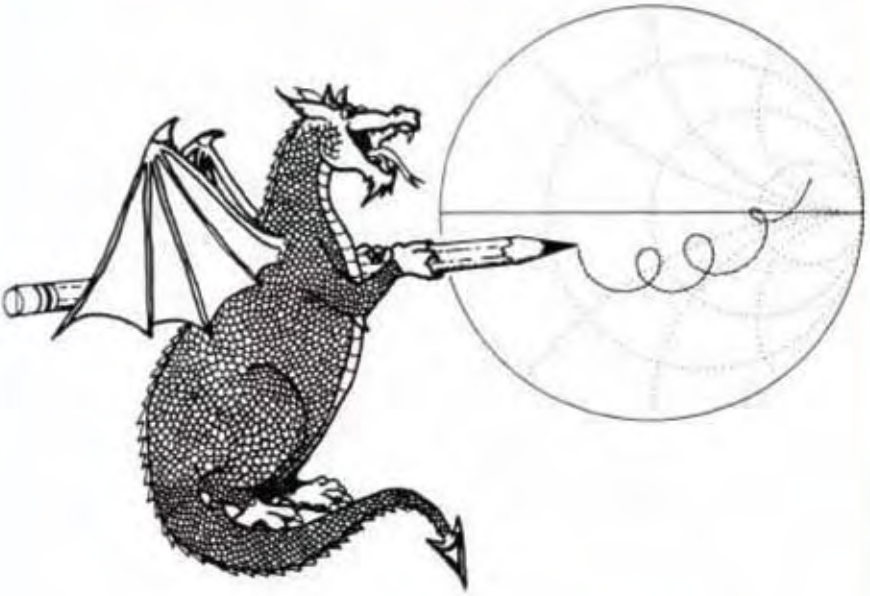
PUFF

Computer Aided Design for Microwave Integrated Circuits



Scott W. Wedge
Richard Compton
David Rutledge

PUFF

Computer Aided Design for Microwave Integrated Circuits Version 2.0

Scott W. Wedge *California Institute of Technology*

Richard Compton *Cornell University*

David Rutledge *California Institute of Technology*

Illustrations by **Dale Yee**

Distribution:

A package containing both printed PUFF manual and Software on diskette is available from:

UKW-Berichte
P.O. Box 80
D-91081 Baiersdorf, GERMANY
Fax +49 9133 779833
email: ukwberichte@aol.com

Support:

If you discover bugs or have problems, please FAX a detailed list of the commands that lead to the problem, your daytime phone or FAX number, and a description of your computer system to Puff Support at +1.818.395.2137.

Preface

PUFF IS A SCATTERING PARAMETER and layout calculator for microwave circuits. Named after the magic dragon in the Peter, Paul and Mary song, *Puff* originated as a teaching tool for Caltech's microwave circuits course. It was created as an inexpensive and simple-to-use alternative to professional software whose high costs, copy protection schemes, and training requirements create difficulties in the academic environment. *Puff* uses a simple interactive schematic-capture type environment. After a circuit is laid out on the screen using cursor keys, a frequency or time domain analysis is available with a few keystrokes. This process is dramatically faster than with net-lists, and errors are rare since the circuit is always visible on the screen. Intended for students and researchers, public distribution of the program began in 1987. *Puff* use, originally limited to Caltech, UCLA, and Cornell, has since spread to many other universities and colleges. The program has also become popular with working engineers, scientists, and amateur radio operators. Over 8000 copies of versions 1.0 and 1.5 were distributed worldwide.

Puff uses a simple heuristic format. The layout that is created on the *Puff* screen is a scale replica of the circuit, so one cannot avoid discovering how electrical parameters effect component dimensions. In a traditional microwave class, a student may spend many hours deriving a complicated formula, and is then too exhausted to make much sense of it. With *Puff*, we coach our students to play with component values and interpret the effects on the response curves. This helps build their intuition and understanding of the circuit physics. The time-domain plots in *Puff* are also instructive. Even the experienced microwave engineer may never have considered the time-domain

response of a branch-line coupler, and he will find it a good puzzle to interpret the bumps in the plot. *Puff* is also fun. We have overheard otherwise cynical students saying, "You have to be careful; you could play all night."

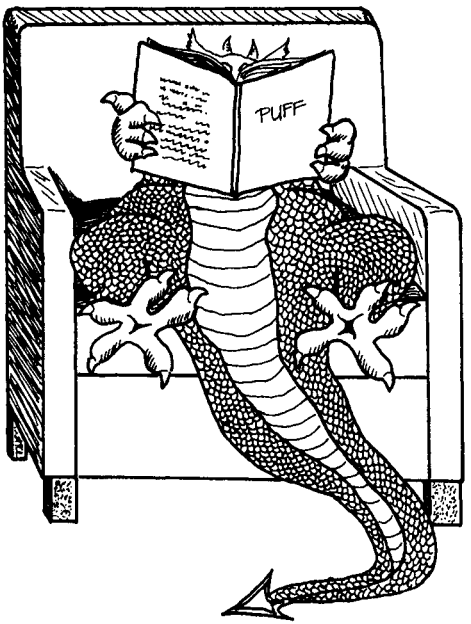
Many technology advances have come since the first release of *Puff*. Clock speed and performance of personal computers have dramatically increased, better graphics displays are now available, and laser printers are now pervasive. Over the years, the average *Puff* user has also become more sophisticated, and has appealed for support of new hardware and for more sophisticated analysis features. To address these needs, we have created version 2.0. Hardware support is now included for *VGA* displays, *LaserJet* printers, and *HP-GL* plotters. New analysis features include the ability to analyze dielectric and metal losses, dispersion in impedance and effective dielectric constant, and effects due to finite strip thickness and surface roughness. As in earlier releases, *Puff* does not automatically model the effects of discontinuities. This must be done manually, and chapter 7 is intended to help with this process. A simple optimizer has been incorporated into *Puff* which we call the *component sweep*. Instead of sweeping with respect to frequency, a circuit may be analyzed at a fixed frequency while sweeping a component's electrical parameter. Despite these extra features, we have tried our best to keep intact that which made the original program appealing: simplicity and ease-of-use.

The development of *Puff* was sponsored by the National Science Foundation, the Jet Propulsion Laboratory, and Hughes Aircraft Company. An Army Research Office equipment grant for computers was instrumental in initiating the work. IBM, the Hewlett Packard Company, and the Rogers Corporation have made generous donations for our classes. At Caltech, Mark Vaughan made many excellent suggestions and helped compare *Puff* results with measurements. Finally, we acknowledge our many beta-testers including students of the 1985 through 1990 EE153 classes at Caltech whose frank comments and diligent bug hunting were instrumental in the evolution of the program.

