterwave transformers.

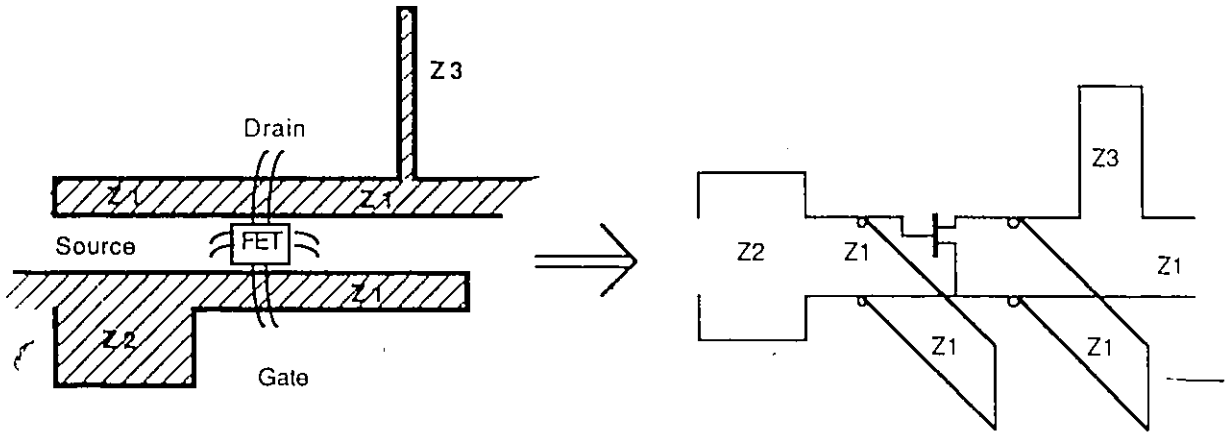


Figure 5.5: Finline Matching Elements

5.2 Microstrip/Finline Doubler

The components required in this design include two waveguide to microstrip transitions, input and output matching circuitry and bias circuits. The development and integration of these elements onto a 0.870"x3.000" Duroid substrate is detailed below, together with a description of the housing. A photograph of the completed doubler is shown in figure 5.6.

Figure 5.6: Microstrip/Finline Doubler

5.2.1 Transitions

The use of an integrated transition has several advantages, particularly for high-frequency applications. It requires no machining or elaborate housing, whose tolerances can be difficult and expensive to maintain for the small dimensions needed at 37.5 GHz. By employing antipodal finline transitions, the circuit layout could be made 'in-line', i.e. with the waveguides end-coupled as opposed to broadside-coupled. This simplified both housing and circuit layout. In addition, results reported for

such transitions, [5.7-5.8], indicate that they provide good return and transmission loss.

Antipodal finline is used to gradually rotate the E-field through 90 degrees, simultaneously concentrating it into the overlapping region of the two fins, which provide a balanced transmission line. A balun section permits the field to transition to the unbalanced microstrip section. Figure 5.7 illustrates these various areas of the transition, together with cross-sections which show the field distributions.

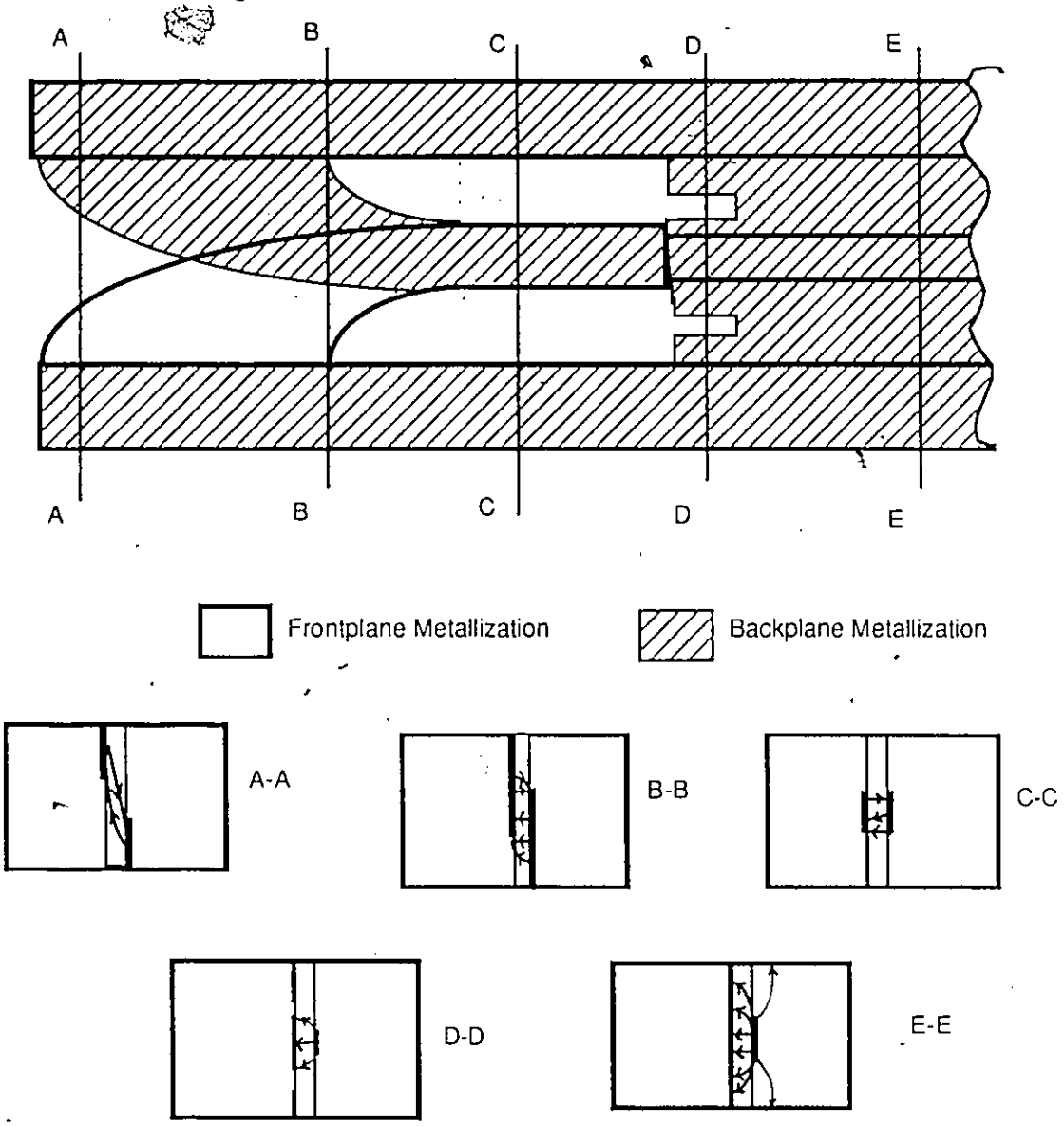


Figure 5.7: Waveguide-Microstrip Transition Using Antipodal Finline

The microstrip line (E-E) was chosen to 50 Ω, as this is a standard value for matching. The effect of the lower half of the waveguide is negligible, due to the presence of the ground-plane. Propagation through the waveguide in this region is inhibited, since the cutoff frequency in the lower half at the output is 53 GHz.

The balun section (D-D) must provide the transition between an asymmetrical structure (microstrip) to a symmetrical structure (the balanced line C-C). Two quarterwave shortcircuited stubs present an open-circuit at the ground plane discontinuity. An in-house computer program ('GPLINES') was used to characterize the coplanar line: the gaps were made 0.010" wide, corresponding to 50Ω. The input stubs were 0.150" long ($\lambda/4$ at f_o) and the output, 0.075", ($\lambda/4$ at $2f_o$).

The impedance of the balanced transmission line (C-C) is chosen so as to match the microstrip, and is approximated by considering its capacitance per unit length, neglecting field fringing and enclosure effects, (Mizuno *et.al.*,[5.9]). For an impedance of 50 Ω on 0.010" RT-Duroid, (ε_r = 2.22), the corresponding line width is 0.050". The balanced line was made one wavelength long to provide adequate separation between the ground-plane discontinuity and the transition tapers.

The input and output tapers create a gradual transition from the b dimension of the empty waveguides to the 0.050" balanced line sections. This is accomplished over 1.5 wavelengths, using an exponential taper profile. The taper should be as short as possible, while maintaining low mismatch reflections. The exponential

taper is calculated using

$$y = \frac{y_e^x}{x_e^\beta}$$

where x_e, y_e are the terminal point of the taper and ß defines the taper profile. The values selected for β were 3 and 2.5 for the outer and inner tapers, respectively. Plated-through holes are used to ensure continuity in the broad wall of the waveguide, where the substrate is clamped. As the field becomes concentrated between the two fins, electrical contact no longer needs to be maintained with the walls, (permitting the use of the balanced line structure). The concentration of the field in the center of the waveguide means that fewer plated-through holes are required to prevent radiation loss in section B, C, D and E, reducing fabrication costs.

The back-to-back transitions as illustrated in Figure 5.8 were fabricated and measured for both the WR22 and WR42 dimensions. The results are given in Chapter 6.

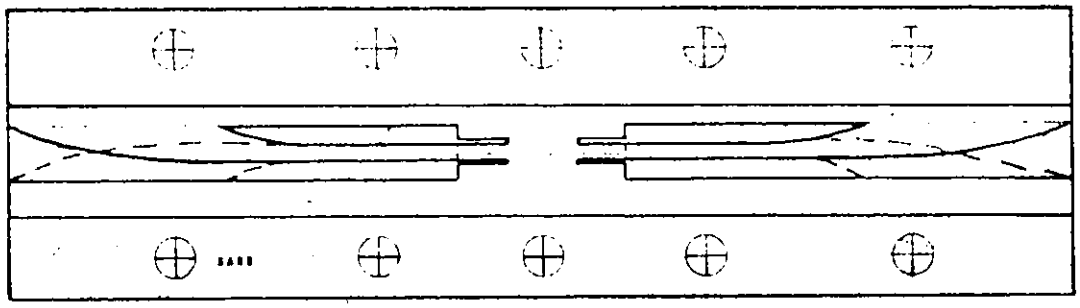


Figure 5.8: Back-to-Back Transition (WR42)

5.2.2 Matching and Bias Circuits

Estimated S-parameters from the FET model predicted that the input match at f_o could be done using an open-circuit stub. Initial values were obtained using a Smith Chart and the results from the small-signal model at 18.75 GHz. Since microstrip was being used, a SUPERCOMPACT optimization was done to calculate final values, taking into account substrate parameters and discontinuity effects. The constraint on housing width (WR42) made it necessary to use a lower impedance stub (30 Ω) , so that it would be short enough to fit. The output match at the second harmonic was done using a low-impedance quarterwave transformer.

Since preliminary results for the first doubler had shown that the best efficiency was obtained when V_{GS} = 0, this doubler was designed for the same operating point. Accordingly, the input was left DC-doupled to the waveguide wall. The drain bias circuit was of the standard type: an open-circuit stub, transformed through a 3/4 λ high-impedance line causes the bias network to be transparent at $2f_o$. A series RC branch is connected in parallel to the stub to provide loss at lower frequencies, in the region of potential instability for the FET. A 0.3 pF capacitor was used for DC blocking on the output line.

The bias circuit must be introduced in the circuit so as to have minimal effect at the frequencies of interest, while presenting lossy termination to lower frequencies. The optimum was location for the bias circuit was found using SUPERCOMPACT™. Varying the position of the junction has a significant effect on

output peformance. In this case, the bias branch was placed 0.071" from the drain (0.20λ at f_o).

The housing required for this design was straightforward, consisting of WR42 waveguide at the input with a step transistion to WR22 the output port. The step in width occurs in the microstrip portion of the circuit, so it has minimal effect on operation. As in the first design, fundamental rejection at the output is achieved without filtering by the waveguide beyond cutoff.

During testing, the top half of the housing was made with a removable portion which exposed the area around the FET for tuning, but left the finline transitions uncovered. Thus adjustments could be made with very little disturbance to circuit operation. Figure 5.9 shows a closeup of the FET mounted in the circuit.

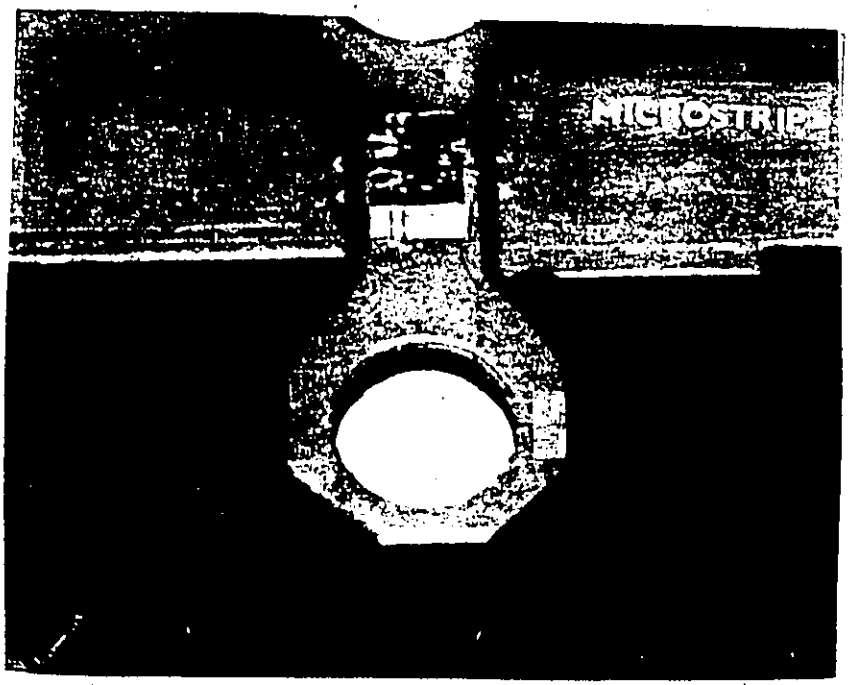


Figure 5.9: FET Mounted in Microstrip/Finline Doubler

Chapter 5 References

[5.1] R. Gilmore, "Octave-Bandwidth Microwave FET Doubler", *Electronics Letters*, **21**, No. 12, June 1985.

[5.2] P. Chen, P. Wang, "Dual-Gate GaAs FET As A Frequency Multipier At Ku Band", 1978 *IEEE MTT Symposium*, 309-311.

[5.3] F. Mantovani, "Active MESFET multipliers solve low signal levels", *Microwaves & RF*, **23**, No: 8, 129-131, August 1984.

[5.4] P. Chen, C.-T. Li, P. Wang, "Performance of a Dual-Gate GaAs MESFET as a Frequency Multipler at Ku-Band", *IEEE Trans. on MTT*, **MTT-27**, No.5, 411-415, May 1975.

[5.5] J. L'Ecuyer, G. Gajda, W. Hoefer,"A FET Amplifier in Fin-Line Technique", *1986 IEEE MTT Symposium*, 287-290.

[5.6] M. Burton, W. Hoefer," An Improved Model for Short- and Open-Circuited Stubs in Finlines", *1984 IEEE MTT Symposium*, 330-332.

[5.7] J. van Heuven," A New Integrated WaveguiDe-Microstrip Transition:, *IEEE Trans. on MTT*, **MTT-24**, No. 3, 144-146, March 1976.

[5.8] L. Lavedan," Design of Waveguide-to-Microstrip Transitions Specially Suited to Millimetre-Wave Applications", *Electronics Lett.*, 1977.

[5.9] H. Mizuno *et. al.*," Propagation in Broadside-Coupled Suspended-Substrate Stripline in E-Plane", *IEEE Trans. on MTT*, **MTT-33**, No. 10, 948-951, October 1985.

6. MEASUREMENT AND PERFORMANCE OF THE FREQUENCY MULTIPLIERS

6.0 Introduction

After fabrication, the input and output return loss for each doubler was measured. Due to variations in the FET mounting technique as compared to the device model, tuning was required, after which the circuits were remeasured. The drain bias and match at f_o were adjusted for minimum conversion loss. The signal power transfer characteristic was measured, as was conversion loss over a range of input frequencies and power levels.