Pasadena, California
June, 1991

Scott W. Wedge
Richard Compton
David Rutledge

Contents



1. Getting Started

TO RUN *Puff*, you need an IBM PC, AT, PS/2 or compatible, a graphics card and display, and DOS 3.0 or later. For best performance we recommend a computer with 640 kilobytes of memory, a math-coprocessor chip, and a hard disk. Limited operation is possible with less memory, and floating point emulation is used if no coprocessor is present. The program works best with a *Video Graphics Array* (*VGA*) display and adapter. All the screens shown in this manual use the *VGA*. The smaller text in the *VGA* mode makes room for more characters, allowing an extra decimal point of numerical precision to be displayed. *Puff* will also work with the *Color Graphics Adapter* (*CGA*) and the *Enhanced Graphics Adapter* (*EGA*), but the screens will appear different than shown here and some of the numbers displayed will lack the extra decimal. In order to make a hard copy, you will need a screen-dump routine for your display and printer. The following programs have been included on your *Puff* diskette:

`ega2eps.com`	`vga2eps.com`
`ega2pro.com`	`vga2pro.com`
`ega2lasr.com`	`vga2lasr.com`

These are stand-alone programs for doing *EGA* and *VGA* screen dumps on *Epson*, *IBM Proprinters*, and *HP LaserJet* printers, respectively. Use them in the same way as the `graphics.com` program that comes with DOS. Just run the program that matches your graphics card and printer once before you start

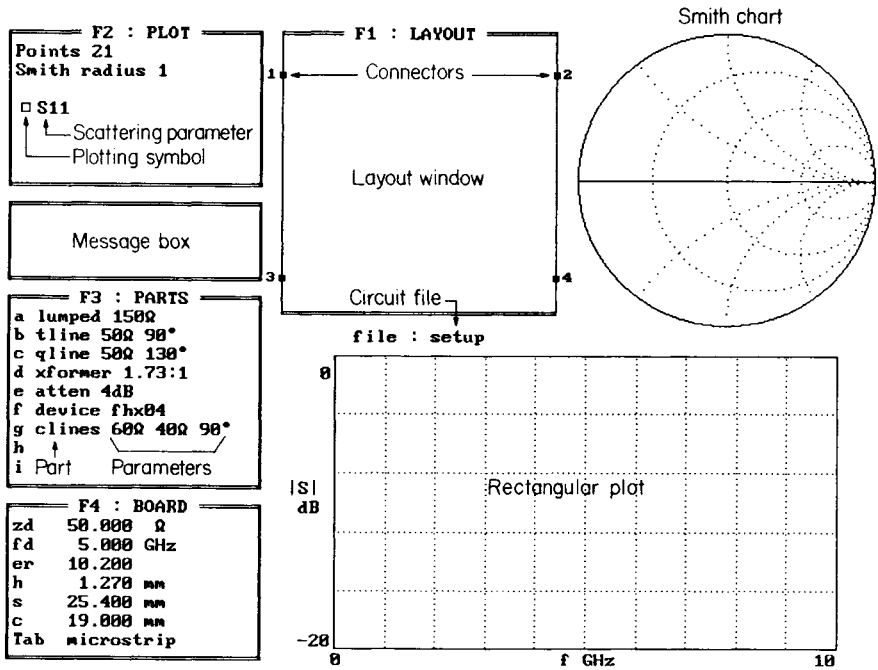


Figure 1.1 A *Puff* screen dump from a *VGA* display. The *VGA* has a resolution of 640 by 480 pixels, and *Puff* will use 16 of the available colors. The *EGA* and *CGA* screens are similar, but will have reduced resolution.

Puff. A screen dump will then be possible using the *PrtSc* key. If you have a *CGA* you can use `graphics.com` for your screen dumps. If you are going to work much with *Puff* you will need an ASCII text editor for modifying circuit files. There is a simple line editor that comes with DOS, `edlin.com`, that may be used. Do not use a word processing editor that inserts extra control characters into the file.

To get started, make a copy of the original *Puff* disk, or if you have a hard disk, make a *Puff* directory and transfer the files to it. If you have not worked with personal computers before, ask a knowledgeable friend to help you get started. To run the program, put the copy in the logged disk drive, or with a hard disk, change to the *Puff* directory. Type `puff` and the return key. The computer will first display a system detect screen with a copyright notice and information concerning the hardware that *Puff* recognizes. After typing another key, the circuit file `setup.puf` will be loaded. *Puff* will automatically

detect the type of display in use and set up a screen similar to Fig. 1.1. If something is wrong with the screen, edit the `setup.puf` file, listed in Fig. 1.2, to try a different display type and override automatic selection. If it still does not work, check for the necessary graphics hardware. Once *Puff* has loaded, the word `setup` will appear on the screen to indicate that this file was used to start. You can specify a different starting file, say `yourfile.puf`, by typing `puff yourfile` when you start the program. *Puff* will add the extension `.puf` if none is given. If you specify a filename other than `setup` when starting, *Puff* will not show the system detect screen.

There are several files on the *Puff* diskette. The program file `puff.exe` and overlay file `puff.ovr` must be present. In addition, the `setup.puf` file should be available, as well as any device files that are called in its parts list. Device files have the extension `.dev`, and there are three on your diskette: a Fujitsu FHX04 HEMT, `fhx04.dev`, a voltmeter, `vmeter.dev`, and a voltage source, `vsource.dev`. In addition to the screen dump programs there are also a few example `.puf` files that are included for reference.

The *Puff* screen is subdivided into *windows*. Movement from window to window is accomplished by pressing function keys *F1* through *F4*. These keys are hot: they are always active. Other hot keys include *Esc* which allows you to exit *Puff* and return to the operating system, *F10* which toggles a small help window, and the cursor keys. At top center is the *Layout* window, where the circuit is constructed. This window is reached by typing the *F1* key. The numbers around the side of the circuit represent external connectors, or ports. There are two connectors on the left side, placed symmetrically above and below the center, and two on the right. The network appearing in the *Layout* window is analyzed from the *Plot* window. Located in the upper left corner of the screen, the *Plot* window is reached by typing *F2*. Here the scattering parameters are specified and their numerical values displayed. The rectangular and Smith chart plots are also controlled from the *Plot* window. The rectangular plot gives either a log-magnitude plot in the frequency domain, or a linear impulse or step response in the time domain. The Smith chart, located at top right, gives a polar plot of the scattering coefficients. The circles inside the Smith chart are curves of constant resistance and reactance. Just below the *Plot* window, in the left center of the screen, is a three-line *Message* box, where error messages and requests for file names appear. Below the *Message* box is the *Parts* window, reached by typing *F3*. This window gives the current list of parts that may appear in the circuit. This list can only be edited from the *Parts* window. The equivalent physical dimensions of the *Layout* window are specified in the *Board* window, accessed by typing *F4*. The parameters given here also influence individual component dimensions.

```
\b{oard} {setup.puf file for PUFF, version 2.0}
d  0 {display: 0 VGA or PUFF chooses, 1 EGA, 2 CGA, 3 Mono}
o  1 {artwork output: 0 dot-matrix, 1 LaserJet, 2 HPGL file}
t  0 {type: 0 microstrip, 1 stripline, 2 Manhattan}
zd   50.000 Ohms {normalizing impedance. zd>0}
fd    5.000 GHz  {design frequency. fd>0}
er   10.200      {dielectric constant. er>0}
h     1.270 mm   {dielectric thickness. h>0}
s    25.400 mm   {circuit-board side length. s>0}
c    19.000 mm   {connector separation. c>=0}
r     0.200 mm   {circuit resolution, r>0, Um=micrometers}
a     0.000 mm   {artwork width correction.}
mt    0.010 mm   {metal thickness, Um=micrometers.}
sr    0.000 Um   {metal surface roughness.}
lt    0.0E+00    {dielectric loss tangent.}
cd    5.8E+07    {conductivity of metal in mhos/meter.}
p     5.000      {photo reduction ratio. p<=203.2mm/s}
m     0.600      {mitering fraction. 0<=m<1}
\k{ey for plot window}
du    0   {upper dB-axis limit}
dl  -20   {lower dB-axis limit}
fl    0   {lower frequency limit. fl>=0}
fu   10   {upper frequency limit. fu>fl}
pts  51   {number of points, positive integer}
sr    1   {Smith-chart radius. sr>0}
S    11   {subscripts must be 1, 2, 3, or 4}
\p{arts window} {O=Ohms, D=degrees, U=micro, |=parallel}
lumped 150O
tline 50O 90D
qline 50O 130D
xformer 1.73:1
atten 4dB
device fhx04
clines 60O 40O 90D
```

Figure 1.2 Listing of `setup.puf`. It is safest to edit a copy of this file, rather than the original. Be very careful when you edit; if some of the board parameters are missing, the program will abort. Comments may be added at the end of the lines in braces, but they should not extend into the next line.