6.1 Finline Doubler Performance

The conversion loss ($P_{in}[f_o]$:$P_{out}[2f_o]$) was initially measured at about 13 - 14 dB. Tuning was done through a window in the narrow wall of the waveguide located above the FET. Removal of the window cover had little effect on circuit response because at that point in the circuit the energy is largely concentrated between the fins in the centre of the guide. The lengths of the short-circuit stubs and the impedance step sections were altered using small tabs of 0.010" Kovar, thus modifying circuit performance. Doubler performance before and after tuning is shown in Figure 6.1. Frequency response and input return loss are plotted in Figures 6.2 and 6.3, respectively.

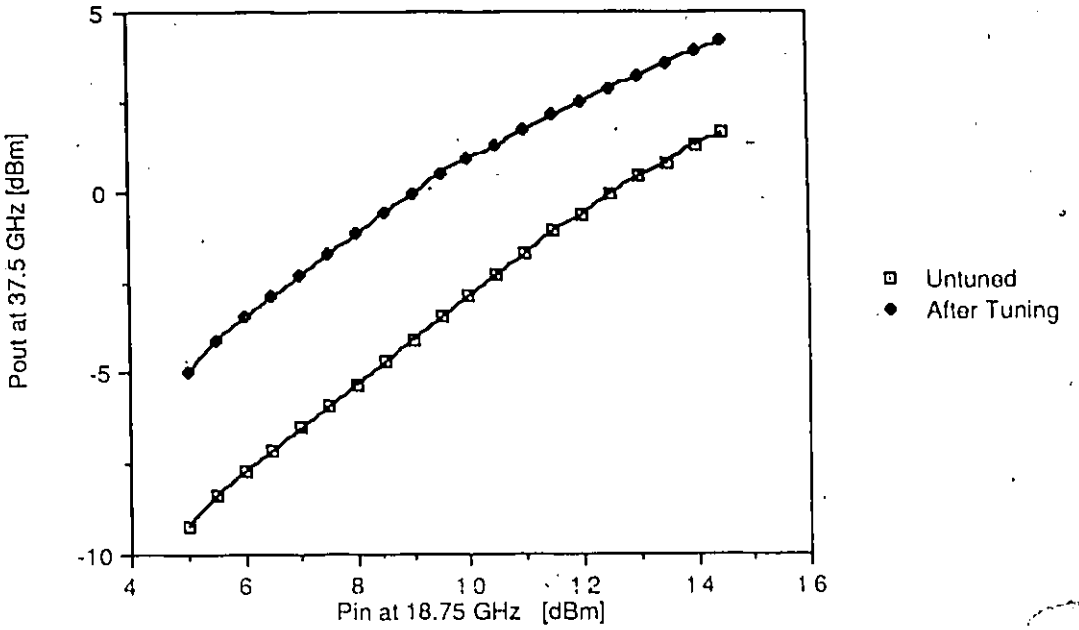


Figure 6.1: Finline Doubler Performance

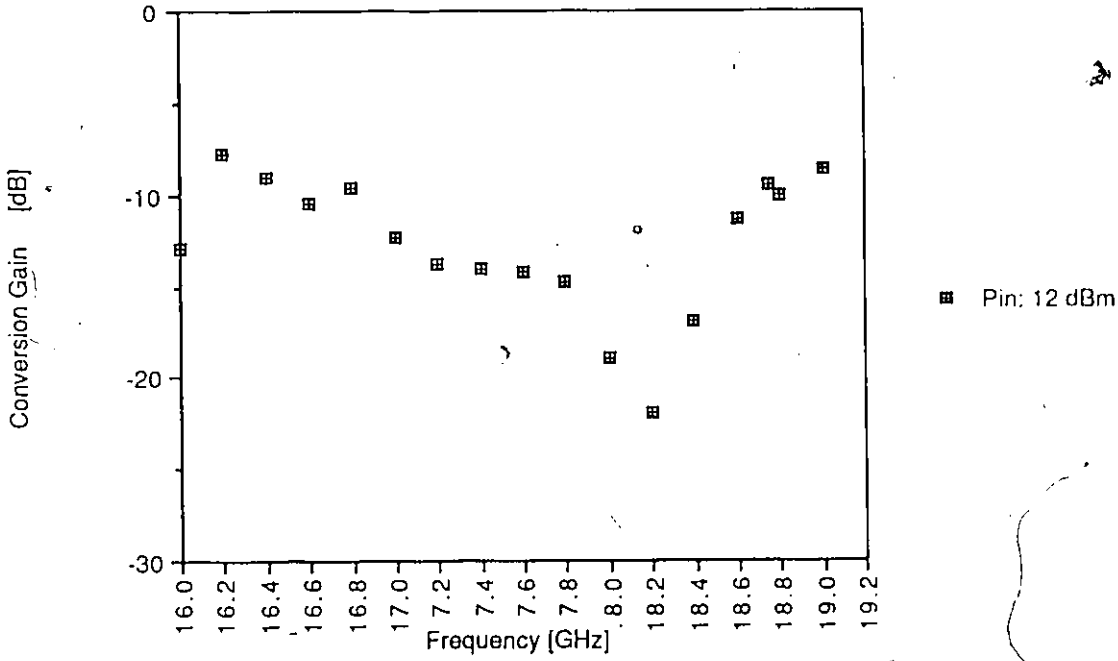


Figure 6.2: Conversion Gain of Finline Doubler Over Frequency

6.2 Microstrip/Finline Doubler

6.2.1 Waveguide-to-Microstrip Transitions

The antipodal finline transitions described in section 5.2.1 were measured separately, (in the back-to-back configuration as shown in Figure 5.8), for return and transmission loss. The results for the WR42 transition are depicted in Figures 6.4 and 6.5. It was found by experimenting with the length of the balun ground plane slots that the calculated value of 0.150" produced the best input match. The measured S_{11} and S_{21} for the WR22 transition are shown in Figure 6.5. In both cases, the transitions exhibited satisfactorily low loss, particularly at 37.5 GHz. The antipodal-microstrip transition thus seems to be a simple, easily-designed alternative for realising waveguide-compatible components for high-frequency applications.

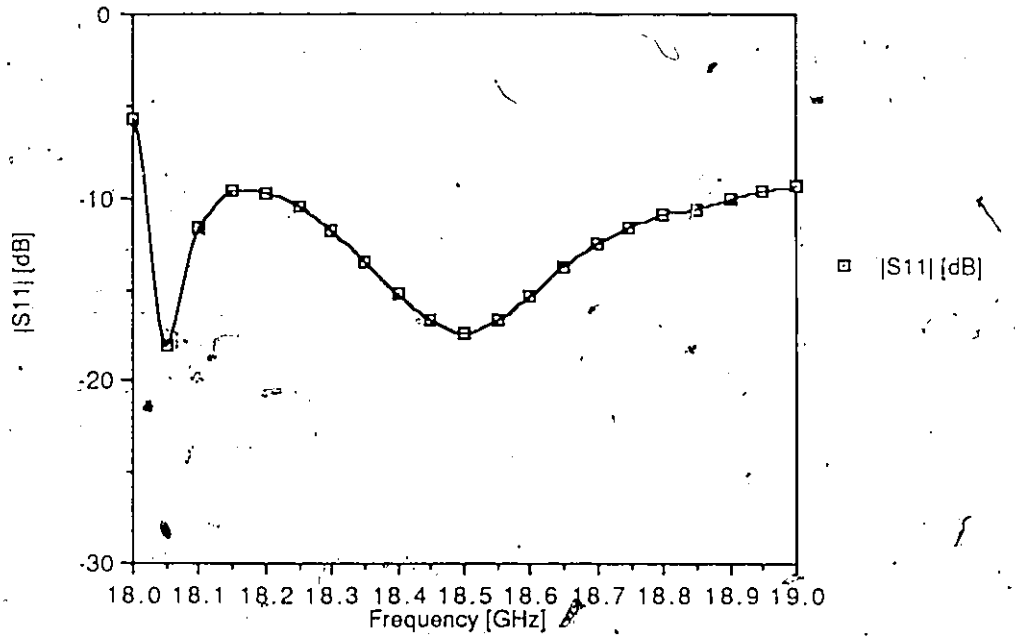


Figure 6.4: Return Loss for WR42 Transitions

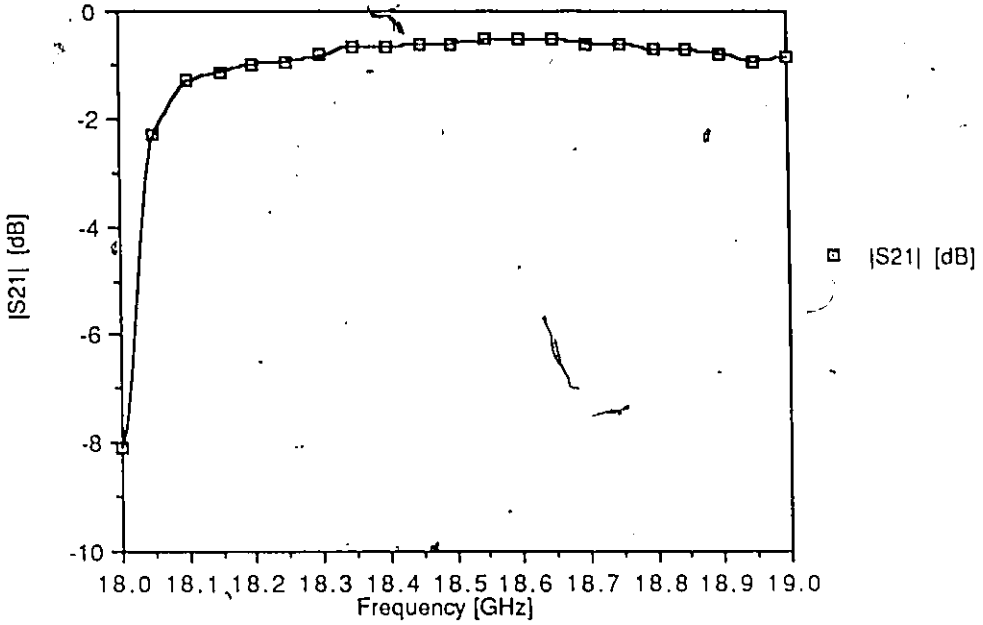


Figure 6.5: Transmission Loss for 2 WR42 Transitions

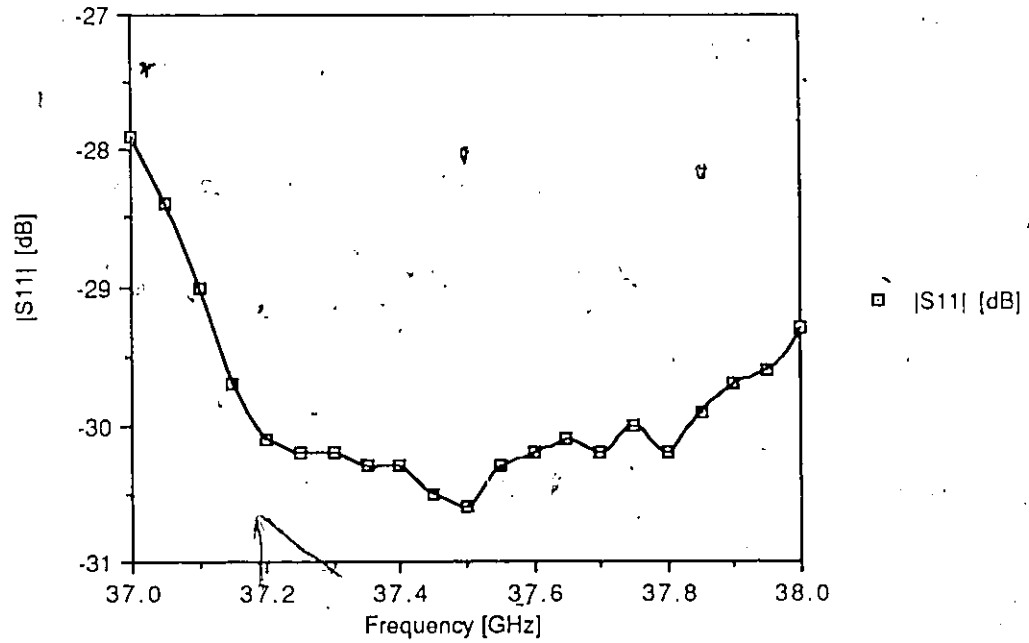


Figure 6.6: |S11| for WR22 Transitions

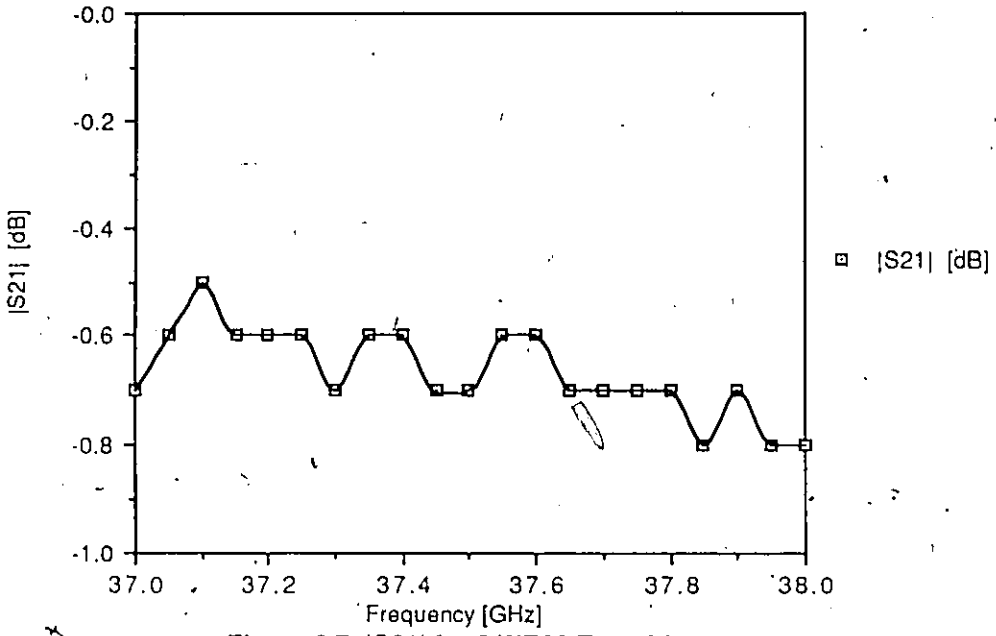


Figure 6.7: |S21| for 2 WR22 Transitions

6.2.2 Input and Output Match

When the assembled multiplier was measured,the input match was found to occur at 16.2 GHz instead of the design frequency. This was due to the method of mounting the FET : the device surface is 0.006" above the substrate, resulting in loops in the bondwires which increase their inductance. In addition, this technique required longer bondwires than had been expected. A Kovar tab was used at the input to tune out this effect. Figure 6.8 is a comparison of the results before and after tuning.

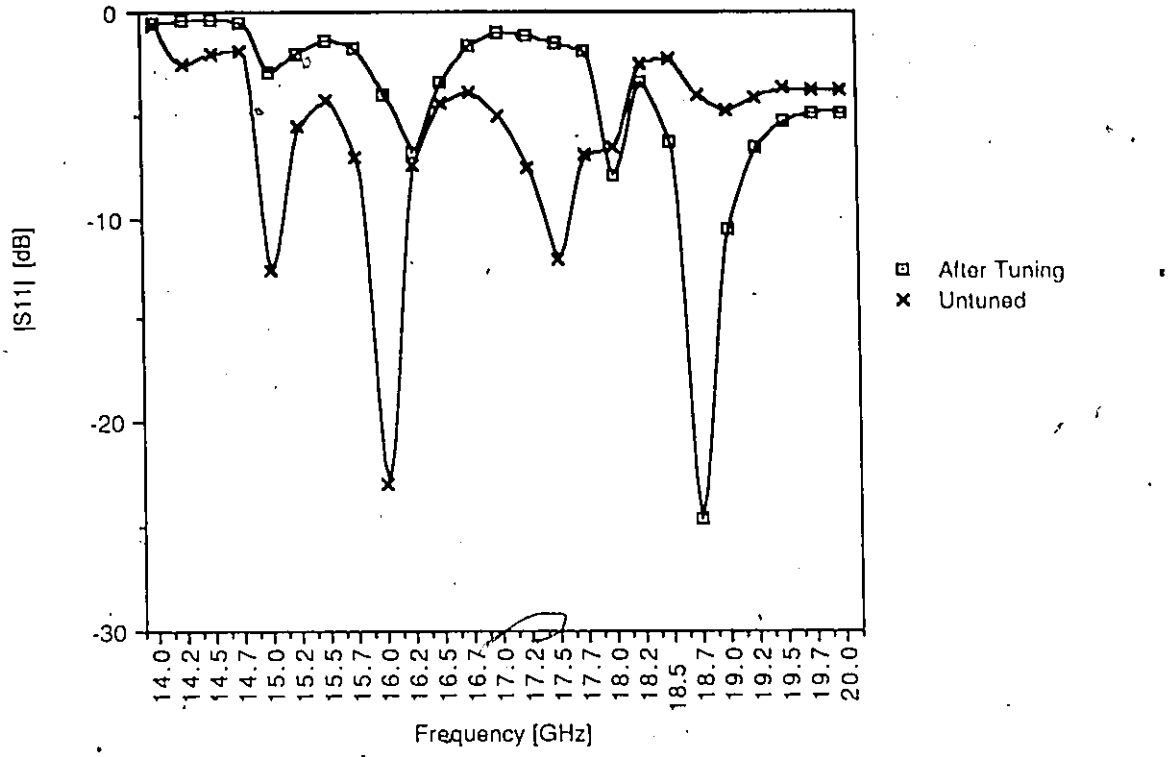


Figure 6.8: |S11| Before and After Tuning - Microstrip/FInline Doubler

6.2.3 Conversion Loss

The conversion loss as a function of input power at 18.75 GHz and of frequency was measured. The results prior to tuning were no better than 11 dB loss. However, it proved relatively easy to adjust circuit performance by removing a portion of the housing and using a small chip of Kovar. The junction of the bias circuit and the output line proved to be particularly sensitive, indicating insufficient rejection at $2f_0$. The results before and after tuning (Figure 6.9) show the response to be narrowband (5.80 dB +/- 0.15 across 350 MHz) and centred

about 150 MHz below the design frequency. Figure 6.10 shows the output power at $2f_o$ as a function of the input power at f_o. The doubler was driven with the maximum available power at 18.75 GHz (15 dBm) without going into saturation, indicating that even higher output power levels could be obtained with a suitable source.

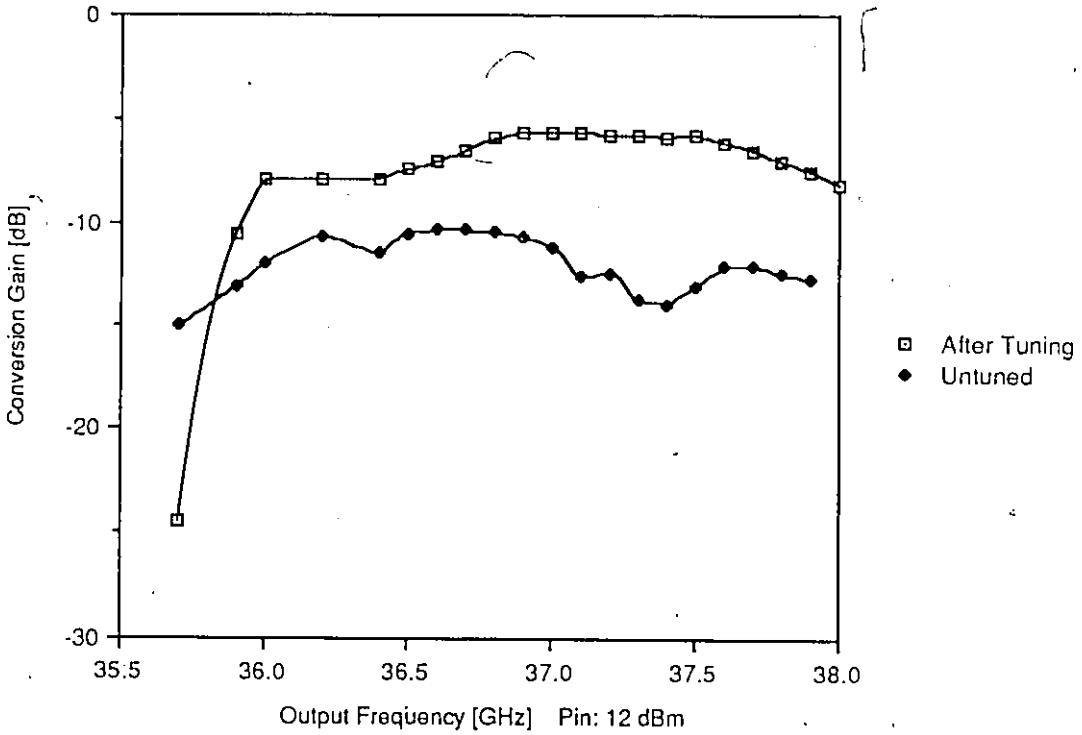


Figure 6.9: Microstrip/Finline Doubler Performance Before and After Tuning

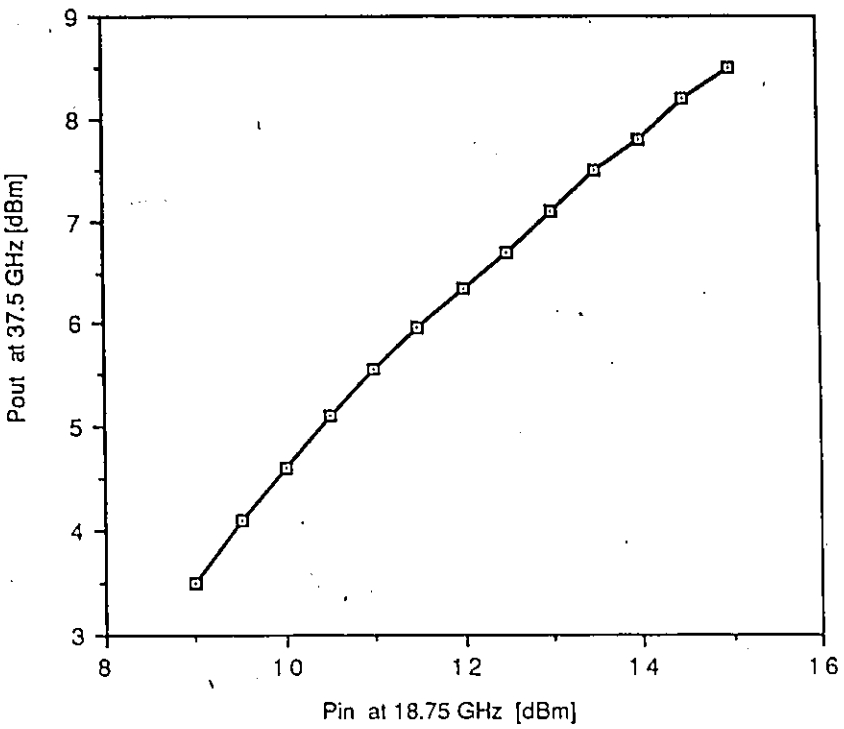


Figure 6.10: Pout(2fo) vs. Pin(fo)

6.3 Discussion and Conclusions

The microstrip/finline design demonstrated less loss (5.8 dB vs. 9 dB) , more bandwidth and higher output power (8.5 dBm vs. 4.5 dBm) than the all finline circuit. This is primarily due to the flexibility in tuning, permitting any changes required to optimize performance to be readily implemented.

The results obtained for the microstrip/finline doubler supported the FET model used, once the actual bonding conditions were taken into account. However the final finline circuit does not corroborate the small-signal model initially used in its

design. Any further finline work using FETs should be preceded by the development of calibration standards which can be used to characterize the device in this medium. The Through-Line-Reflect procedure mentioned in Chapter 4 would be a good method to apply in this case, since it does not require loads, and provides accurate de-embedding.

The use of FETs in finline is seen to be somewhat problematic, due to difficulties with stability, impedance levels and matching flexibility. The question of stability, caused by the arbitrary impedance presented at low frequencies by the waveguide beyond cutoff, can be solved by using a broadside-coupled housing, however the other problems are less tractable. In view of the fact that finline FET designs are largely empirical until the device can be modeled *in situ*, it is important that the final circuit can be fine-tuned in the lab to optimize performance. However adjusting the finline doubler, although possible, was limited by the topology of the circuit, i.e tuning stubs are easily shortened but are difficult to lengthen and impossible to relocate. The impedance levels required for FET matching are low enough to require impractically small feature sizes, which further limits the tunability of the structures. This is reflected in the low output power obtained from the all-finline doubler, indicating sub-optimum output match at $2f_0$.

The microstrip/finline doubler offered more flexibility both in terms of the matching and tuning. In addition, it retained the advantages of finline's E-plane configuration, providing waveguide compatibility and relatively relaxed dimensional tolerances.

7. CONCLUSIONS

7.0 Conclusions

The objectives of this thesis were twofold: to implement a large-signal model for the GaAsFET which would be readily obtainable using automated RF measurements, demonstrating its use in a nonlinear analysis simulation; to design and demonstrate a 18.75/37.5 GHz FET frequency doubler in E-plane technology.

The programs written for FET model parameter extraction provide an accurate and rapid means of characterizing devices over their entire operating range. It was shown that the model, based on S-parameters measured up to 10 GHz, could be extrapolated at least up to 18 GHz with acceptable results. While the programs are simple to execute, they do require some insight to determine the valid range of gate bias (for parasitic extraction) and frequency (for the intrinsic parameters) over which to take measurements. An improvement could be made by quantifying these restrictions more precisely to reduce uncertainty in the model.

The harmonic balance program written for use with the nonlinear FET model provides a basic analysis of the frequency multipler circuit with a relatively quick execution time, (due to its simplicity). This implementation could be upgraded by the addition of an optimization routine and more generalized input/output procedures. However, it served its primary purposes, which were to demonstrate how the FET model could be integrated into an analysis program and to be a learning tool for understanding FET multipliers. 'MHB' has been superseded at C.R.C. by the

advent of Libra™, so no further work will be done on this program.

Two frequency doublers were demonstrated, and illustrated both the advantages and disadvantages of E-plane technology. The all-finline design was hampered by lack of a reliable FET model and inability to correct for the error this created after fabrication, (i.e. no 'tunability'). This could be partly corrected by characterizing the FET directly in the finline environment using the Through-Line-Reflect calibration technique and the programs described in Chapter 2, however the basic inflexibility of this medium limits its usefulness in this application to those cases where planar integration is a must.