The two most important parameters are the normalizing impedance `zd`, and the design frequency `fd`. The normalizing impedance is used to calculate the scattering parameters. The design frequency is used to calculate the electrical length of transmission lines. Typing *F10* in any of the windows will bring up a small help screen. This screen will usually list the commands that are active for the current window. When in the *Board* window, however, *F10* will provide an explanation of the board parameters.

Exercise 1.1
Maneuver through the *Puff* windows by pressing function keys *F1* through *F4*. In each of the windows, test the effects of typing and retyping the *F10* and *Tab* keys. When done, return *Puff* to its original state.

Exercise 1.2
The spacing of the connectors in the *Layout* window is controlled by parameter `c` in the *Board* window. Hit *F4* to move to the *Board* window. Use the cursor keys and the number keys at the top of your keyboard to set `c` to zero. Type *F1* and see what happens to the connectors. When done, return `c` to its original value.

A step-by-step *Puff* example is now given. We begin with the screen in Fig. 1.1. Hit the *F3* key to make the *Parts* window active. This is indicated by the flashing cursor for part `a`, and the highlighted `F3` at the top of the window. Now hit *F1* to move to the *Layout* window. A large white × will appear in the center of the circuit board, and the first line in the *Parts* window will become highlighted. Part `a` is a lumped 150-Ω resistor. Push ↓, and *Puff* will draw a hollow blue rectangle, labeled `a`, down the screen, and the × will move to the other end of the resistor. In the *Message* box, `Δy -2.540mm` will appear. This shows you the change in the y coordinate. When no length is given for a lumped part, *Puff* assumes it to be one tenth the size of the layout. Now hit the `=` key to ground the end of the resistor and type ↑ to move to the other end. Without the ground, *Puff* thinks that the end of the resistor is open-circuited. Now a 90° section of 50-Ω transmission-line shall be added. This is the `tline` part specified on line `b` of the *Parts* window. Type `b` to select the part, and ↑ to draw it. A copper rectangle is drawn representing a quarter-wavelength transmission-line section at the 5 GHz design frequency (`fd`). The width and length of the line are drawn to scale. In the *Message* box, `Δy 5.690mm` appears, telling you the line length. Next type *F3* to move back to the *Parts* window, and move the cursor down into the description for part `b`. Hit the `=` key again. In the *Parts* window the `=` key is used to display the length and width of the `tline` in the *Message* box. Next, return to the *Layout* window by typing *F1* and then type `1`. You should use the number key at the top of the keyboard rather than the numeric keypad. In

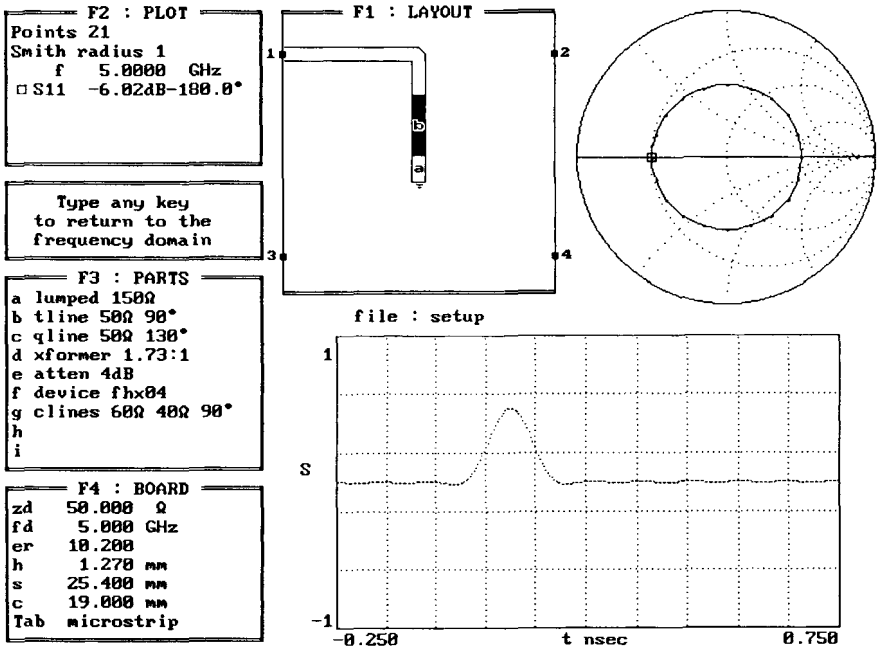


Figure 1.3 Circuit consisting of a quarter-wavelength section of 50-Ω line with a 150-Ω load showing the impulse response.

fact, the *NumLock* key is disabled when *Puff* runs, to avoid mix ups between arrows and digits. *Puff* will make a gray outlined path up and to the left to the first connector. The width of the gray path is the same as that for a 50-Ω line. The point at which it connects to the circuit will be the reference point for calculating phase, and so the length of the gray path has no effect on the analysis. The × will not have moved. This completes our circuit, which consists of a section of 50-Ω line with a 150-Ω load. The screen is shown in Fig. 1.3.

When you type a key, *Puff* first checks for a valid keystroke. If it is not, *Puff* will beep, and do nothing else. You can press `z` to hear the beep. Next, *Puff* checks to see if it can carry out the command. If it cannot, it will give an error message in the red *Message* box. For example, if you retype `1` in the present circuit, *Puff* will say, `Port 1 is already joined`, because multiple connections to the same port are not allowed.

This circuit is ready to be analyzed. Push *F2* to go to the *Plot* window and `p` to plot. The calculated s_{11} values will appear on the Smith chart as small dots joined by a cubic spline curve. A square marker indicates the reflection coefficient at the design frequency, and the numerical values are

given in the *Plot* window. Any magnitude greater than 100 dB will be reported as ∞, and any magnitude as small as -100 dB will be listed as zero. When *Puff* finishes, the *Message* box indicates the time required for the calculation. The plot is a circle of radius one-half on the Smith chart. The rectangular graph shows the magnitude. It is not very exciting because it is a constant, -6.02 dB. To look at the impulse response, push `i`, and give a carriage return when *Puff* prompts for an integer. *Puff* will first calculate the frequency response and then use an inverse fast Fourier transform to calculate the time response. The result is a reflected pulse of height 0.5, delayed 100 ps. The delay comes from the transmission line. The resulting *Puff* screen is shown in Fig. 1.3.

The 150-Ω load may be matched with a quarter-wavelength section. Push the space bar to leave the time-domain plot. A section of 87-Ω transmission line will match a 50-Ω line to the load. The impedance of part `b` is changed by first pushing *F3* to move to the *Parts* window. Use the cursor keys to move to the `5` on line `b`, and type `87` to change the impedance. Typing the `=` key displays the new dimensions of the `tline` in the *Message* box. Now push *F2* to return to the *Plot* window. *Puff* will update the circuit, and you will notice that the transmission-line section has become narrower, because of the impedance increase. Push `p` to plot again. The screen will appear as in Fig. 1.4. The circle on the Smith chart has shrunk so that it passes through the origin at the design frequency. The curve on the magnitude plot is off the screen at `fd`, but you can check the numerical value in the *Plot* window, which shows s_{11} equal to -46.78 dB.