The microstrip/finline doubler demonstrated good performance, particularly in terms of output power. It exhibited the low-cost fabrication, waveguide compatibility and ruggedness associated with E-plane technology. A possible improvement in this component would be to increase the bandwidth by using a balanced topology, perhaps by exploiting the balanced line structure already present at the input.

```
PROGRAM  EXTPAR(input,output);
  {=============================================================================}
  { This program determines the extrinsic parameters of a GaAsFET (Rs,Rd,Rg,
  Ls,Ld,Lg) based on low-frequency S-parameter measurements made over a range of
  gate voltages,Vg, with Vds=0.  The approach used is based on a paper by
  Diamand, "Measurement of the Extrinsic Series Elements of a Microwave MESFET
  Under Zero Current Conditions", Proc. 12th EuMC, pp.451-456, 1982.
    Measurements are made on the HP8510, using the program "PARFET".  The pro-
  gram requires prior knowledge of device pinch-off voltage, Vp, and built-in
  Schottky barrier potential, phi.  The RF measurements should be made over
  at least 10-12 frequency points (PARFET outputs 25 points), and should
  not exceed 12GHz.  The gate bias should be maintained above pinchoff, i.e.
          1-((-Vg+phi)/(Vp+phi))^0.5 > 0.20
  The program inputs S-parameters at 51 frequency points for up to 40 values
  of bias.}
  {=============================================================================}

  CONST
       pi = 3.141592654;
   max_bias = 40;
      freq = 25;
     total = 1000;     { total = num_freq*max_bias}

  TYPE
   complex    = array [1..2]       of real;
   data_array  = array [1..total,1..4]  of complex;
   L_results   = array [1..freq,1..3]   of real;
   R_results   = array [1..2]       of real;
   vector     = array [1..freq]      of real;

  VAR
   Ls,Ld,Lg:            L_results;
   Rs,Rd,Rg:            R_results;
   phi,Vp,Vg:           real;
   conditions:           array[1..total,1..2] of real;
   Params:              data_array;
   num_bias,pointer,
   num_freq:             integer;
   one:                complex;
   filename:            string[12];
   out_data,in_data:       text;
   name,comment:          string[80];
   zero_drain,flag:       boolean;

  {$I CMPLX.PAS}
  {=============================================================================}

  PROCEDURE get_input;
  {=============================================================================}
  { This procedure inputs the name of the S-parameter data file and the values
  of phi and Vp, which must be known beforehand.  The gate voltage and frequency
  for each point are read into the array 'conditions'. The S-parameters are read
  into complex array 'Params'. (In order S11,S21,S12,S22.)}
  VAR
   header:             string[4];
   info,Vd,Vg_limit:      real;
   datum:              complex;
   index1,index2,index3:   integer;
   title:              string[80];

  BEGIN
    clrscr;
    zero_drain := true;
```

```
flag := false;
one[1] := 1;
one[2] := 0;
writeln('              EXTPAR - FET Extrinsic Parameter Calculation');
writeln('              ==========================================');
writeln;
writeln('Value of built-in barrier, phi      >      V');
writeln('Value of pinchoff voltage, Vp       >      V');
writeln('Enter input data filename           > ');
WRITELN('Number of frequencies measured      > ');
gotoxy(42,4);     read(phi);
gotoxy(42,5);     read(Vp);
gotoxy(42,6);     read(filename);
gotoxy(42,7);     readln(num_freq);
assign(in_data,filename);
{$I-} reset(in_data); {$I+}
if IOresult=0 then
begin
  flag := true;
  if Vp<0 then Vp:=-Vp;
  if phi<0 then phi := -phi;
  Vg_limit := 0.64*(Vp+phi)-phi;
  writeln('Limits for gate bias Vgs are -',Vg_limit:6:3,' to ',phi:6:3,' V.');
  readln(in_data,name); readln(in_data,comment);
  readln(in_data,num_bias);
  for index1 := 1 to num_bias do
  begin
    readln(in_data,header,Vg);
    if (abs(Vg)>Vg_limit) or (Vg>phi) then
      writeln('WARNING: Vg=',Vg:8:4,' exceeds valid Vg range.');
    readln(in_data,header,Vd);
    if Vd <> 0 then
    begin
      writeln('Input data invalid. Non-zero drain bias voltage (Vds=',Vd:5:2,')');
      zero_drain := false;
    end;
    readln(in_data,title);
    for index2 := 1 to num_freq do
    begin
      pointer := (index1-1)*num_freq+index2;
      conditions[pointer,1] := Vg;
      read(in_data,info);
      conditions[pointer,2] := info;
      for index3 := 1 to 4 do
      begin
        read(in_data,info);
        datum[1] := info;
        read(in_data,info);
        datum[2] := info;
        Params[pointer,index3] := datum;
      end;
      readln(in_data);
    end;
  end;
end
  else writeln('File not found.');
END;

PROCEDURE convert_to_Z;
{=============================================================================}
{ This procedure translates the S-parameters in 'Params', which are in [RI]
format, into Z-parameters and re-stores the results.  The complex arithmetic
routines used are from the include-file 'CMPLX'. The characteristic impedances
```

```
at input and output, Z01 and Z02, are assumed to be 50{ohms, but can be
changed.}
VAR
 Z01,Z02:       complex;        {Characteristic impedances}
 D:             complex;        {Conversion factor}
 a,b,c,p:       complex;        {Dummy variables for complex operations}
 S11,S21,S12,S22: complex;
 index:         integer;

BEGIN
  Z01[1] := 50; Z01[2] := 0;   { Change characteristic impedance here.}
  Z02[1] := 50; Z02[2] := 0;
  for index := 1 to pointer do
  begin
    S11 := Params[index,1];
    S21 := Params[index,2];
    S12 := Params[index,3];
    S22 := Params[index,4];
  { ****   Calculate expression for 'D', the common denominator.}
    csub(one,S11,b);
    csub(one,S22,c);
    cmult(b,c,a);
    cmult(S21,S12,p);
    csub(a,p,D);
  { ****   Calculate Z11 and store.}
    cadd(one,S11,b);
    csub(one,S22,c);
    cmult(b,c,a);
    cadd(a,p,c);
    cdiv(c,D,a);
    cmult(Z01,a,Params[index,1]);
  { ****   Calculate Z22.}
    csub(one,S11,b);
    cadd(one,S22,c);
    cmult(b,c,a);
    cadd(a,p,c);
    cdiv(c,D,a);
    cmult(Z02,a,Params[index,4]);
  { ****   Calculate Z21.}
    cdiv(S21,D,a);
    a[1] := 2*a[1];
    a[2] := 2*a[2];
    cmult(Z01,a,Params[index,2]);
  { ****   Calculate Z12.}
    cdiv(S12,D,a);
    a[1] := 2*a[1];
    a[2] := 2*a[2];
    cmult(Z02,a,Params[index,3]);
  end;
END;

PROCEDURE variance( S: real;  var sigma_S: vector);
{=================================================================================}
{ A procedure to determine the variance over frequency of the various access
elements calculated in 'extract_L' and 'extract_R'.  This is intended to give
a rough guideline on the reliability fo the results.}
VAR
  sum_S:     real;
  index:     integer;

BEGIN
  sum_S := 0;
  for index := 1 to num_freq do
```

```
  begin
    sum_S := sum_S+sqr(sigma_S[index]-S);
  end;
  sigma_S[1] := sum_S/num_freq;
END;

PROCEDURE intercept(DATA: L_results; NUM_PT: INTEGER; var M,intercept: real);
{=============================================================================}
{ This procedure finds the y-intercept of the input dataset DATA using linear
regression:
        y = mx + (<y>-m<x>)       (<> denotes mean value)

             S(x-<x>)(y-<y>)
          m = ---------------      (S denotes sum over frequencies)
              S (x-<x>)^2

}

VAR
  y,x,numer,denom: real;
  index:           integer;

BEGIN
  x := 0;
  y := 0;
  numer := 0;
  denom := 0;
  for index := 1 to num_pt do
  begin
    x := x+data[index,1];
    y := y+data[index,2];
  end;
  x := x/num_pt;
  y := y/num_pt;
  for index := 1 to num_pt do
  begin
    numer := numer+(data[index,1]-x)*(data[index,2]-y);
    denom := denom+sqr(data[index,1]-x);
  end;
  m := numer/denom;
  intercept := y-m*x;
END;

PROCEDURE extract_L;
{=============================================================================}
{ This procedure calculates the parasitic lead inductances at source, drain and
gate: Ls,Ld and Lg.  Following the analysis given by Diamand:

              Ls = 0.5*(X12+X21)/omega
              Ld = (X22/omega) - Ls

                 (X11*omega-X11'*omega')
              Lg = ----------------------- - Ls
                   (omega^2-omega'^2)

The results are stored into matrices 'Ls','Ld' and 'Lg'.
The standard deviation of Ls and Ld over frequency is also found and
stored in Lx[3], x=s,d.}
VAR
  X11,X22,X21,X12,omega:    REAL;
  datum:                    complex;
  sigma_Ls,sigma_Ld,sigma_Lg: vector;
```

```
  index1,index2,count:      integer;
  omega_p,X11_p:            real;

BEGIN
  for index1 := 1 to num_bias do
  begin
    Ls[index1,1] := 0;    Ls[index1,2] := 0;    Ls[index1,3] := 0;
    Ld[index1,1] := 0;    Ld[index1,2] := 0;    Ld[index1,3] := 0;
    Lg[index1,1] := 0;    Lg[index1,2] := 0;    Lg[index1,3] := 0;
    for index2 := 1 to num_freq do
    begin
      count := (index1-1)*num_freq+index2;
      omega := 2*pi*conditions[count,2];
      datum := Params[count,2];
      X21 := datum[2];
      datum := Params[count,3];
      X12 := datum[2];
      datum := Params[count,4];
      X22 := datum[2];
      DATUM := PARAMS[COUNT,1];
      X11 := datum[2];
      sigma_Ls[index2] := 0.5*(X12+X21)/omega;
      sigma_Ld[index2] := (X22/omega)-sigma_Ls[index2];
      if index2>1 then
      begin
        datum := Params[count-1,1];
        X11_p := datum[2];
        omega_p := 2*pi*conditions[count-1,2];
        sigma_Lg[index2] :=(X11_p*omega_p-X11*omega)/(sqr(omega_p)-sqr(omega))
                - sigma_Ls[index2];
      end;
       Ls[index1,1] := Ls[index1,1]+sigma_Ls[index2];
       Ld[index1,1] := Ld[index1,1]+sigma_Ld[index2];
       Lg[index1,1] := Lg[index1,1]+sigma_Lg[index2];
    end;
    Ls[index1,1] := Ls[index1,1]/num_freq;
    Ld[index1,1] := Ld[index1,1]/num_freq;
    Lg[index1,1] := Lg[index1,1]/(num_freq-1);
    variance(Ls[index1,1],sigma_Ls);
    variance(Ld[index1,1],sigma_Ld);
    variance(Lg[index1,1],sigma_Lg);
    Ls[index1,2] := conditions[count,1];
    Ld[index1,2] := conditions[count,1];
    Lg[index1,2] := conditions[count,1];
    Ls[index1,3] := sqrt(sigma_Ls[1]);
    Ld[index1,3] := sqrt(sigma_Ld[1])-Ls[index1,3];
    Lg[index1,3] := sqrt(sigma_Lg[1])-Ls[index1,3];
  end;
END;


PROCEDURE extract_R;
{=============================================================================}
{ This procedure calculates the parasitic resistances by finding the
y-intercept of the plots of the real parts of the Z-matrix against the inverse
normalized channel opening, 'b_a'. R11 vs. b_a gives Rs+Rd, R22 vs. b_a gives
Rg+Rs and 0.5*(R21+R12) vs. b_a gives Rs. This approach assumes a flat
doping profile. Results are stored in the vectors Rs,Rd and Rg, together
with standard deviation over frequency. A file of the R-matrix values vs. the
inverse of the normalized channel opening is output, as this can be useful for
graphing variation over frequency - the file is called 'R_MATRIX.OUT'.}

VAR
```

```
  R11,R21,R22:              L_results;
  sigma_Rs,sigma_Rd,sigma_Rg:   vector;
  b_a,SLOPE_RG,SLOPE_RD,SLOPE_RS: real;
  index1,index2,count:      integer;
  datum:                    complex;
  R_out:                    text;

BEGIN
  Rs[1] := 0; Rs[2] := 0;
  Rd[1] := 0; Rd[2] := 0;
  Rg[1] := 0; Rg[2] := 0;
  assign(R_out,'R_MATRIX.OUT');
  rewrite(R_out);
  writeln(R_out,'      R Matrix vs. Inverse of Normalized Channel Opening');
  writeln(R_out,'      ---------------------------------------------');
  for index1 := 1 to num_freq do
  begin
    writeln(R_out,conditions[index1,2]*1e-9:8:4,' GHz:');
    writeln(R_out,'      1/(B/A)       R11       R21       R22');
    writeln(R_out,'      ------       ---       ---       ---');
    for index2 := 1 to num_bias do
    begin
      count := (index2-1)*num_freq+index1;
      b_a := 1-sqrt((-1*conditions[count,1]+phi)/(Vp+phi));
      datum := Params[count,1];
      R11[index2,1] := 1/b_a;  R11[index2,2] := datum[1];
      datum := Params[count,2];
      R21[index2,1] := 1/b_a;  R21[index2,2] := datum[1];
      datum := Params[count,3];
      R21[index2,2] := 0.5*(datum[1]+R21[index2,2]);
      datum := Params[count,4];
      R22[index2,1] := 1/b_a;  R22[index2,2] := datum[1];
writeln(R_out,'      ',1/b_a:8:4,'      ',R11[index2,2]:8:4,'      ',R21[index2,2]:8:4,'      ',R22[index2,2]:8:4);
    end;
    intercept(R21,num_bias,slope_Rs,sigma_Rs[INDEX1]);
    intercept(R22,num_bias,slope_Rd,sigma_Rd[index1]);
    intercept(R11,num_bias,slope_Rg,sigma_Rg[index1]);
    sigma_Rd[index1] := sigma_Rd[index1]-sigma_Rs[index1];
    sigma_Rg[index1] := sigma_Rg[index1]-sigma_Rs[index1];
    Rs[1] := Rs[1]+sigma_Rs[index1];
    Rd[1] := Rd[1]+sigma_Rd[index1];
    Rg[1] := Rg[1]+sigma_Rg[index1];
  end;
  Rs[1] := Rs[1]/num_freq;
  Rd[1] := Rd[1]/num_freq;
  Rg[1] := Rg[1]/num_freq;
  variance(Rs[1],sigma_Rs);
  variance(Rd[1],sigma_Rd);
  variance(Rg[1],sigma_Rg);
  Rs[2] := sqrt(sigma_Rs[1]);
  Rd[2] := sqrt(sigma_Rd[1]);
  Rg[2] := sqrt(sigma_Rg[1]);
  close(R_out);
END;

PROCEDURE output;
{=============================================================================}
VAR
  index: integer;
  line:  string[80];

BEGIN
  line :='=============================================================================';
```

```
  writeln;  writeln(line);
  writeln('Calculations completed.');
  writeln('Enter output data filename        > ');
  write('(Older versions will be overwritten.)');
  gotoxy(42,12); readln(filename); gotoxy(1,13);
  assign(out_data,filename);
  rewrite(out_data);
  writeln(out_data,name);  writeln(out_data,comment);
  WRITELN(OUT_DATA,'PHI:  ',PHI:8:4,' V');
  WRITELN(OUT_DATA,'Vp :  ',VP:8:3,' V');
  write(out_data,'"SIGMA" = parameter variance over ',conditions[1,2]*1E-9:6:2);
  writeln(out_data,'-',conditions[num_freq,2]*1E-9:6:2,' GHz.');
  writeln(out_data,line); writeln(out_data);
  writeln(out_data,'Extrinsic Resistances (units are ohms):');  writeln(out_data,line);
  writeln(out_data,' PARAMETER     VALUE     SIGMA'); writeln(out_data,line);
  write(out_data,'    Rs    '); write(out_data,'   ',Rs[1]:5:2,'  ');
  writeln(out_data,'   ',Rs[2]:6:3,'  ');
  write(out_data,'    Rd    '); write(out_data,'   ',Rd[1]:5:2,'  ');
  writeln(out_data,'   ',Rd[2]:6:3,'  ');
  write(out_data,'    Rg    '); write(out_data,'   ',Rg[1]:5:2,'  ');
  writeln(out_data,'   ',Rg[2]:6:3,'  ');
  writeln(out_data,line);  writeln(out_data); writeln(out_data);
  writeln(out_data,'Extrinsic Inductances (units are nanohenries)');  writeln(out_data,line);
  writeln(out_data,'  Vg      Ls   SIGMA(Ls)   Ld   SIGMA(Ld)   Lg   ');
  writeln(out_data,line);
  for index := 1 to num_bias do
  begin
     write(out_data,'  ',Ls[index,2]:6:3,'    ',Ls[index,1]*1e9:6:3,'    ',Ls[index,3]*1e9:6:4);
     write(out_data,'    ',Ld[index,1]*1e9:6:3,'    ',Ld[index,3]*1e9:6:4);
     writeln(out_data,'    ',Lg[index,1]*1e9:6:3,'    ',Lg[index,3]*1e9:6:4);
  end;
  writeln(out_data,line);
  close(out_data);
END;
                { MAIN PROGRAM }
{-------------------------------------------------------------------------------}

BEGIN
  zero_drain := true;
  get_input;
  if (zero_drain = true) and (flag=true) then
  begin
    convert_to_Z;
    extract_L;
    extract_R;
    output;
  end;
END.
```

```
PROGRAM INTPAR(input,output);
{=============================================================================}
{ This program determines the intrinsic parameters of a GaAsFET (Cgd,Cgs,Cds,
Ri,gm,gd,tau) based on low-frequency S-parameter measurements made over a range
of bias points. The approach used is based on a paper by R.A. Minasian,
"Simplified GaAs MESFET Model to 10 GHz", Elect. Lett.,Vol.13,No.8,pp. 549-551
September 1, 1977.
   Measurements are made on the HP8510, using the program "PARFET". The pro-
gram requires prior knowledge of device access elements Rs,Rd,Rg,Ls,Ld,and Lg,