We conclude by changing to a stub-matching circuit. Type *F3* to return to the *Parts* window and change both impedance and length specifications for part `b` to read `tline 50Ω 60°`. If you need to replace an Ω or $^\circ$ symbol, use the *Alt*-`o` or *Alt*-`d` keys, respectively. Type *F1* to go to the *Layout* window, and check that the $\times$ is at the top of the `b` transmission-line. Type `c` and $\rightarrow$ to add the open circuit stub-matching section shown in Fig. 1.5. As specified here, the `qline` part used for `c` is identical to a `tline`. Push *F2* and `p` to analyze. This circuit matches over a narrower frequency band than the quarter-wave match.

Exercise 1.3

The `qline` is equivalent to the `tline`, but with loss effects included. To change the `qline` from a lossless transmission line to that with $Q = 10$ at `fd`, append the specification for part `c` to read `qline 50Ω 130° 10Q`. What effect does this have on the analysis? What are the differences between substituting `10Qc` and then `10Qd`?

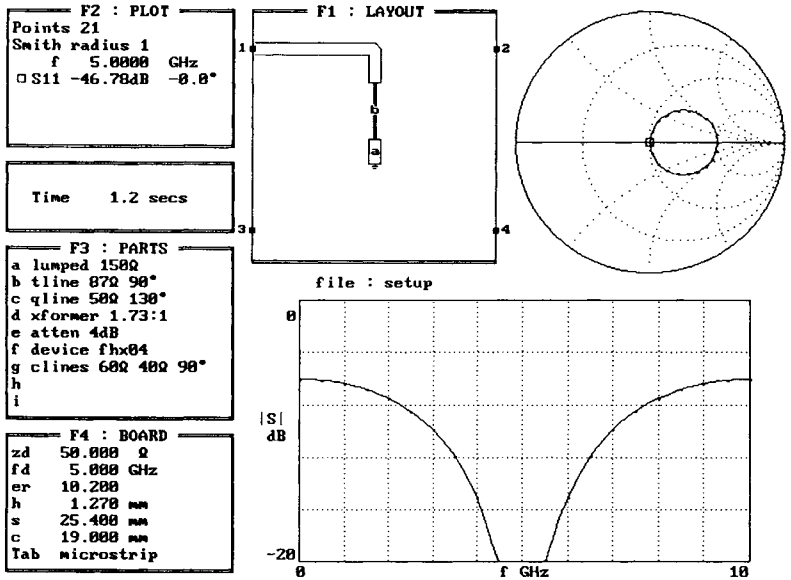


Figure 1.4 Quarter-wave matching.

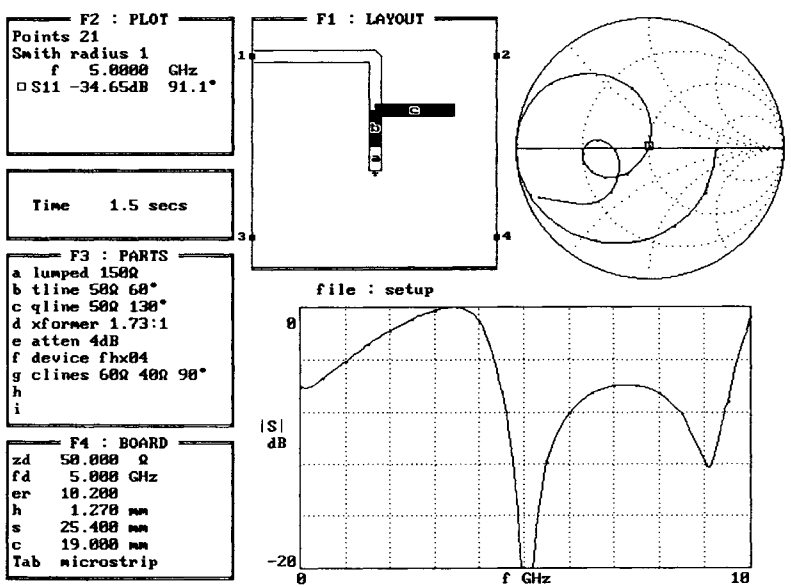


Figure 1.5 Shunt-stub matching.

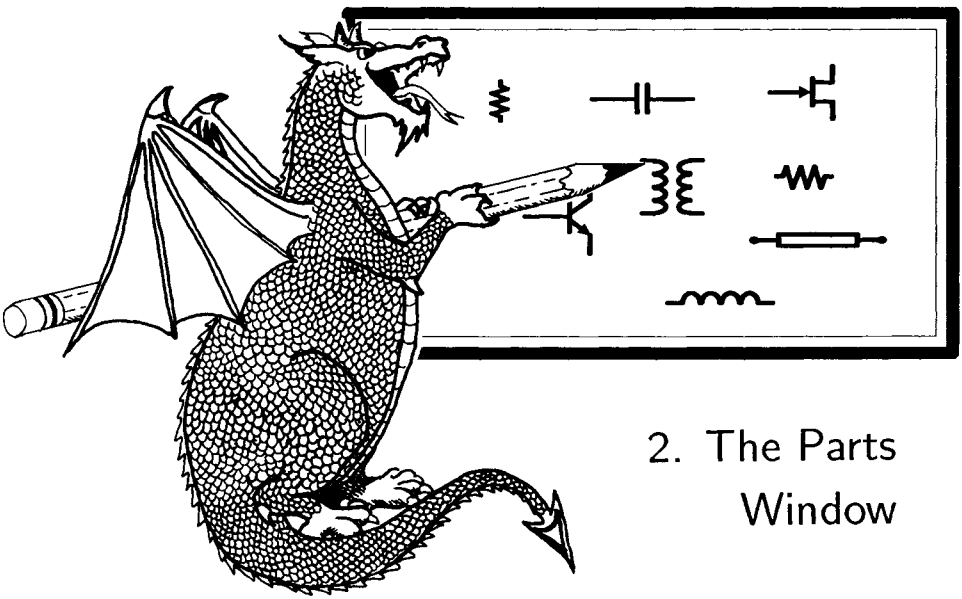
Exercise 1.4
Go into the *Layout* window and make sure that the × is at the right of part `c`. Hold down the shift key and type ←. Such a shift-cursor operation in the direction of a part will cause it to be erased. Hold down the shift key and type `1` to erase the path to the connector. What happens if the shift-cursor operation is not in the direction of a part?

Exercise 1.5
Erase the entire layout by typing *Ctrl*-`e` from either the *Parts* or *Layout* window. Reattempt the 150-Ω matching problem using the `xformer` part. How does the bandwidth compare?

Exercise 1.6
Puff will automatically analyze a `tline` or `cline` for losses and dispersion when their specification includes an exclamation point. Repeat the quarter-wave matching procedure, but this time substitute `tline!` for `tline`. Compare the plots with and without the exclamation point.

Exercise 1.7
The file `fhx04.dev` that was included on your *Puff* diskette contains the two-port scattering parameters for the Fujitsu FHX04 high electron mobility transistor. Erase the current layout by typing *Ctrl*-`e`, select and draw this `device` part, and connect it between ports 1 and 2. Plot its scattering parameters. What happens when you repeat this procedure after substituting `indef` for `device` in the part's specification. How can you get the same scattering parameters using `indef` as you did with `device`?

Exercise 1.8
Erase the layout, go to the *Board* window, and use the `Tab` key to select a *Manhattan* circuit. See what happens when you then draw the `tline` and `clines`.



2. The Parts Window

THE PROGRAM BEGINS in the *Parts* window. The initial parts list is taken from the setup file. A previously saved circuit file can be read in by typing *Ctrl*-`r`. *Puff* will prompt for a file name, and will add `.puf` if no extension is given. You can edit the parts list with the arrow keys, the backspace key, the carriage return (or enter) key, *Ins*, and *Del*. The first letter on each line, from `a` to `i`, identifies each part for use in the layout. By hitting the *Tab* key, the parts list may be doubled in size, allowing extra parts `j` through `r` to be defined. Hitting the *Tab* key a second time will shrink the parts list back to its normal size, unless an extra part was defined and used in the layout. The types of parts available in *Puff* are listed in Fig. 2.1, including examples of their use. Only the first letter of a part name is needed for it to be recognized: `l` is equivalent to `lumped`, `t` is equivalent to `tline`, `q` is equivalent to `qline`, etc. Free up space in the parts list by using single character part descriptions. Commas and periods are both treated as decimal points. Spaces are ignored, except around `device` and `indef` file names. When you leave the *Parts* window, the list will be checked. There will be an error message if a part is longer than the board or shorter than the circuit resolution `r` given in the circuit file. *Puff* will redraw the circuit if it has changed, and report an error if changes result in a part being drawn off the board.

Part	*Description and Examples*
`atten`	Ideal attenuator. Enter attenuation in dB: `atten 3dB`
`xformer`	Ideal transformer. Enter turns ratio: `xformer 2:1`
`lumped`	Resistor, capacitor, and/or inductor: `lumped 50Ω||1nH||1pF` {parallel RLC} `lumped 10+j10-j10Ω` {series RLC resonant at `fd`}
`tline`	Ideal transmission line. Enter impedance and length: `tline 50Ω 90°` {$\lambda/4$ length at `fd`} `tline 1.0z 5mm +1.0h` {artwork length correction added} `tline! 50Ω 90°` {analyze with advanced models}
`qline`	Finite Q `tline`: `qline 50Ω 90° 75Qd` {$Q = 75$ due to dielectric losses} `qline 20mS 90° 50Qc` {$Q = 50$ due to conductor losses}
`clines`	Ideal coupled transmission lines. Enter Z_e, Z_o, and length: `clines 60Ω 90°` {Z_e specified} `clines 60Ω 40Ω 6mm` {Z_e and Z_o specified} `clines! 60Ω 90°` {analyze with advanced models}
`device`	Read file with s-parameter data. Specify filename: `device fhx04` {read filename `fhx04.dev`} `device atf131.s2p` {read file in EEsof format}
`indef`	Indefinite s-parameter generator. Converts n port s-parameter file to $n+1$ port device. Specify filename as in `device`.

Figure 2.1 Descriptions and examples of the parts available in *Puff*.

The simplest part available in *Puff* is the attenuator, written as `atten`. It is drawn as an open blue square with red dots denoting its two ports. It is ideal; *Puff* will always consider it as matched to the normalizing impedance `zd`, and its attenuation will be frequency independent. The desired amount of attenuation is entered in `dB`. If no units are entered, *Puff* will assume dB. The `atten` part is reciprocal and symmetrical.

Exercise 2.1
Plot the scattering parameters for an `atten` part that has a negative value of attenua-

tion. Why is this different than an ideal amplifier?

The `xformer` part denotes a lossless and frequency independent transformer. The only numerical parameter needed is the (unitless) turns ratio. *Puff* accepts a single real number for the turns ratio, not a ratio of two numbers. However, a colon may be used to write the number as a ratio to one. Therefore, `1.5:1` is a valid entry, whereas `3:2` is not. The transformer is antisymmetric, so it is drawn in the layout as a trapezoid. Red dots are again used to denote the ports. For an $n:1$ transformer, the wide end of the trapezoid represents the n side, while the narrow end represents the 1 side. If a negative turns ratio is entered, *Puff* will add a 180° phase shift to the transmission parameters.

Exercise 2.2
Layout and plot the scattering parameters for an ideal 180° phase shifter using an `xformer` with a `-1:1` turns ratio. How would the schematic diagrams differ for a positive and negative turns ratio? Create a three-port 0°/180° power splitter using two transformers with turns ratios of $\sqrt{2}$ and $-\sqrt{2}$.

The `lumped` part is used to specify series and parallel combinations of resistance, capacitance, and/or inductance. It is drawn as an open blue rectangle. *Puff* understands four units for impedance and admittance: Ω (Ohms, typed as *Alt*-`o`), `S` (Siemens), `z` (normalized impedance), and `y` (normalized admittance). For example, a 100-Ω resistor may be specified as `100Ω`, `0.01S`, `2z`, or `0.5y`, assuming that `zd` is 50 Ω. Capacitance and inductance values are entered in units of Farads (`F`) and Henries (`H`), respectively. Note that these must be upper case. Resistance, capacitance, and inductance values may be positive, zero, or negative. Reactance values are specified by placing a `j` before or after a number, and using the impedance units Ω or `z`. For example, `25jΩ` and `0.5jz` specify a 25-Ω reactance at `fd`. *Puff* scales positive reactances proportionally and negative reactances inversely with frequency, interpreting them as inductive and capacitive reactances, respectively. This means that at `2fd`, $50j\,\Omega$ becomes $100j\,\Omega$, and $-50j\,\Omega$ becomes $-25j\,\Omega$. Specifications with the admittance units `S` and `y` are treated in a dual way: positive susceptances scale proportionally with frequency, and the negative susceptances scale inversely.

Series circuits with two or three different lumped elements are specified by combining a real number, a positive imaginary number, and a negative imaginary number, all in the same `lumped` part. For example, `1+j10-j10z` specifies a circuit that is resonant at the design frequency with a Q of 10. The resistance is equal to `zd` and the inductive and capacitive reactance are 10 `zd` at the design frequency. Note that the unit appears only once, after all the

numbers. This type of lumped element entry is very convenient, for example, when taking values from filter tables. Parallel circuits are specified in a dual way using admittance units `S` and `y`. Series and parallel lumped elements that combine resistance and values of inductance and capacitance may also be formed. *Puff* has a special character for forcing a `lumped` part to be a parallel circuit when it does not contain admittance units. This character is ||, typed as *Alt*-p, and it is referred to as the parallel sign. When used with a `lumped` part, the parallel sign forces the part's description to be interpreted as a parallel circuit, regardless of the unit. This allows part entries such as `50Ω||1nH||1pF` for a parallel RLC circuit. Note that you can use symbols for the engineering prefixes *nano* and *pico*. These symbols may appear in front of units in the parts list, in `.puf` files, and in some *Board* window parameters. *Puff* recognizes the symbols for the following engineering prefixes:

Multiplier	*Prefix*	*Symbol*
10^{18}	exa	E
10^{15}	peta	P
10^{12}	tera	T
10^{9}	giga	G
10^{6}	mega	M
10^{3}	kilo	k
10^{-3}	milli	m
10^{-6}	micro	μ
10^{-9}	nano	n
10^{-12}	pico	p
10^{-15}	femto	f
10^{-18}	atto	a

Note that each symbol is case sensitive, and there is a big difference, for example, between `10mΩ` and `10MΩ`. The μ symbol is obtained by typing ***Alt*-m**.