which are obtained using program "EXTPAR". The RF measurements should be made
over the 100 MHz - 4 GHz range, using a maximum of 20 bias points. If more
bias points are required, PARFET and EXTPAR must be run several times consec-
utively, (due to memory restrictions).}
{=============================================================================}

CONST
 pi      = 3.141592654;
 max_bias = 40;
 freq = 25;
 total   = 1000;

TYPE
 complex    = array [1..2]          of real;
 data_array  = array [1..total,1..4]    of complex;
 results    = array [1..max_bias,1..2]  of real;
 vector     = array [1..freq]      of real;

VAR
 Params:              data_array;
 Cgd,Cgs,Cds,gm,gd,Ri,tau:        results;
 Rs,Rd,Rg,Ls,Ld,Lg:             real;
 frequencies:                array [1..freq] of real;
 voltages:                  array [1..max_bias,1..2] of real;
 num_bias,pointer,num_freq:       integer;
 out_data,in_data:              text;
 filename:                 string[12];
 name,comment:                  string[80];
 ONE:                    COMPLEX;

{$I CMPLX.PAS}
{=============================================================================}

PROCEDURE get_input;
{=============================================================================}
{ This procedure inputs the values of the extrinsic parameters Rs,Rd,Rg,Ls,Ld,
and Lg, which must be known beforehand, as well as the name of the data file
containing the S-parameters. The bias voltages for each point are read into
the array 'voltages'and the frequency, into 'conditions'. The S-parameters are
read into complex array 'S_params'. (In order S11,S21,12,S22.)}

VAR
 header:             string[4];
 index1,index2,index3:    integer;
 info,Vg,Vd:           real;
 datum:              complex;
 title:            string[80];

BEGIN
  clrscr;
  ONE[1]:=1;
  ONE[2]:=0;
  writeln;
  writeln('Please enter the following extrinsic parameters (obtained by');
```

```
  writeln('running EXTPAR) In ohms and nanohenries:');
  writeln('          Source resistance,   Rs >');
  writeln('          Drain resistance,    Rd >');
  writeln('          Gate resistance,     Rg >');
  writeln('          Souce inductance,    Ls >');
  writeln('          Drain inductance,    Ld >');
  writeln('          Gate inductance,     Lg >');
  writeln('          Name of input data file >');
  writeln('      Number of frequencies measured >');
  gotoxy(42,4); read(Rs);
  gotoxy(42,5); read(Rd);
  gotoxy(42,6); read(Rg);
  gotoxy(42,7); read(Ls);
  gotoxy(42,8); read(Ld);
  gotoxy(42,9); read(Lg);
  gotoxy(42,10); read(filename);
  gotoxy(42,11); read(num_freq);
  assign(In_data,filename);
  reset(In_data);
  Ls := Ls*1e-9;
  Ld := Ld*1e-9;
  Lg := Lg*1e-9;
  readln(in_data,name); readln(in_data,comment);
  readln(in_data,num_bias);
  for index1 := 1 to num_bias do
  begin
    readln(in_data,header,Vg);
    readln(in_data,header,Vd);
    voltages[index1,1] := Vg;
    voltages[index1,2] := Vd;
    readln(in_data,title);
    for Index2 := 1 to num_freq do
    begin
      pointer := (index1-1)*num_freq+index2;
      read(in_data,info);
      frequencies[index2]:= info;
      for Index3 := 1 to 4 do
      begin
        read(in_data,datum[1],datum[2]);
        Params[pointer,index3] := datum;
      end;
      readln(in_data);
    end;
  end;
END;

PROCEDURE convert_to_Z;
{=============================================================================}
{ This procedure translates the S-parameters in 'Params', which are in [RI]
format, into Z-parameters and re-stores the results into array 'Params'.  The
complex arithmetic routines used are from the include-file 'CMPLX'.  The
characteristic impedances at input and output, Z01 and Z02, are assumed to be
50 ohms, but can be changed.}

VAR
 Z01,Z02:          complex;       {Characteristic impedances}
 D:                complex;       {Conversion factor}
 a,b,c,p:          complex;       {Dummy variables for complex operations}
 index:            integer;
 S11,S21,S12,S22:   COMPLEX;

BEGIN
  Z01[1] := 50; Z01[2] := 0;   { Change characteristic impedance here.}
```

```
Z02[1] := 50; Z02[2] := 0;
for index := 1 to pointer do
begin
  S11 := PARAMS[INDEX,1];
  S21 := PARAMS[INDEX,2];
  S12 := PARAMS[INDEX,3];
  S22 := PARAMS[INDEX,4];
{ ****   Calculate expression for 'D', the common denominator.}
  csub(one,S11,b);
  csub(one,S22,c);
  cmult(b,c,a);
  cmult(S21,S12,p);
  csub(a,p,D);
{ ****   Calculate Z11 and store.}
  cadd(one,S11,b);
  csub(one,S22,c);
  cmult(b,c,a);
  cadd(a,p,c);
  cdiv(c,D,a);
  cmult(Z01,a,Params[index,1]);
{ ****   Calculate Z22.}
  csub(one,S11,b);
  cadd(one,S22,c);
  cmult(b,c,a);
  cadd(a,p,c);
  cdiv(c,D,a);
  cmult(Z02,a,Params[index,4]);
{ ****   Calculate Z21.}
  cdiv(S21,D,a);
  a[1] := 2*a[1];
  a[2] := 2*a[2];
  cmult(Z01,a,Params[index,2]);
{ ****   Calculate Z12.}
  cdiv(S12,D,a);
  a[1] := 2*a[1];
  a[2] := 2*a[2];
  cmult(Z02,a,Params[index,3]);
end;
END;

PROCEDURE subtract_extrinsic;
{=============================================================================}
{ This procedure substracts the series extrinsic parameters from the
measured Z-parameters, leaving the intrinsic Z-parameters.}

VAR
  index1,index2,count: Integer;
  datum:          complex;
  omega:          real;

BEGIN
  for index1 := 1 to num_bias do
  begin
    for index2 := 1 to num_freq do
    begin
      count := (index1-1)*num_freq+index2;
      omega := frequencies[index2]*2*pi;
      {*****   Z11      *****}
      datum := Params[count,1];
      datum[1] := datum[1]-Rg-Rs;
      datum[2] := datum[2]-omega*(Lg+Ls);
      Params[count,1] := datum;
      {*****   Z22      *****}
      datum := Params[count,4];
```

```
      datum[1] := datum[1]-Rd-Rs;
      datum[2] := datum[2]-omega*(Ld+Ls);
      Params[count,4] := datum;
      {*****   Z21      *****}
      datum := Params[count,2];
      datum[1] := datum[1]-Rs;
      datum[2] := datum[2]-omega*Ls;
      Params[count,2] := datum;
      {*****   Z12      *****}
      datum := Params[count,3];
      datum[1] := datum[1]-Rs;
      datum[2] := datum[2]-omega*Ls;
      Params[count,3] := datum;
    end;
  end;
END;

PROCEDURE convert_to_Y;
{=============================================================================}
{ This procedure translates the Z-parameters in 'Params', which are in [RI]
format, into Y-parameters and restores the results into 'Params'.  The complex
arithmetic routines used are from the include-file 'CMPLX'.  }

VAR
 D:       complex;       {Conversion factor}
 a,b,c:   complex;       {Dummy variables for complex operations}
 minus_one: complex;     { -1 + j0 }
 index:   integer;

BEGIN
  minus_one[1] := -1;   minus_one[2] := 0;
  for index := 1 to pointer do
  begin
  { ****   Calculate expression for 'D', the determinant.}
    cmult(Params[index,1],Params[index,4],a);
    cmult(Params[index,2],Params[index,3],b);
    csub(a,b,D);
  { ****   Calculate Y11 and Y22 and store.}
    cdiv(Params[index,4],D,a);
    cdiv(Params[index,1],D,b);
    Params[index,1] := a;
    Params[index,4] := b;
  { ****   Calculate Y21 and Y12 and store.}
    cdiv(Params[index,3],D,a);
    cmult(minus_one,a,b);
    cdiv(Params[index,2],D,a);
    cmult(minus_one,a,c);
    Params[index,3] := b;
    Params[index,2] := c;
  end;
END;

PROCEDURE variance(index1: integer; var sig_gm,sig_gd,sig_Cgd,sig_Cgs,
                   sig_Cds,sig_Ri,sig_tau: vector);
{=============================================================================}
{ A procedure to determine the variance over frequency of the various
intrinsic elements calculated in 'extract_intrinsic'.}

VAR
  index: integer;
  sum_gm,sum_gd,sum_Cgd,sum_Cgs,
  sum_Cds,sum_Ri,sum_tau: real;
```

```
BEGIN
  sum_gm := 0; sum_gd := 0; sum_Cgd := 0; sum_Cgs := 0;
  sum_Cds:= 0; sum_Ri := 0; sum_tau := 0;
  for index := 1 to num_freq do
  begin
    sum_gm  := sum_gm  + sqr(sig_gm[index]  - gm[index1,1]);
    sum_gd  := sum_gd  + sqr(sig_gd[index]  - gd[index1,1]);
    sum_Cgd := sum_Cgd + sqr(sig_Cgd[index] - Cgd[index1,1]);
    sum_Cgs := sum_Cgs + sqr(sig_Cgs[index] - Cgs[index1,1]);
    sum_Cds := sum_Cds + sqr(sig_Cds[index] - Cds[index1,1]);
  end;
    sig_gm[1]  := sum_gm/num_freq;
    sig_gd[1]  := sum_gd/num_freq;
    sig_Cgd[1] := sum_Cgd/num_freq;
    sig_Cgs[1] := sum_Cgs/num_freq;
    sig_Cds[1] := sum_Cds/num_freq;
END;

PROCEDURE extract_intrinsic;
{=============================================================================}
{ This procedure calculates the intrinsic model parameters according to the
analysis given by Minasian:
                gm = Re[Y21]
                gd = Re[Y22]
               Cgd = -Im[Y12]/omega
               Cgs = (Im[Y11]/omega)  - Cgd
               Cds = (Im[Y22]/omega)  - Cgd
                Ri = Re[Y11]/(Cgs*omega)^2
                   -(Im[Y21]/omega) - Cgd
               tau = ------------------  - Ri*Cgs
                          gm
The values for each parameter are averaged over frequency and the results
stored in the appropriate array, together with the corresponding bias voltage
and standard deviation of the parameter over frequency.}

VAR
sigma_gm,sigma_gd,sigma_Cgd,
sigma_Cgs,sigma_Cds,sigma_Ri,
sigma_tau:              vector;
info,omega:             real;
datum:                  complex;
index1,index2,count:        integer;
RI_INDEX:               INTEGER;
BEGIN
  for index1 := 1 to num_bias do
  begin
    gm[index1,1] := 0;    gm[index1,2] := 0;
    gd[index1,1] := 0;    gd[index1,2] := 0;
   Cgd[index1,1] := 0;   Cgd[index1,2] := 0;
   Cgs[index1,1] := 0;   Cgs[index1,2] := 0;
   Cds[index1,1] := 0;   Cds[index1,2] := 0;
    Ri[index1,1] := 0;    Ri[index1,2] := 0;
   tau[index1,1] := 0;   tau[index1,2] := 0;
   RI_INDEX :=0;
    for index2 := 1 to num_freq do
    begin
      count := (index1-1)*num_freq+index2;
      omega := frequencies[index2]*2*pi;
      datum := Params[count,3];
      sigma_Cgd[index2] := -datum[2]/omega;
      datum := Params[count,1];
      sigma_Cgs[index2] := datum[2]/omega-sigma_Cgd[index2];
      datum := Params[count,4];
```

```
      sigma_gd[index2] := datum[1];
      sigma_Cds[index2] := (datum[2]/omega)-sigma_Cgd[index2];
      datum := Params[count,2];
      sigma_gm[index2] := datum[1];

      gm[index1,1] := gm[index1,1]+sigma_gm[index2];
      gd[index1,1] := gd[index1,1]+sigma_gd[index2];
     Cgd[index1,1] := Cgd[index1,1]+sigma_Cgd[index2];
     Cgs[index1,1] := Cgs[index1,1]+sigma_Cgs[index2];
     Cds[index1,1] := Cds[index1,1]+sigma_Cds[index2];
    WRITELN;
    IF FREQUENCIES[INDEX2]>=2.5 THEN
    BEGIN
      WRITELN('CALCULATING RI. ',FREQUENCIES[INDEX2]*1E-9:8:4);
      rI_INDEX := RI_INDEX+1;
     datum := Params[count,1];
      sigma_Ri[index2] := datum[1]/sqr(sigma_Cgs[index2]*omega);
     datum := Params[count,2];
      sigma_tau[index2] := ((-datum[2]/omega)-sigma_Cgd[index2])/
            sigma_gm[index2] - sigma_Ri[index2]*sigma_Cgs[index2];

    END;
     Ri[index1,1] := Ri[index1,1]+sigma_Ri[index2];
     tau[index1,1] := tau[index1,1]+sigma_tau[index2];
     {** NB: expression for gm valid as omega tends to 0, while Ri is
      particularly sensitive to variations in S11 and may not valid for
      low frequencies.}
   end;
    gm[index1,1] := gm[index1,1]/num_freq;
    gd[index1,1] := gd[index1,1]/num_freq;
   Cgd[index1,1] := Cgd[index1,1]/num_freq;
   Cgs[index1,1] := Cgs[index1,1]/num_freq;
   Cds[index1,1] := Cds[index1,1]/num_freq;
    Ri[index1,1] := Ri[index1,1]/RI_INDEX;
   tau[index1,1] := tau[index1,1]/RI_INDEX;
    variance(index1,sigma_gm,sigma_gd,sigma_Cgd,sigma_Cgs,
        sigma_Cds,sigma_Ri,sigma_tau);
    gm[index1,2] := sqrt(sigma_gm[1]);
    gd[index1,2] := sqrt(sigma_gd[1]);
   Cgd[index1,2] := sqrt(sigma_Cgd[1]);
   Cgs[index1,2] := sqrt(sigma_Cgs[1]);
   Cds[index1,2] := sqrt(sigma_Cds[1]);
    Ri[index1,2] := 0;
   tau[index1,2] := 0;
  end;
END;

PROCEDURE output;
{=============================================================================}
VAR
  index1,num_Vd,num_Vg,
  pointer,index2,count: integer;
  title1,title2,line:  string[80];
  parameter: char;
  Vd,Vg:    real;

BEGIN
  line :='=============================================================================';
  title1:='    gm    SIG(gm)    gd    SIG(gd)    Ri    SIG(Ri)';
  title2:='    Cgd  SIG(Cgd)  Cgs  SIG(Cgs)    Cds  SIG(Cds)    Tau  SIG(Tau)';
  writeln;  writeln(line); writeln;
  writeln('Calculations completed.');
  writeln('Enter output data filename              > ');
```

```
writeln('(Older versions will be overwritten.)');
writeln('Which bias voltage to be taken as parameter,');
writeln('Vd or Vg  [D/G]                    > ');
gotoxy(47,15); read(filename);
gotoxy(47,18); read(parameter);
gotoxy(1,20);
assign(out_data,filename);
rewrite(out_data);
writeln(out_data,name);
writeln(out_data,comment);
write(out_data,'"SIG" = parameter standard deviation over ');
write(out_data,'(Units:Siemens, pF, ohms, ps.)');
writeln(out_data,frequencies[1]*1E-9:8:2,' to ',frequencies[num_freq]*1E-9:8:2,' GHz.');
num_Vd := 1;
repeat
  num_Vd := num_Vd+1;
until (voltages[num_Vd,2]=voltages[1,2]) or (num_Vd=num_bias);
if num_Vd <> num_bias then num_Vd := num_Vd - 1;
num_Vg := trunc(num_bias/num_Vd);
if (parameter='D') or (parameter='d') then
begin
  for index1 := 1 to num_Vd do
  begin
    Vd := voltages[index1,2];
    writeln(out_data,'Vd : ',Vd:6:2);
    writeln(out_data,line);
    writeln(out_data,'  Vg',title1);
    for index2 := 1 to num_Vg do
    begin
      count := (index2-1)*num_Vd+index1;
      Vg := voltages[count,1];
       write(out_data,Vg:6:2,' ',gm[count,1]:8:5,'  ',gm[count,2]:8:5);
       write(out_data,'    ',gd[count,1]:8:5,'  ',gd[count,2]:8:5,'    ');
       writeln(out_data,Ri[count,1]:8:4,'  ',Ri[count,2]:8:4);
    end;
    writeln(out_data,line);
    writeln(out_data,'  Vg',title2);
    for index2 := 1 to num_Vg do
    begin
      count := (index2-1)*num_Vd+index1;
      Vg := voltages[count,1];
       write(out_data,Vg:5:2,'  ',Cgd[count,1]*1e12:6:3,'  ');
       write(out_data,Cgd[count,2]*1e12:6:3,'    ',Cgs[count,1]*1e12:6:3);
       write(out_data,'  ',Cgs[count,2]*1e12:6:3,'    ');
       write(out_data,Cds[count,1]*1e12:6:3,'  ',Cds[count,2]*1e12:6:3);
       write(out_data,'    ',tau[count,1]*1e12:6:3,'  ');
       writeln(out_data,tau[count,2]*1e12:6:3);
    end;
    writeln(out_data,line); writeln(out_data); writeln(out_data);
  end;
end
else begin
  for index1 := 1 to num_Vg do
  begin
    pointer := (index1-1)*num_Vd+1;
    Vg := voltages[pointer,1];
    writeln(out_data,'Vg : ',Vg:6:2);
    writeln(out_data,line);
    writeln(out_data,'  Vd',title1);
    for index2 := 1 to num_Vd do
    begin
      count := (index1-1)*num_Vd+index2;
      Vd := voltages[count,2];
```

```
          write(out_data,Vd:6:2,' ',gm[count,1]:8:5,'  ',gm[count,2]:8:5);
          write(out_data,'    ',gd[count,1]:8:5,'  ',gd[count,2]:8:5,'    ');
          writeln(out_data,Ri[count,1]:8:4,'  ',Ri[count,2]:8:4);
        end;
        writeln(out_data,line);
        writeln(out_data,' Vd',title2);
        for index2 := 1 to num_Vd do
        begin
          count := (index1-1)*num_Vd+index2;
          Vd := voltages[count,2];
          write(out_data,Vd:5:2,'  ',Cgd[count,1]*1e12:6:3,'  ');
          write(out_data,Cgd[count,2]*1e12:6:3,'    ',Cgs[count,1]*1e12:6:3);
          write(out_data,'  ',Cgs[count,2]*1e12:6:3,'   ');
          write(out_data,Cds[count,1]*1e12:6:3,'  ',Cds[count,2]*1e12:6:3);
          write(out_data,'    ',tau[count,1]*1e12:6:3,'  ');
          writeln(out_data,tau[count,2]*1e12:6:3);
        end;
      writeln(out_data,line); writeln(out_data); writeln(out_data);
      end;
    end;
    close(out_data);
END;
                    { MAIN PROGRAM }
{=============================================================================}

BEGIN
  get_input;
  convert_to_Z;
  subtract_extrinsic;
  convert_to_Y;
  extract_intrinsic;
  output;
END.
{}
```

APPENDIX II: HARMONIC BALANCE PROGRAM

```
PROGRAM ModifiedHarmonicBalance;
{-------------------------------------------------------------------------}
{FET MULTIPLIER ANALYSIS PROGRAM: (Refer to Chapter 3). This program
 calculates conversion loss of a FET multiplier as a function of bias and
 embedding impedances presented at all frequencies of interest.

 Data must be entered via a separate file using the below format. Procedures for calculating nonlinear
  FET parameters are contained in a separate source code, and must be developed prior to program execution,
 using INTPAR and EXTPAR.  (refer to Chapter 2 and Appendix II).

 INPUT FILE ORGANIZATION:

 Vsource,gbias,dbias                                                    SIGNAL INPUT PARAMETERS
 Fo [GHz]                                                               FUNDAMENTAL FREQUENCY
 Zcin,Zcout                                                             CHARACTERISTIC IMPEDANCES
 Tolerance                                                              CONVERGENCE TOLERANCE
 Re(Zi[0]) Im(Zi[0]) Re(Zi[1]) Im(Zi[1]) Re(Zi[2]) Im(Zi[2]) Re(Zi[3]) Im(Zi[3])   INPUT EMBEDDING IMPEDANCES

 Re(Zo[0]) Im(Zo[0]) Re(Zo[1]) Im(Zo[1]) Re(Zo[2]) Im(Zo[2]) Re(Zo[3]) Im(Zo[3])   OUTPUT EMBEDDING IMPEDANCE

 phi,Ri,Rs,Ls                                                           FET MODEL PARAMETERS}

CONST
  pi = 3.1415926535898 ;

TYPE
  complex    = array [1..2]      of real ;
  wave       = array [0..127]    of real ;
  freqvector = array [0..64]     of real ;
  refl_coeff = array [0..3]      of complex;
  impdvector = array [0..3]      of complex;
  sense      = string[3];

VAR
  pavail,pdc,pwrdiss      : real;    { Variables used to describe power input, output and efficiency.  }
  Input_dBm,Input_power   : real;
  conv_loss,rev_loss      : real;

  Vsource,gbias,dbias     : real;    { Input excitation.    .}


  Fo                      : real;    { Fundamental frequency.  }

  Zembout,Zembin          : impdvector ; { Embedding impedances seen at harmonics and DC.   }

  one,zero,j              : complex ;  { 1+j0, 0+j0, 0+j1      }

  IN_error,OUT_ERROR,tolerance : real ;    { Limit for convergence.  }

  infile,outfile,spectrum : text ;    { Input and output files . }

  filename                : string [20] ; { User-input filenames.  }

  Zcin,Zcout              : real ;    { Characteristic impedance of transmission lines.  }

  phi,Ri,Rs,Ls            : real ;    { FET MODEL  parameters.  }


  iteration               : integer;  { Loop counter in harmonic iteration loop.      }
  Vds,Ids,Vgs,Igs         : wave;
  Vf,Ifi,Igf,Idf,Vg,Vd    : wave;

{$I INT_FET.PAS}          {Large-signal FET characterization.}
```

```
{$I FFT.PAS}                    {FFT & IFT procedure}
{$I SCIFUNC.PAS}                {Contains scientific function subroutines}
{$I CMPLXOPS.PAS}               {Contains complex arithmetic subroutines}

{-------------------------------------------------------------------------}

procedure UPDATE_ERRORS(old_input,new_input,old_output,new_output: wave);
{-------------------------------------------------------------------------}
var
  old_in,delta_in,
  old_out,delta_out,
  in_average,out_average:   real;
  index :         integer;

begin
  old_in := IN_error;
  IN_error := 0;
  old_out := OUT_error;
  OUT_error := 0;
  in_average := 0;
  out_average := 0;
  for index := 0 to 127 do
  begin
    IN_error := IN_error+abs(old_input[index]-new_input[index]);
    in_average := in_average+new_input[index];
   OUT_ERROR := OUT_ERROR+abs(old_output[index]-new_output[index]);
    out_average := out_average+new_output[index];
  end;
  IN_error := IN_error/128.0;
  in_average := in_average/128;
  out_average := out_average/128;
  DELTA_IN:= old_in-IN_error;
  OUT_error := OUT_error/128.0;
  delta_out:= old_out-OUT_error;
  writeln('RMS err AT INPUT          = ',IN_error:12:6);
  writeln('Previous RMS error AT INPUT = ',old_in:12:6);
  writeln('Change                    = ',DELTA_IN:8:4);
  writeln('Average Vgs               = ',in_average:10:5);
  writeln;
  writeln('RMS err AT OUTPUT         = ',OUT_error:12:6);
  writeln('Previous RMS error AT OUTPUT = ',old_out:12:6);
  writeln('Change                    = ',delta_out:8:4);
  writeln('Average Vds               = ',out_average:10:5);
end;

procedure MULTIPLY_BY_RHO(rho: refl_coeff;
            var coscoeff,sincoeff: freqvector);
{-------------------------------------------------------------------------}
var
  index             :       integer;
  Vref_freq,new_Vref_freq :       complex;
begin
  for index := 0 to 3 do
  begin
    Vref_freq[1] := coscoeff[index];
    Vref_freq[2] := sincoeff[index];
    cmult(rho[index],Vref_freq,new_Vref_freq);
    coscoeff[index] := new_Vref_freq[1];
    sincoeff[index] := new_Vref_freq[2];
  end;
end;
```

```
procedure INTEGRATE_FET_IV_EQUATIONS( Vgs_inc,Vds_inc: wave; var Vgs_ref,Vds_ref: wave);
{-------------------------------------------------------------------------------}
{ Based on the updated incident waves, the intrinsic FET response is
  calculated.  The resulting voltages represent the waves that will be
  reflected back along the transmission lines.                          }
var
  j,k,jm1,jm2,index          :   integer;

  r1,r2,dIgdT,diddt,dt,error     :   real;
  internal:                  text;
   imvd,revd,imid,reid,imig,reig,imvg,revg: freqvector;

begin
  dt := 1/(128*fo);
  if iteration = 1 then
  begin
    Igl[127] := 0;
    Idl[127] := 0;
    Ifi[127]  := 0;
    dIgdt :=0;
    index := 0;
    error := 0;
    repeat
      index := index+1;
      Idl[127] :=0.001*(INDEX-1);
       Vg[127] := 2*Vgs_inc[127]-Idl[127]*Rs;
       Vd[127] := 2*Vds_inc[127]-(zCOUT+rs)*Idl[127];
      R1 := GM(gbias,dbias);
      R2 := GD(gbias,dbias);
       error := Idl[127]-R1*VG[127]-R2*VD[127];
    until (abs(error)<=0.005) ;
    Vf[127] := Vg[127]-Vd[127];
    writeln('Initialized results : Error  ',error:10:6);
    writeln('                      Idl ',Idl[127]*1000:10:4,' mA');
    writeln('                      Vg ',vg[127]:10:4);
    writeln('                      Vd ',vd[127]:10:4);
    writeln('                      Vf ',vf[127]:10:4);
    writeln('                    Index  ',index:4);
  end;
  for k := 1 to 7 do
  begin
    writeln(k:4);
    for j := 0 to 127 do
    begin
      jm1 := j-1;
      if j = 0 then jm1 := 127;
       Vf[j] := Vf[jm1]+Ifi[jm1]*dt/cgd(vg[jm1],vd[jm1]);
       Vg[j] := Vg[jm1]+Igl[jm1]*dt/cgs(vg[jm1],vd[jm1]);
      Igl[j] := Igl[jm1]+dIgdT*dt;
      Vd[j]  := Vg[j]-Vf[j]+Ri*Igl[j];
       Idl[j] := GM(vg[j],vd[j])*VG[J]+GD(vg[j],vd[j])*VD[J];
      Ifi[j] := (2*Vgs_inc[j]-2*Vds_inc[j]-Vf[j]-ZCIN*IGL[j]+ZCOUT*Idl[j])/(ZCIN+ZCOUT);
       dIgdT  := ((2*vds_inc[j]-vd[j]-zcout*(idl[j]-ifi[j])-rs*(igl[j]+idl[j]))/Ls)-(Idl[j]-Idl[jm1])/dt;
    end;
  end;
  for j := 0 to 127 do
  begin
     Ids[j] := Idl[j]-Ifi[j];
     Igs[j] := Igl[j]+Ifi[j];
  end;
 FFT('FFT',Vg,revg,imvg) ;        { FFT to get frequency domain reflected wave.}
 FFT('FFT',Vd,revd,imvd);
```

```
FFT('FFT',Ids,reid,imid);
FFT('FFT',igs,reig,imig);
for index:= 1 to 16 do
begin
   revd[index] := revd[index] +rs*(reid[index]+reig[index])-reid[index]*zcout;
   imvd[index] := imvd[index]+Ls*index*2*3.141592654*fo*(imid[index]+imig[index]);
   revg[index] := revg[index]-reig[index]*zcin*rs*(reig[index]+reid[index]);
   imvg[index] := imvg[index]+Ls*index*2*3.141592654*fo*(imid[index]+imig[index])
          +imig[index]*index*2*3.141592654*fo*0.5e-12;
end;

FFT('IFT',Vds_ref,revd,imvd);
FFT('IFT',Vgs_ref,revg,imvg);
 for j := 0 to 127 do
 begin
 writeln(j:4,'  ',Vds_ref[j]:9:5,'  ',Vgs_ref[j]:9:5);
 writeln(outfile,j:4,'  ',Vds_ref[j]:9:5,'  ',Vgs_ref[j]:9:5);
 end;
end;


procedure find_initial_voltage_wave(z: impdvector; source,zchar,vbias: real;
                    var initial,prev,vinc: wave);

{-------------------------------------------------------------------------------}
{ Refer to equations 3.6, 3.7 in text.                          }
var
  j         : integer;
  Z_DC,Z_fo    : complex;
  phase       : real;
begin
  Z_DC  := z[0] ;
  Z_fo  := z[1] ;
  phase := atan2(Z_fo[2],Z_fo[1] + zchar) ;
  for j := 1 to 128 do
  begin
    initial[j-1] := vbias * zchar / (zchar + Z_DC[1])
            + source* zchar * COS(2.0*pi*j/128 - phase) /
              sqrt(sqr(zchar + Z_fo[1])+sqr(Z_fo[2])) ;
    vinc[j-1] := initial[j-1] ;
    prev[j-1] := 0.0 ;
  end ;
end;

procedure CALCULATE_PERFORMANCE;
{-------------------------------------------------------------------------------}
{ The initial travelling wave is found using equations 3.6 and 3.7 from text.
  The 'V_inc_init' waves retain this value throughout the program. The
  'V_inc_prev'wave is initialized to this value and altered during iteration.}
var
   zsource,vref_freq,new_vinc_freq,

  Vgs_inc_init,Vgs_inc_prev,              { Time representation of initial          }
  Vds_inc_init,Vds_inc_prev,              { incident and reflected voltage waves    }
  Vds_ref,Vgs_ref          : wave ;   { sampled 128 times over 1 period of Fo.    }

  coscoeff,sincoeff          : freqvector ; { 16-element arrays of Fourier coefficients.}

  rho_in,rho_out             : refl_coeff;  { Complex arrays of reflection coefficients for harmonics.}

  index,index1,index2        : integer;    { Loop pointers.          }

  in_converged,out_converged   : boolean;    { Comvergence flags.      }
```

```
  temp1,temp2,A,B,C,D    : complex;    { Dummy variables for rho. }

  conv_loss,rev_loss     : real;       { output and efficiency.  }

  reply                  : char;

begin
  reply := 'y';
  iteration := 0 ;                    { Initialize loop pointer }
  in_converged := false ;                  { and convergence flags.  }
  out_converged := false;

  temp1[1]:=Zcin;                      { Initialize dummy variables }
  temp1[2]:=0;                        { for calculating rho.     }
  temp2[1]:=Zcout;
  temp2[2]:=0;


  for index := 0 to 3 do
  begin
    csub(zembin[index],temp1,A) ;      { Find the input and output   }
    cadd(zembin[index],temp1,B) ;      { relection coeffiecients     }
    cdiv(A,B,C);
    RHO_IN[INDEX] := C;              { at each harmonic and DC.    }
    csub(zembout[index],temp2,A) ;
    cadd(zembout[index],temp2,B) ;
    cdiv(A,B,D) ;
    RHO_OUT[INDEX] := D;
  end;

   find_initial_voltage_wave(zembin,Vsource,Zcin,gbias,Vgs_inc_init,Vgs_inc_prev,Vgs);
   find_initial_voltage_wave(zembout,0,Zcout,dbias,Vds_inc_init,Vds_inc_prev,Vds);

  repeat                              { Begin iterative procedure. }
    iteration := iteration + 1 ;
    writeln('Harmonic iteration no: ',iteration:4) ;
    for index := 0 to 127 do                { Save old incident wave    }
    begin                             { to use later, in determining}
      Vgs_inc_prev[index] := Vgs[index] ;  { convergence }
      Vds_inc_prev[index] := Vds[index] ;
    end;

    integrate_fet_iv_equations(Vgs,Vds,        { Calculate reflected waves}
                    Vgs_ref,Vds_ref) ; { at input and output.    }
    FFT('FFT',Vgs_ref,coscoeff,sincoeff);
writeln(spectrum,'Vgs spectrum for iteration: ',iteration:5);
FOR INDEX := 1 TO 16 DO
BEGIN
  writeln(INDEX:4,' ',COSCOEFF[INDEX]:10:5,' ',SINCOEFF[INDEX]:10:5);
  writeln(spectrum,INDEX:4,' ',COSCOEFF[INDEX]:10:5,' ',SINCOEFF[INDEX]:10:5);
END;
    multiply_by_rho(rho_in,coscoeff,sincoeff);  { Multiply by reflection coefficients, rho.  }
                              { to get new freq-domain incident wave }
    FFT('IFT',Vgs,coscoeff,sincoeff);        { Revert to time domain.  }

    FFT('FFT',Vds_ref,coscoeff,sincoeff);
writeln(spectrum);
writeln(spectrum,'Vds spectrum for iteration: ',iteration:5);
FOR INDEX := 1 TO 16 DO
BEGIN
  writeln(INDEX:4,' ',COSCOEFF[INDEX]:10:5,' ',SINCOEFF[INDEX]:10:5);
  writeln(spectrum,INDEX:4,' ',COSCOEFF[INDEX]:10:5,' ',SINCOEFF[INDEX]:10:5);
```

```
END;
writeln(spectrum); writeln(spectrum);
     multiply_by_rho(rho_out,coscoeff,sincoeff);
     FFT('IFT',Vds,coscoeff,sincoeff);
    for index := 0 to 127 do                { Add new waveform to original }
    begin
       Vgs[index] := Vgs[index] + Vgs_inc_init[index] ;
       Vds[index] := Vds[index] + Vds_inc_init[index] ;
    end ;
for index := 0 to 127 do
begin
   writeln(index:4,'ref :',vds_ref[index]:10:5,'  inc:',vds[index]:10:5);
end;
     update_errors(Vgs_inc_prev,Vgs,Vds_inc_prev,Vds);
    write('Continue iteration ? ');readln(reply);
  until (reply='n') or (reply='N');
  close(outfile);
  close(spectrum);
end ;

procedure GET_INPUT;
{=============================================================================}
var
  header     : string[80];                  { Dummy variables for }
  temp1,temp2 : complex;                        { reading input data. }
  index      : integer;                   { Loop counter.      }

begin
  writeln ;                              { Get input filename and }
  write('Enter data file name and extension : ') ; { assign to 'infile'.   }
  readln(filename) ;
  Assign(infile,filename) ;
  Reset(infile) ;
                                  { Begin file read.     }
   readln(infile,header);readln(infile,Vsource,gbias,dbias);
   readln(infile,header);readln(infile,Fo);
   readln(infile,header);readln(infile,Zcin,Zcout);
   readln(infile,header);readln(infile,tolerance);
  readln(infile,header) ;
  for index := 0 TO 3 do
  begin
    read(infile,temp1[1],temp1[2]) ;
    zembin[index] := temp1 ;
  end ;    readln(infile,header);
  readln(infile,header) ;
  for index := 0 TO 3 do
  begin
    read(infile,temp1[1],temp1[2]) ;
    zembout[index] := temp1 ;
  end ;     readln(infile,header);
  readln(infile,header);readln(infile,PHI,Ri,Rs,Ls);

  if phi<0 then phi := -phi;
  Fo := Fo*1e9;
  Ls := Ls*1e-9;

  write('Name output to go to [include extension]  : '); { Get output filename }
  readln(filename);                              { and assign to 'outfile'. }
   assign(outfile,filename);
  rewrite(outfile);
   assign(spectrum,'spectrum.out');
  rewrite(spectrum);
end;
```

```
procedure INITIALIZE;
{=============================================================================}
begin
  zero[1] := 0.0 ;          { Set complex constants }
  zero[2] := 0.0 ;          { zero = 0 + j0    }
  one[1]  := 1.0 ;
  one[2]  := 0.0 ;          { one  = 1 + j0    }
  j[1]    := 0.0 ;
  j[2]    := 1.0 ;          { j    = 0 + j1    }
end;

{                  MAIN PROGRAM                    }
{=============================================================================}

begin
  initialize;          {Initialize complex constants and boolean variables.}
  get_input;           {Get input filename and read file.}
  calculate_performance;
end.
{=============================================================================}
```