Exercise 2.3
Ground one terminal and compare the one-port scattering parameters for the following lumped parts:

```
lum 50Ω 1nH 1pF
lum 50Ω||1nH||1pF
lum 50Ω +1nH +1pF
lum 50Ω -1nH -1pF
lum 50Ω||-1nH||-1pF
```

Use *Alt*-p to obtain the || sign and *Alt*-o to obtain the Ω sign. Use the **Tab key, while** in the *Plot* window, to align your plots with the Smith chart circles.

The `atten`, `xformer`, and `lumped` parts do not have electrical length. When included in a circuit they are drawn in a default length, referred to here as the *Manhattan* length. An alternate length may be entered for the `lumped` part, although it is optional. This comes in handy when trying to align parts in the layout. The `lumped` length specification should come last, and the units must be in meters (`m` with an appropriate prefix). The *Manhattan* length is one tenth the size of the layout. All parts using *Manhattan* dimensions will be drawn with the same spacing between terminals.

The `tline` part is an ideal transmission line section. It is lossless and without dispersion. In the layout, it is drawn as a copper rectangle. The characteristic impedance or admittance of a `tline` may be specified in the same way as the resistance of a `lumped` part, except that it must be positive. Length units are required. Valid units include: meters (`m` with an appropriate prefix), `h` (the substrate thickness), and ° (degrees, typed as *Alt*-`d`). A transmission line specified with a 360° length will be one wavelength long at the design frequency `fd`. It is usually convenient to specify degrees other than, say millimeters (`mm`), but sometimes the physical units are useful for aligning a `tline` with other parts. You may also specify an artwork length correction to compensate for discontinuities. This is done by adding a plus or a minus sign and a second length. These corrections are made on the screen and in the artwork, but do not affect the electrical length used in the analysis. For example, the length `90°-0.5h` will be treated in the analysis as a quarter-wavelength section, but it will be drawn shorter on the screen and in the artwork. The `h` units are particularly useful here because open-circuit end corrections and phase shifts in tee junctions are proportional to the substrate thickness (See Figs. 7.2 and 7.3).

Often it happens that a desired electrical length results in a physical length larger than the size of the board, or is perhaps negative for a de-imbedding problem. *Puff* provides for this eventuality by allowing you to force *Manhattan* dimensions upon the `tline`. This is done by placing an `M` (must be upper case) as the last character in the part's description. Then, regardless of the parameters specified, *Puff* will make the `tline` length one tenth the size of the layout. To find out the physical dimensions of a `tline`, place the cursor on the part, and hit the = key. *Puff* will calculate the dimensions for the part, and list them in the *Message* box. The dimensions listed will include any artwork corrections, or will be Manhattan dimensions if these were called for. Manhattan dimensions may be forced on all parts using the `Tab` key in the *Board* window.

The `qline` is similar to the `tline`, but is lossy. In addition to impedance and length, you can add a value for the quality factor, or Q. The attenuation

in the line is calculated by enforcing the given Q at the design frequency `fd`. Outside the design frequency, the attenuation is made to follow one of two models. Specify `Qd` or `Q` if you wish a dielectric loss model, and `Qc` if you prefer a conductor loss model. Manhattan dimensions are requested by placing an `M` (must be upper case) as the last character in the part's description. Artwork corrections are not allowed for the `qline`.

Exercise 2.4
Plot and compare the two-port scattering parameters s_{11} and s_{21} for the following `qline` examples:

```
qline 70Ω 90°
qline 70Ω 90° 10Qd
qline 70Ω 90° 10Qc
qline 70Ω -90° 10QM
```

Use a 5 GHz design frequency and plot from 0 to 50 GHz. If needed, enlarge the log and Smith chart plots for the last example.

The `clines` part is a pair of coupled transmission lines. Its specification is similar to a `tline`, except that either one or two impedances or admittances may be given. If only one appears, then the specification looks the same as a `tline`. If this impedance is larger than `zd`, *Puff* interprets it as the even-mode impedance, and if smaller, as the odd-mode impedance. The other mode impedance is chosen to match the lines by forcing the product of even and odd mode impedances to be $\texttt{zd}^2$. If two impedances are given, the larger is the even-mode, and the smaller the odd-mode impedance. As with the `tline`, use an upper case `M` at the end of a `clines` specification to force Manhattan dimensions.

The `device` part is used to read in a file containing multiport scattering parameters. Files may contain transistor data, measured data to be plotted, or parameters which define idealized parts, meters, or sources. A filename specifying the s-parameters must be given, preceded and followed by a space. *Puff* will assume a `.dev` extension if none is given. An optional length may be specified after the filename, as with the `lumped` part. The `device` is drawn in the *Layout* window as a blue arrowhead with red dots denoting the ports. The dot at the wide end of the arrowhead represents port 1 in the file, and the remaining ports follow at equal intervals along the arrow's axis.

If you wish to make your own `device` files, you should study the format of the file `fhx04.dev` given in Fig. 2.2. There is an optional comment line in braces, followed by a template line. The `f` at the beginning of the template stands for frequency. If left out, *Puff* assumes that the scattering coefficients are independent of frequency. The scattering parameters that follow the frequency in the template may appear in any order, and *Puff* will assume that

```
{FHX04 Fujitsu HEMT (89/90), f=0 extrapolated; Vds=2V, Ids=10mA}
  f        s11          s21          s12          s22
 0.0   1.000   0.0   4.375  180.0   0.000   0.0   0.625   0.0
 1.0   0.982 -20.0   4.257  160.4   0.018  74.8   0.620 -15.2
 2.0   0.952 -39.0   4.113  142.0   0.033  62.9   0.604 -28.9
 3.0   0.910 -57.3   3.934  124.3   0.046  51.5   0.585 -42.4
 4.0   0.863 -75.2   3.735  107.0   0.057  40.3   0.564 -55.8
 5.0   0.809 -92.3   3.487   90.4   0.065  30.3   0.541 -69.2
 6.0   0.760 -108.1  3.231   75.0   0.069  21.0   0.524 -82.0
 7.0   0.727 -122.4  3.018   60.9   0.072  14.1   0.521 -93.6
 8.0   0.701 -135.5  2.817   47.3   0.073   7.9   0.524 -104.7
 9.0   0.678 -147.9  2.656   33.8   0.074   1.6   0.538 -115.4
10.0   0.653 -159.8  2.512   20.2   0.076  -4.0   0.552 -125.7
11.0   0.623 -171.1  2.367    7.1   0.076 -10.1   0.568 -136.4
12.0   0.601  178.5  2.245   -5.7   0.076 -15.9   0.587 -146.4
13.0   0.582  168.8  2.153  -18.4   0.076 -21.9   0.611 -156.2
14.0   0.564  160.2  2.065  -31.2   0.077 -28.6   0.644 -165.4
15.0   0.533  151.6  2.001  -44.5   0.079 -36.8   0.676 -174.8
16.0   0.500  142.8  1.938  -58.8   0.082 -48.5   0.707  174.2
17.0   0.461  134.3  1.884  -73.7   0.083 -61.7   0.733  163.6
18.0   0.424  126.6  1.817  -89.7   0.085 -77.9   0.758  150.9
19.0   0.385  121.7  1.708 -106.5   0.087 -97.2   0.783  139.1
20.0   0.347  119.9  1.613 -123.7   0.098 -119.9  0.793  126.6
```

Figure 2.2 The file for the Fujitsu FHX04 HEMT, `fhx04.dev`.

any parameters that are not given on the template line are zero. The program will figure out how many ports the device has from the highest port number that appears in the template line. *Puff* can handle up to four-port device files. The numbers that follow the template are separated by one or more spaces or carriage returns. The frequency is first (if it appears) followed by the magnitude (linear, not dB) and phase (in degrees) of each of the scattering parameters in turn. When *Puff* is calculating scattering parameters in the *Plot* window, it will interpolate linearly between points in the `device` file, as necessary. *Puff* will not extrapolate beyond a `device` file's frequency range, and will give an error when this is attempted. A previously saved `.puf` file (that includes saved scattering parameters) can also be recalled as a `device` file. Complex networks can then be formed by combining many smaller circuits.