The following is the source file FFT.PAS which contains the Fast Fourier Transform code.

```
{=============================================================================}

procedure FFT(transform: sense; VAR fvct:wave ;
          VAR fvctr,fvcti:freqvector) ;

{ Called during the Modified Harmonic Balance Program.}
{  Sin/Cos FFT subroutine.  Ref: Ch. 10, "The Fast Fourier Transform"
   by E. Oran Brigham (Prentice-Hall, 1974).

   fvctr is the cosine (a) coefficient; fvcti is the sin (b) coefficient.
   Form of the series is

   f(t)=a0/2 + a1*cos(wt) - b1*sin(wt) + a2*cos(2wt) - b2*sin(2wt) + ...

   and fvctr(0)=DC component, fvctr(n)=nth harmonic, etc.  Components must
   be entered with these sign conventions and twice the DC value for a0, as
   implied above.  kflag=1 implies time to freq conversion; The variable
   "transform" specifies whether the transform is from time tp frequency
   (FFT) or frequency to time (IFT).

   The transform is performed assuming 128 sampling points are taken from
   the time waveform.}


var
   n,nu,nu1,n2,n2p1,jj,jk,k,p,k1,k1n2,l,i,loop : integer ;
   treal,timag,c,s,coef,arg : real ;
   xreal,ximag : ARRAY [1..128] OF real ;
   index,index1,index2,m:  integer;

function J_TO_THE_K(j,k:integer) : integer ;
{=============================================================================}
var
  x,i : integer ;
begin
```

```
  x := 1 ;
  for i := 1 to k do
  begin
    x := x * j ;
  end ;
  j_to_the_k := x ;
end ;

function IBITR(j,nu:integer) : integer ;
{-------------------------------------------------------------------------------}
var
  j1,j2,i,x : integer ;
begin
  j1 := j ;
  x := 0 ;
  for i := 1 to nu do
  begin
    j2 := j1 DIV 2;
    x := x * 2 + (j1 - 2 * j2) ;
    j1 := j2 ;
  end;
  ibitr := x ;
end;

{                MAIN PROGRAM                }
{-------------------------------------------------------------------------------}
begin
  n := 128;
  nu := 7;
  n2 := n DIV 2 ;
  n2p1 := n2 + 1 ;
  if transform = 'FFT' then
  begin
    for jj := 1 to n do
    begin
      xreal[jj] := tvct[jj-1];
      ximag[jj] := 0.0;
    end;
  end
  else begin
    for jj := 1 to n2 do
    begin
      jk := n-jj+1;
      xreal[jj] := 0.5*fvctr[jj-1];
      xreal[jk] := 0.5*fvctr[jj];
      ximag[jj] := -0.5*fvcti[jj-1];
      ximag[jk] := 0.5*fvcti[jj];
    end;
    ximag[1] := 0.0;
    ximag[n2p1] := 0.0;
  end;
  nu1 := nu-1;
  k := 0;
  FOR l := 1 TO nu DO
  BEGIN
    REPEAT
      FOR i := 1 TO n2 DO
      BEGIN
        m := j_to_the_k(2,nu1) ;
        p := k DIV m;
        p := ibitr(p,nu) ;
        arg := (p * 6.283185307) / n ;
        c := cos(arg) ;
```

```
        s := sin(arg) ;
        k1 := k + 1 ;
        k1n2 := k1 + n2 ;
         treal := xreal[k1n2] * c + ximag[k1n2] * s ;
         timag := ximag[k1n2] * c - xreal[k1n2] * s ;
         xreal[k1n2] := xreal[k1] - treal ;
         ximag[k1n2] := ximag[k1] - timag ;
         xreal[k1] := xreal[k1] + treal ;
         ximag[k1] := ximag[k1] + timag ;
        k := k + 1 ;
      END ;
      k := k + n2 ;
    UNTIL k >= n ;
    k := 0 ;
    nu1 := nu1 - 1 ;
    n2 := n2 DIV 2 ;
  END ;
  for k := 1 to n do
  begin
    i := ibitr(k-1,nu)+1;
    if i>k then
    begin
      treal := xreal[k];
      timag := ximag[k];
      xreal[k] := xreal[i];
      ximag[k] := ximag[i];
      xreal[i] := treal;
      ximag[i] := timag;
    end;
  end;
  if transform = 'FFT' then
  begin
    coef := 2.0/n;
    for loop := 1 to n2p1 do
    begin
      fvctr[loop-1]:=xreal[loop]*coef;
      fvcti[loop-1]:=ximag[loop]*coef;
    end;
     fvcti[n2p1-1] := -fvcti[n2p1-1]
  end
  else begin
    for loop := 1 to n do
    begin
      tvct[loop-1] := xreal[loop];
    end;
  end;
end;
```