The `indef` part is similar to the `device`, but it is used to generate *indefinite* scattering parameters from a file containing definite scattering parameter

data. Indefinite parameters are those with an undefined ground terminal. If a one-port file is specified, **indef** will convert it to a two-port. If a two-port file is specified, **indef** will convert it to a three-port; and so on up to a four-port to five-port conversion. The **indef** part is most often used to model a transistor as a three port. The n port to $n+1$ port conversion is made possible by assuming that Kirchhoff's current law is valid, allowing what was the ground terminal to be converted to a port. The **indef** part is drawn on the screen in the same way as the **device** part, except the port created from ground is drawn as a yellow dot. The extra port generally appears as the last port, except for the two-port to three-port **indef** where the extra port appears in the center of the part. Note that to turn an **indef** part into its **device** equivalent, the extra port should be shorted.

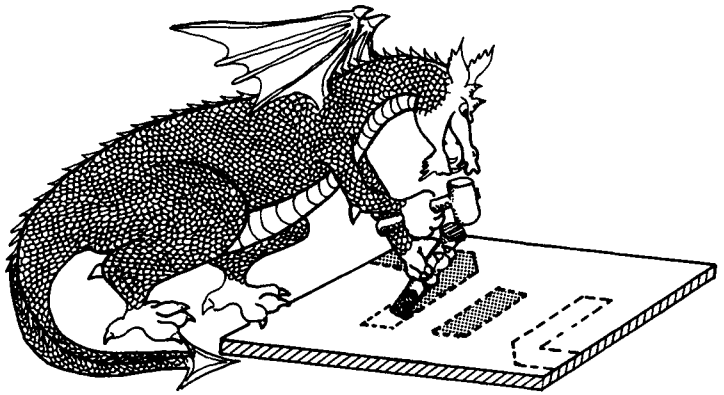
In addition to the **device** file format given in Fig. 2.2, *Puff* can also read s-parameter data files in the EEsof format. This is done when the appropriate file extension is given for **device** or **indef** parts: a one-port file requires a `.S1P` extension, the two-port uses `.S2P`, and so on up to a four-port `.S4P`. *Puff*, however, cannot read every possible data format for these files. It is assumed that they possess the following format:

```
# xHZ S MA R yy
```

where x is the same engineering prefix used for **fd** in the *Board* window, and yy is the value for **zd**. *Puff* reads only scattering parameters given in the magnitude/angle format. These restrictions require some caution. If a device file contains frequencies given in GHz, *Puff* will give erroneous results if you try to plot data in MHz. The prefix for **fd** and the impedance of **zd** in the *Board* window *must* coincide with those used in all **device** and **indef** files. This goes for files in both EEsof and `.dev` formats. Noise parameters present in files are ignored.

Exercise 2.5
Create a two-port circuit and plot all of its scattering parameters (s_{11}, s_{21}, s_{12}, and s_{22}). Use *Ctrl*-**s** from the *Plot* window to save it to a file. Erase the circuit, and then make a **device** part that will recall the scattering parameters from the previously saved file. Be sure to use the `.puf` extension. What happens if the number of points is different?

Exercise 2.6
Generate **device** files for an ideal microwave isolator and circulator. Make them frequency independent by leaving out the **f** frequency designation in the template. Each should be a simple two-line file.



3. The Layout Window

IN THE UPPER PORTION of the screen is the *Layout* window. The square represents the substrate, and the numbers on the sides represent connectors. Typing an arrow key will draw the selected part in the *Parts* window in the direction of the arrow. The *Message* box will show the change in the x and y coordinates. *Puff* starts out drawing part `a`, but you can select another part in the list by typing the letter for the part desired. The circuit can be grounded at any point by pushing the `=` key. If there is already a part in the direction of the arrow key, *Puff* will move to the other end rather than draw over it. If the ends of two parts are closer together than the circuit resolution `r`, *Puff* will connect them together. *Puff* will stop you from drawing a part off the edge, but it will not stop you from crossing over a previous part. You can make a path to a connector by pushing one of the number keys `1`, `2`, `3`, or `4` on the top row of the keyboard. Notice that *Puff* does this by first moving up or down, then right or left. The electrical length of a connector path is not taken into account in the analysis, and it is drawn in a different color to indicate this. You can erase the entire circuit and start over by pushing *Ctrl*-`e`. *Ctrl*-`n` moves the × to the nearest node. This can be useful if you are off the network and want to get back, or if you want to see if two nodes are connected.

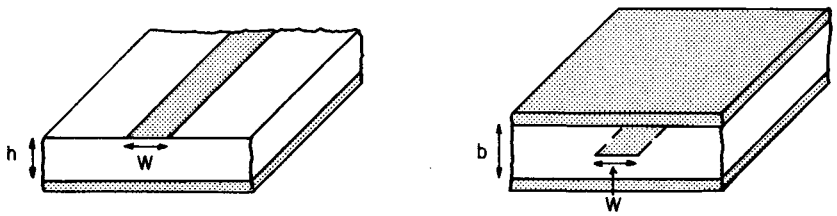
The shift key is used for erasing and moving around the layout. *Puff* will erase a part rather than move over it if you hold the shift key down while pushing an arrow key. A ground can be removed with shift =. The path to a connector is erased by holding the shift key down while typing the connector number. If you are not at a connection to a port, this shift-number operation will move you to that port number without drawing the path. This is useful if you would rather start a circuit at a connector than in the center. The shift-arrow operation moves the × when there is no part to erase. The × moves half the length of the currently selected part. Half steps are used rather than full steps to allow centered and symmetrically laid out circuits. To move the full length of the part, step twice. It is important to note that this shift-arrow operation is actually drawing an invisible part. If you later change the size of an invisible part, it will have an impact on the layout, possibly resulting in a part being drawn off the board. If this happens, *Puff* will give you an error message that you may not believe. As a precaution, keep the number of invisible parts to a minimum.

There are special rules for drawing **clines**. Use the arrows to move along the lines and *Ctrl*-n to jump from one line to another. When connecting **clines** together, if you draw the second **clines** in the same direction as the first, the new lines will join the previous pair. This is the usual arrangement in a directional coupler. If you change directions, the **clines** will be staggered so that only one of the lines in each pair is connected. This is appropriate for a band-pass filter.

Typing *Ctrl*-a from the *Plot* window will activate *Puff*'s photographic artwork routines. The layout produced will be magnified by the photographic reduction ratio (p) in the circuit file. The artwork output parameter (o) in the circuit file allows you to specify dot-matrix or HP LaserJet printouts, or the production of a Hewlett-Packard Graphics Language (HP-GL) file. *Puff* will prompt for titles to be placed atop the printout, or for an HP-GL file name. Only the **tlines**, **qlines**, and **clines** will appear in the artwork, and the corners will be mitered. You may adjust **tline** and **clines** lengths using discontinuity corrections in the parts list. Widths may be adjusted using the artwork width correction factor a in the circuit file.