APPENDIX III: UNILATERAL FINLINE ANALYSIS PROGRAM

The procedures 'impedance', 'plusplus', 'plusmoins', 'moinsmoins', as well as all comments in French were written by Solange Monnier. The rest of this listing is the author's work, based on the Accelerated Spectral Domain Technique as suggested by Dr. Hoefer.

```
                {FILE UNIFIN.PAS}

program unifin(input,output);

{ This program calculates the effective dielectric constant, the propagation constant and the power-voltage impedance of
unilateral finlines. One constant basis function is used to approximate the transverse electric field in the slot. The
characteristic equation is approximated over the desired range of terms using estimates for epsilon effective and/or cutoff
frequency. This reduces greatly computation time with only small loss of accuracy. }

const
  c= 2.9979e11;

var
  ic,n1,n2,block:    integer;
  l,neta:            integer;
  ep0,mu0,result:    real;
  a,b,s,epr,d,h:     real;
  f,fc,fce:          real;
  k1,k2,k3:          real;
  ica,icd,lumpsum:   real;
  bet1,bet2,bet3:    real;
  res1,res2,res3:    real;
  eeffe,eeff,ko:     real;
  ob,omg,ash,g12:    real;
  Ey2,P1,P2,P3:      real;
  eta1,gd:           real;
  output:            boolean;
  diag,est:          char;


function sinh(x: real): real;
begin
  if x>42 then x:=42;
  sinh:=0.5*(exp(x)-exp(-x));
end;

function cosh(x: real): real;
begin
  if x>42 then x:=42;
  cosh:=0.5*(exp(x)+exp(-x));
end;

function tanh(x: real): real;
begin
  if x<14
  then tanh:=(exp(x)-exp(-x))/(exp(x)+exp(-x))
  else tanh:=1;
end;

function tan(x: real):real;
begin
  tan:=sin(x)/cos(x);
end;

procedure guesseff;
{ Epsilon effective is estimated.}

var
```

```
  eta1:=(s/a)/(b/a)/(d/b);
  eta2:=eta1*d/b;
  gd:=eta1*arctan(1/eta1)+ln(sqrt(1+sqr(eta1)));
  ga:=eta2*arctan(1/eta2)+ln(sqrt(1+sqr(eta2)));
  xb:=2*ln(1/sin(pi*d/2/b))+epr*(gd+ga);
  y:=a/s;
  if (d/b)<=0.5
  then
    Fsa:=-25.1223+31.524*ln(y)-12.504*sqr(ln(y))+1.9454*ln(y)*sqr(ln(y))
  else begin
    if (d/b)<=0.75
    then
      Fsa:=-33.934+42.451*ln(y)-17.057*sqr(ln(y))+2.5885*ln(y)*sqr(ln(y))
    else
      Fsa:=-48.0487+59.2846*ln(y)-23.77*sqr(ln(y))+3.47*ln(y)*sqr(ln(y));
  end;
   lcd:=2*(a-s)*sqrt(1+nv*(1-(s/a)*(-0.0769*epr+1.231)*Fsa)*xb);
   lca:=2*a*sqrt(1+(4*b/pi/a)*(1+0.2*sqrt(b/a))*ln(1/sin(pi*d/2/b)));
   eeffe:=sqr(lcd/lca)-sqr(c/f/lca);
end;


procedure getInput;

begin
  clrscr;
  writeln('      Spectral Domain Analysis of Unilateral Finlines');
  writeln(' ');
  writeln(' This program calculates the unilateral finline parameters using');
  writeln('the accelerated SDA-method. A single basis function is used for');
  writeln('the slot field.');writeln(' ');
  writeln('  Please specify the following data:');
  write('Waveguide width  (a) in mm          > ');readln(a);
  write('Waveguide height (b) in mm          > ');readln(b);
  write('Metallization distance from wall (h) in mm> ');readln(h);
  write('Fin line gap width (d) in mm        > ');readln(d);
  write('Substrate thickness (s) in mm       > ');readln (s);
  write('Relative dielectric constant (EPR)  > ');readln(epr);
  write('Operating frequency in GHz?         > ');readln(f);
  writeln(' ');writeln('What do you want to calculate?');
  writeln('  1- cutoff frequency only');
  writeln('  2- dispersion characteristics and impedance');
  write('  3- all                    > ');readln(ic);writeln(' ');
  writeln('Would you like :');
  write('   - diagnostics (y/n)?          > n');gotoxy(39,22);
  readln(diag);if (diag='y') or (diag='Y') then diag:='y';
  write('   - to see estimates (y/n)?     > n');gotoxy(39,23);readln(est);

  f:=f*1e9;          { mise en Hz de la frequence }
(* Formules iteratives pour les bornes des equations caracteristiques. *)
  if (d/b)>0.05
  then begin
    n1:=trunc(250*exp(-0.2314*ln(d/b)));  { determine number of terms required for calculation of epsilon effective and Zo }
    n2:=trunc(50*exp(-0.769*ln(d/b)));    {  and beta}
  end
  else begin
    n1:=500;
    n2:=500;
  end;
(if cutoff frequency is to be calculated, find initial estimate)
  if ic<>2
  then begin
     eta1:=(s/a)/(b/a)/(d/b);
```

```
      gd:=eta1*arctan(1/eta1)+ln(sqrt(1+sqr(eta1)));
       fce:=c/2.2/(a-s)/sqrt(1+4/pi*(1+0.2*sqrt(b/(a-s)))*b/(a-s)*
      ln(1/sin(pi*d/2/b)+epr*gd));
    end;
  )
    if ic <>1
    then begin
      guesseff;
      l:=trunc(3*exp(-0.6*ln(d/b)));   {nbr de termes de la sum intermediaire}
      if eeffe<0
      then begin
        writeln('Below cutoff frequency.');
        ic:=99;
      end;
    end;
    if (est='y') or (est='Y')
    then begin
     writeln(' ');
     writeln('Number of spectral terms for Zo and the cutoff frequency is ',n1);
     writeln('   Estimated cutoff frequency is ',fce*1e-9:6:3,' GHz');
     writeln('Number of spectral terms for epsilon effective and beta is ',n2);
     writeln('Number of iterated terms for finding beta is    ',l);
     writeln('   Estimated epsilon effective is ',eeffe:6:4);
    end;
  end;


procedure gety11(an,bet,k: real; var y11: real);
(* Calcul de l'admittance Y11 a utiliser dans l'equation caracteristique. *)

var
  eeff,g1,g2:          real;
  t1,t2,t3,ta1,ta2,ta3: real;
  eg,r,q,Ay,By:        real;

begin
  eeff:=sqr(bet/k);
  r:=-sqr(k)+sqr(an)+sqr(bet);
  q:=-epr*sqr(k)+sqr(an)+sqr(bet);

  g1:=sqrt(abs(r));
  g2:=sqrt(abs(q));
  eg:=epr/g2;
  t1:=tanh(g1*(a-s-h));
  t2:=tanh(g2*s);
  t3:=tanh(g1*h);
  ta1:=tan(g1*(a-s-h));
  ta2:=tan(g2*s);
  ta3:=tan(g1*h);

  case block of
    1: begin                    { calcul de fc,beta=0 }
      if r<0 then
          Ay:=-(1/g1/ta3+eg*(1/g1/ta1-eg*ta2)/(eg+ta2/g1/ta1))
      else begin
        if q<0 then
          Ay:=1/g1/t3+eg*(1/g1/t1+eg*ta2)/(eg-ta2/g1/t1)
        else begin
          Ay:=1/g1/t3+eg*(1/g1/t1+eg*t2)/(eg+t2/g1/t1);
        end;
      end;

      y11:=k*Ay;
```

```
    end;

    2: begin                    { calcul de beta }
      if r<0
      then begin
           Ay:=-(1/g1/ta3+eg*(1/g1/ta1-eg*ta2)/(eg+ta2/g1/ta1));
           By:=g1/ta3+g2*(sqr(g1)/g1/ta1-g2*ta2)/(g2+sqr(g1)*ta2/g1/ta1);
      end
      else begin
        if q<0
        then begin
           Ay:=1/g1/t3+eg*(1/g1/t1+eg*ta2)/(eg-ta2/g1/t1);
           By:=g1/t3+g2*(sqr(g1)/g1/t1-g2*ta2)/(g2+sqr(g1)*ta2/g1/t1);
        end
        else begin
           Ay:=1/g1/t3+eg*(1/g1/t1+eg*t2)/(eg+t2/g1/t1);
           By:=g1/t3+g2*(sqr(g1)/g1/t1+g2*t2)/(g2+sqr(g1)*t2/g1/t1);
        end;
      end;

      y11:=k*(sqr(an)*Ay-eeff*By)/(sqr(an)+sqr(bet));

    end;
  end;
end;


procedure sums(l,n: integer; var lumpsum: real);
(* Calcul de lumpsum,la somme residuelle {superieure} de l'equation
  caracteristique en utilisant une estimation de fc ou de epsilon. *)

var
  g1,g2,eg:   real;
  beta,y11,an: real;
  t1,t2,t3:   real;
  Ay,By:      real;
  i:          integer;

begin
  lumpsum:=0;
  beta:=sqrt(sqr(ko)*eeffe);          { estimation de beta }

  for i:=l to n
  do begin
    an:=i*2*pi/b;                  { an:coeff de Fourier }
(* Au dela de ce rang,gama1 et gama2 ont des carres positifs *)
    g1:=sqrt(-sqr(ko)+sqr(an)+sqr(beta));
    g2:=sqrt(-epr*sqr(ko)+sqr(an)+sqr(beta));
    eg:=epr/g2;
    t1:=tanh(g1*(a-s-h));
    t2:=tanh(g2*s);
    t3:=tanh(g1*h);

    Ay:=1/g1/t3+eg*(1/g1/t1+eg*t2)/(eg+t2/g1/t1);
    By:=g1/t3+g2*(sqr(g1)/g1/t1+g2*t2)/(g2+sqr(g1)*t2/g1/t1);

    y11:=ko*(sqr(an)*Ay-eeffe*By)/(sqr(an)+sqr(beta));

    lumpsum:=lumpsum+y11*sqr(d*sin(an*d/2)/(an*d/2));
  end;
end;
```

```
procedure res(bet,k: real; l: integer; var resd: real);
(* Calcul de l'equation caracteristique. *)

var
  an,sum,y11,y10: real;
  l:          integer;
begin
  sum:=0;
  an:=0;

  gety11(an,bet,k,y10);              { calcul du 1er terme }
  y10:=sqr(d)/2*y10;

  for l:=1 to l-1
  do begin
    an:=l*2*pi/b;
    gety11(an,bet,k,y11);
    sum:=sum+y11*sqr(d*sin(an*d/2)/(an*d/2)); { somme inferieure }
  end;

  resd:=sum+lumpsum+y10;             { equ. caracteristique }
end;


procedure regulafalsi(var it:integer; var x1,x2,x3,y1,y2,y3: real);
(* Calcul de la racine. *)

var
  hold: real;

begin
  x3:=x1-y1*(x2-x1)/(y2-y1);
  if (abs(x2/x1-1.0)<=1e-8) or (abs(x2/x3-1.0)<=1e-8)
  then begin
    case block of
      1:fc:=k2*1e-9*c/2/pi;
      2:eeff:=sqr(x2/ko);
    end;
    output:=true;
  end
  else begin
    hold:=x2;
    x2:=x3;
    x3:=hold;
    y3:=y2;
    it:=it+1;
  end;
end;


procedure cutoff;
(* Determination de fc. *)

var
  it: integer;

begin
  writeln(' ');
  block:=1;
  ko:=2*pi*fce/c;
  lumpsum:=0;
  result:=0;
```

```
k1:=ko*0.9;
it:=0;
output:=false;

res(0,k1,n1+1,res1);
if (diag='y') then writeln('    k1 ',k1:15:10,' res1 ',res1:15:10);
k2:=ko*1.1;
while output<>true
do begin
  res(0,k2,n1+1,res2);
  if (diag='y') then writeln(it:2,'  k2 ',k2:15:10,' res2',res2:15:10);
  fc:=k2*c*1e-9/2/pi;
  if abs(fc-result)<1e-3        { Arret des que le 3ieme chiffre}
  then begin                    {apres la virgule reste stable.}
    if (diag='y') then writeln('No change in 3rd decimal of fc ');
    output:=true;
  end;

  if abs(res2)<=1e-8
  then begin
    if (diag='y') then writeln('Finished due to small res2');
    output:=true;
  end;

  if ((res1*res2)<0) and (output<>true)
  then begin
     regulafalsi(it,k1,k2,k3,res1,res2,res3);
    if it>15
    then begin
      if (diag='y') then writeln('Finished due to it>15');
      output:=true;
    end;
  end;

  if ((res1*res2)>0) and (output<>true)
  then begin
    if it=0
    then begin
       k1:=k1*0.7;
       k2:=k2*1.3;
       res(0,k1,n1+1,res1);
      if (diag='y') then writeln('    k1 ',k1:15:10,' res1 ',res1:15:10);
    end
    else begin
       k1:=k2;
       k2:=k3;
       res1:=res2;
       res2:=res3;
        regulafalsi(it,k1,k2,k3,res1,res2,res3);
    end;
  end;

   result:=fc;
 end;if (diag='y') then writeln(' ');
 writeln('===> The cutoff frequency is    ',fc:6:3,' GHz');
end;


procedure propconst;
(* Determination de beta et de epsilon. *)

var
  it:  integer;
```

```
begin
  writeln(' ');
  block:=2;
  ko:=2*pi*f/c;
  result:=0;
  it:=0;
  output:=false;
  bet1:=sqrt(eeffe)*ko*0.9;
  sums(l,n2,lumpsum);

  res(bet1,ko,l,res1);
  if (diag='y') then writeln('  bet1 ',bet1:15:10,'  res1 ',res1:15:10);
  bet2:=sqrt(eeffe)*ko*1.1;
  while output<>true
  do begin
    res(bet2,ko,l,res2);
    if (diag='y') then writeln(it:2,' bet2 ',bet2:15:10,'  res2 ',res2:15:10);
    (*if abs(bet2-result)<1e-3  { Arret des que le 3ieme chiffre}
    then begin                  {apres la virgule reste stable.}
      output:=true;             {NE MARCHE PAS!!!}
      if (diag='y') then writeln('No change in 3rd decimal of bet2');
    end;*)

    if abs(res2)<=1e-8  {Criterion for termin. of iteration}
    then begin
      if (diag='y') then writeln('Finished due to small res2');
      output:=true;
    end;

    if ((res1*res2)<0) and (output<>true)
    then begin
      regulafalsi(it,bet1,bet2,bet3,res1,res2,res3);
      if it>15  {Program stops after 15 iterations}
      then begin
        if (diag='y') then writeln('Finished due to it>15');
        output:=true;
      end;
    end;

    if ((res1*res2)>0) and (output<>true)
    then begin
      if it=0
      then begin
        bet1:=bet1*0.9;
        bet2:=bet2*1.1;
        res(bet1,ko,l,res1);
      if (diag='y') then writeln('  bet1 ',bet1:15:10,'  res1 ',res1:15:10);
      end
      else begin
        bet1:=bet2;
        bet2:=bet3;
        res1:=res2;
        res2:=res3;
        regulafalsi(it,bet1,bet2,bet3,res1,res2,res3);
      end;
    end;

    result:=bet2;
  end;if (diag='y') then writeln(' ');
  eeff:=sqr(bet2/ko);
  writeln('===> The epsilon effective is      ',eeff:6:3);
  writeln('===> The propagation constant is   ',bet2:6:3,' rad/mm');
```

```
end;


procedure plusplus(n: integer;E,g1,g2: real;var P1,P2,P3: real);
(* Calcul de P1,P2,P3 avec gama1 et gama2 reels positifs. *)

var
  S1,C1,T1,S2,C2,T2,S3,C3: real;
  V,W,Y,Z,SC2:          real;
  P11,P12,P31,P32:        real;
  P21,P22,P23:          real;
  a1,a2,a3,b1,b2,b3:      real;

begin
  S1:=sinh(g1*ash);
  C1:=cosh(g1*ash);
  T1:=tanh(g1*ash);
  S2:=sinh(g2*s);
  C2:=cosh(g2*s);
  T2:=tanh(g2*s);
  S3:=-sinh(g1*h);
  C3:=cosh(g1*h);

  Y:=2*n*pi*E/b/C1/C2/g2/(T2/epr+g12*T1);
  Z:=-bet2*E/omg/mu0/C1/C2/(T1+g12*T2);
  W:=2*n*pi*E/b/g1/S3;
  V:=-bet2*E/omg/mu0/S3;

  SC2:=S2*C2/g2;{writeln('SC2 ',SC2);}

 P11:=ep0*(S1*Y*C1*Y/g1+ash*sqr(Y));
 P12:=mu0*(S1*Z*C1*Z/g1-ash*sqr(Z));
 P1:=ob*(P11+P12)-n*pi*neta*S1*Y*C1*Z;

 a1:=SC2*(sqr(1/epr)+sqr(g12*T1));
 a2:=s*(sqr(1/epr)-sqr(g12*T1));
 a3:=2*sqr(g12*S2)*T1/g1/epr;
 P21:=ep0*epr*sqr(Y*C1)*(a1+a2+a3);
 b1:=SC2*(sqr(T1)+sqr(g12));
 b2:=s*(sqr(T1)-sqr(g12));
 b3:=2*sqr(g12*S2)*T1/g1;
 P22:=mu0*sqr(Z*C1)*(b1+b2+b3);
 P23:=g1*SC2*(1/epr+sqr(T1))+sqr(S2)*T1*(1/epr+sqr(g12));
 P2:=ob*(P21+P22)-n*pi*neta*Y*C1*Z*C1*P23;

 P31:=ep0*(h*sqr(W)-S3*W*C3*W/g1);
 P32:=-mu0*(S3*V*V*C3/g1+h*sqr(V));
 P3:=ob*(P31+P32)+n*pi*neta*S3*W*C3*V;

end;


procedure plusmoins(n: integer;E,g1,g2: real;var P1,P2,P3: real);
(* Calcul de P1,P2,P3 avec gama2-carre negatif et gama1 reel. *)

var
  S1,C1,T1,S2,C2,T2,S3,C3: real;
  V,W,Y,Z,SC2:          real;
  P11,P12,P31,P32:        real;
  P21,P22,P23:          real;
  a1,a2,a3,b1,b2,b3:      real;

begin
```

```
  S1:=sinh(g1*ash);
  C1:=cosh(g1*ash);
  T1:=tanh(g1*ash);
  S2:=sin(g2*s);
  C2:=cos(g2*s);
  T2:=tan(g2*s);
  S3:=-sinh(g1*h);
  C3:=cosh(g1*h);

  Y:=2*n*pi*E/b/C1/C2/g2/(g12*T1-T2/epr);
  Z:=-bet2*E/omg/mu0/C1/C2/(T1+g12*T2);
  W:=2*n*pi*E/b/g1/S3;
  V:=-bet2*E/omg/mu0/S3;

  SC2:=S2*C2/g2;

 P11:=ep0*(S1*Y*C1*Y/g1+ash*sqr(Y));
 P12:=mu0*(S1*Z*C1*Z/g1+ash*sqr(Z));
 P1:=ob*(P11+P12)-n*pi*neta*S1*Y*C1*Z;

 a1:=SC2*(sqr(1/epr)-sqr(g12*T1));
 a2:=s*(sqr(1/epr)+sqr(g12*T1));
 a3:=2*sqr(g12*S2)*T1/g1/epr;
 P21:=ep0*epr*sqr(C1*Y)*(a1+a2+a3);
 b1:=SC2*(sqr(T1)-sqr(g12));
 b2:=s*(sqr(T1)+sqr(g12));
 b3:=2*sqr(g12*S2)*T1/g1;
 P22:=mu0*sqr(C1*Z)*(b1+b2+b3);
 P23:=g1*SC2*(1/epr+sqr(T1))-sqr(S2)*T1*(1/epr-sqr(g12));
 P2:=ob*(P21+P22)-n*pi*neta*Y*C1*Z*C1*P23;

 P31:=ep0*(h*sqr(W)-S3*W*C3*W/g1);
 P32:=-mu0*(S3*V*C3*V/g1+h*sqr(V));
 P3:=ob*(P31+P32)+n*pi*neta*S3*W*C3*V;

end;

procedure moinsmoins(n: integer;E,g1,g2: real;var P1,P2,P3: real);
(* Calcul de P1,P2,P3 avec gama1-carre et gama2-carre negatifs. *)

var
  S1,C1,T1,S2,C2,T2,S3,C3: real;
  V,W,Y,Z,SC2:          real;
  P11,P12,P31,P32:      real;
  P21,P22,P23:          real;
  a1,a2,a3,b1,b2,b3:    real;

begin
  S1:=sin(g1*ash);
  C1:=cos(g1*ash);
  T1:=tan(g1*ash);
  S2:=sin(g2*s);
  C2:=cos(g2*s);
  T2:=tan(g2*s);
  S3:=-sin(g1*h);
  C3:=cos(g1*h);

  Y:=-2*n*pi*E/b/C1/C2/g2/(g12*T1+T2/epr);
  Z:=-bet2*E/omg/mu0/C1/C2/(T1+g12*T2);
  W:=-2*n*pi*E/b/g1/S3;
  V:=-bet2*E/omg/mu0/S3;

  SC2:=S2*C2/g2;
```

```
  P11:=ep0*(ash*sqr(Y)+S1*Y*C1*Y/g1);
  P12:=mu0*(ash*sqr(Z)-S1*Z*C1*Z/g1);
  P1:=ob*(P11+P12)+n*pi*neta*S1*Y*C1*Z;

  a1:=SC2*(sqr(1/epr)-sqr(g12*T1));
  a2:=s*(sqr(1/epr)+sqr(g12*T1));
  a3:=-2*sqr(g12*S2)*T1/g1/epr;
  P21:=ep0*epr*sqr(C1*Y)*(a1+a2+a3);
  b1:=SC2*(sqr(g12)-sqr(T1));
  b2:=-s*(sqr(g12)+sqr(T1));
  b3:=-2*sqr(g12*S2)*T1/g1;
  P22:=-mu0*sqr(C1*Z)*(b1+b2+b3);
  P23:=g1*SC2*(sqr(T1)-1/epr)+T1*sqr(S2)*(1/epr+sqr(g12));
  P2:=ob*(P21+P22)-n*pi*neta*Y*C1*Z*C1*P23;

  P31:=ep0*(h*sqr(W)-S3*W*C3*W/g1);
  P32:=mu0*(h*sqr(V)+S3*V*C3*V/g1);
  P3:=ob*(P31+P32)-n*pi*neta*S3*V*C3*W;

end;


procedure impedance;
(* Calcul de l'impedance tension-puissance. *)

var
  n:            integer;
  r,q,gama1,gama2:   real;
  k0,an,P:        real;

begin
  writeln(' ');
  a:=a*1e-3;
  b:=b*1e-3;
  d:=d*1e-3;
  h:=h*1e-3;
  s:=s*1e-3;
  bet2:=bet2*1e3;
  ash:=a-s-h;
  omg:=2*pi*f;
  k0:=omg/(c*1e-3);
  P:=0.;
  result:=0;
  output:=false;
  n:=0;

  while output<>true
  do begin
     an:=2*n*pi/b;
     r:=-sqr(k0)+sqr(an)+sqr(bet2);
     q:=-epr*sqr(k0)+sqr(an)+sqr(bet2);
     gama1:=sqrt(abs(r));
     gama2:=sqrt(abs(q));
     g12:=gama1/gama2;

     if n=0 then
     begin
       Ey2:=d/b;
       neta:=1;
     end
     else begin
       neta:=2;
       Ey2:=sin(n*pi*d/b)/n/pi;
```

```
    if odd(n)=true then Ey2:=-Ey2;
   end;
   ob:=omg*bet2*b*neta/4;

   if q>0 then
      plusplus(n,Ey2,gama1,gama2,P1;P2,P3)
   else begin
    if r>0 then
      plusmoins(n,Ey2,gama1,gama2,P1,P2,P3)
    else
      moinsmoins(n,Ey2,gama1,gama2,P1,P2,P3);
   end;

   P:=2*((P1+P2+P3)/(sqr(an)+sqr(bet2)))+P;
if abs(sqr(d)/P-result)<1e-3  { Arret des que le 3ieme chiffre}
   then begin                 {apres la virgule reste stable.}
    if (diag='y') then writeln('No change in 3rd decimal of Zo');
    output:=true;
   end;
   if n=n1 then output:=true;
   n:=n+1;
   result:=sqr(d)/P;
 end;
   writeln('===> The impedance Z(V,P) is     ',sqr(d)/P:6:2,' Ohms');
end;

            {MAIN BLOCK BEGINS}

begin
  textmode(bw80);        { screen mode: 25 lignes de 80 caracteres }
  getinput;
  ep0:=8.854188e-12;       { permitivite du vide en F/m }
  mu0:=pi*4e-7;           { permeabilite ----- en H/m }

  case ic of
    1: cutoff;
    2: begin
         propconst;
          if abs(eeff-eeffe)>(0.1*eeff)
         then begin
            eeffe:=eeff;
            propconst;
         end;
        impedance;
      end;
    3: begin
        cutoff;
        propconst;
          if abs(eeff-eeffe)>(0.1*eeff)
         then begin
            eeffe:=eeff;
            propconst;
         end;
        impedance;
      end;
  end;
end.
```