The rest of this chapter is a summary of the synthesis formulas used for determining the layout dimensions on the screen and in the artwork. We start with the **tline** and **qline** transmission line sections. The microstrip width W (Fig. 3.1a) is given by Wheeler [1,2]. For high-impedance lines,

$$\frac{W}{h} = \frac{8 \exp H'}{\exp(2H') - 2} \tag{3.1}$$



(a) (b)

Figure 3.1 The `tline` dimensions for microstrip (a), and stripline (b).

where

$$H' = \frac{Z_0\sqrt{2(\epsilon_r+1)}}{120} + \frac{1}{2}\left(\frac{\epsilon_r-1}{\epsilon_r+1}\right)\left(\ln\frac{\pi}{2} + \frac{1}{\epsilon_r}\ln\frac{4}{\pi}\right), \tag{3.2}$$

where ϵ_r is the dielectric constant and Z_0 is the given characteristic impedance. For lines with characteristic impedances smaller than $(44 - 2\epsilon_r)\,\Omega$ [3], the microstrip formulas are

$$\frac{W}{h} = \frac{2}{\pi}\left[(d_\epsilon - 1) - \ln(2d_\epsilon - 1)\right] + \frac{\epsilon_r - 1}{\pi\epsilon_r}\left[\ln(d_\epsilon - 1) + 0.293 - \frac{0.517}{\epsilon_r}\right] \tag{3.3}$$

where $d_\epsilon = 60\pi^2/(Z_0\sqrt{\epsilon_r})$. Microstrip lengths in degrees are related to lengths in millimeters by the effective dielectric constant ϵ_{re} [4],

$$\epsilon_{re} = \frac{\epsilon_r+1}{2} + \frac{\epsilon_r-1}{2}\left(1 + 10\frac{h}{W}\right)^{-\frac{1}{2}}. \tag{3.4}$$

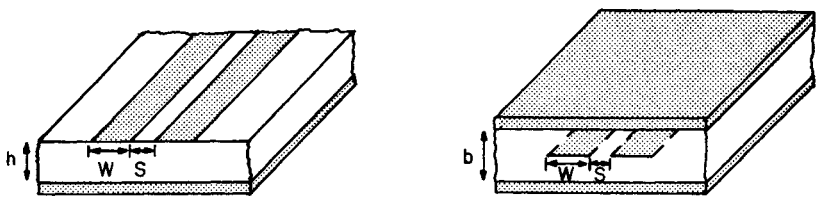
The stripline formulas are simpler because the effective dielectric constant is just ϵ_r. The width W (Fig. 3.1b) is given by Cohn's equations [5]

$$\frac{W}{b} = \frac{2}{\pi}\tanh^{-1}k \tag{3.5}$$

where

$$k = \begin{cases} \sqrt{1 - \left[\frac{e^{\pi x}-2}{e^{\pi x}+2}\right]^4}, & 1 < x; \\ \left[\frac{e^{\pi/x}-2}{e^{\pi/x}+2}\right]^2, & 0 \le x \le 1; \end{cases} \tag{3.6}$$

with $x = Z_0\sqrt{\epsilon_r}/(30\pi)$.



(a) (b)

Figure 3.2 The `clines` dimensions for microstrip (a), and stripline (b).

For `clines`, we start with the even and odd mode impedances Z^e and Z^o. In microstrip, the dimensions W and S (Fig. 3.2a) are obtained by solving the following equations [6],

$$\frac{W_e}{h} = \frac{2}{\pi}\cosh^{-1}\left(\frac{2d-g+1}{g+1}\right)$$

$$\frac{W_o}{h} = \begin{cases} \frac{2}{\pi}\cosh^{-1}\left(\frac{2d-g-1}{g-1}\right) + \frac{1}{\pi}\cosh^{-1}\left(1+\frac{2W}{S}\right), & \text{if } \epsilon_r \geq 6 \\ \frac{2}{\pi}\cosh^{-1}\left(\frac{2d-g-1}{g-1}\right) + \\ \qquad \frac{4}{\pi(1+\epsilon_r/2)}\cosh^{-1}\left(1+\frac{2W}{S}\right), & \text{if } \epsilon_r < 6 \end{cases} \tag{3.7}$$

where $g = \cosh(\frac{\pi S}{2h})$ and $d = \cosh(\frac{\pi W}{h} + \frac{\pi S}{2h})$, and W_e and W_o are the widths of a single microstrip transmission line with characteristic impedance $Z^e/2$ and $Z^o/2$. For ϵ^e_{re} and ϵ^o_{re}, we follow Garg and Bahl [7]. These formulas are long, and are not given here. The length that appears on the screen and in the artwork is the average of the lengths for the even and odd modes.

The stripline formulas for `clines` are simpler. The effective dielectric constant is ϵ_r for both modes. The following formulas give W and S (Fig. 3.2b) [8],

$$\frac{W}{b} = \frac{2}{\pi}\tanh^{-1}\sqrt{k_e k_o}$$

$$\frac{S}{b} = \frac{2}{\pi}\tanh^{-1}\left[\frac{1-k_o}{1-k_e}\sqrt{\frac{k_e}{k_o}}\right] \tag{3.8}$$

where k_e and k_o are obtained by substituting Z^e and Z^o into Eqn. 3.6.

You can inspect *Puff*'s layout calculations from the *Parts* window. Place

the cursor in any `tline`, `qline`, or `clines` description and hit the = key. *Puff* will tell you the length and width for these parts, as well as the spacing for `clines`.

Exercise 3.1
Lay out a simple circuit consisting of `tlines` and `clines`. Go to the *Board* window and use the `Tab` key to change the circuit type. Return to the *Layout* window and see how the circuit is effected. Repeat for microstrip, stripline, and Manhattan layouts. Can you make a microstrip circuit that is difficult to realize in stripline? What stripline circuits cannot be realized in microstrip?

Exercise 3.2
Lay out a simple circuit. Go to the *Plot* window and save the circuit using *Ctrl*-`s`. Exit *Puff*, then use an ASCII editor to open the saved *Puff* file. Go to the section of the file that begins `\c{ircuit}`. *Puff* has saved your keystrokes in the *Layout* window in what we call a *keylist*. When reading a new file, *Puff* uses the keylist to redraw the circuit. What keystrokes are not saved? Is it better to erase all the parts using *Ctrl*-`e`, or with repeated shift-arrow operations? Can you think of some advantages in keeping the keylist as short as possible?

Exercise 3.3
The most common *Puff* layout errors involves invisible parts. Starting from a blank layout, select a `tline` part and make repeated shift-arrow operations to move the × cursor about the *Layout* window. Go to the *Parts* window, increase the length of the `tline` used, then return to the *Layout* window. See how long you can make the `tline` before causing a layout error. What happens if you try to delete the `tline` from the parts list?

References

[1] H. A. Wheeler, "Transmission line properties of parallel wide strips by a conformal mapping approximation," *IEEE Trans. on Microwave Theory and Tech.*, vol. MTT-12, pp. 280–289, May 1964.

[2] H. A. Wheeler, "Transmission line properties of parallel strips separated by a dielectric sheet," *IEEE Trans. on Microwave Theory and Tech.*, vol. MTT-13, pp. 172–185, March 1965.

[3] R. P. Owens, "Accurate analytical determination of quasi-static microstrip line parameters," *The Radio and Electronic Eng.*, vol. 46, pp. 360–364, July 1976.

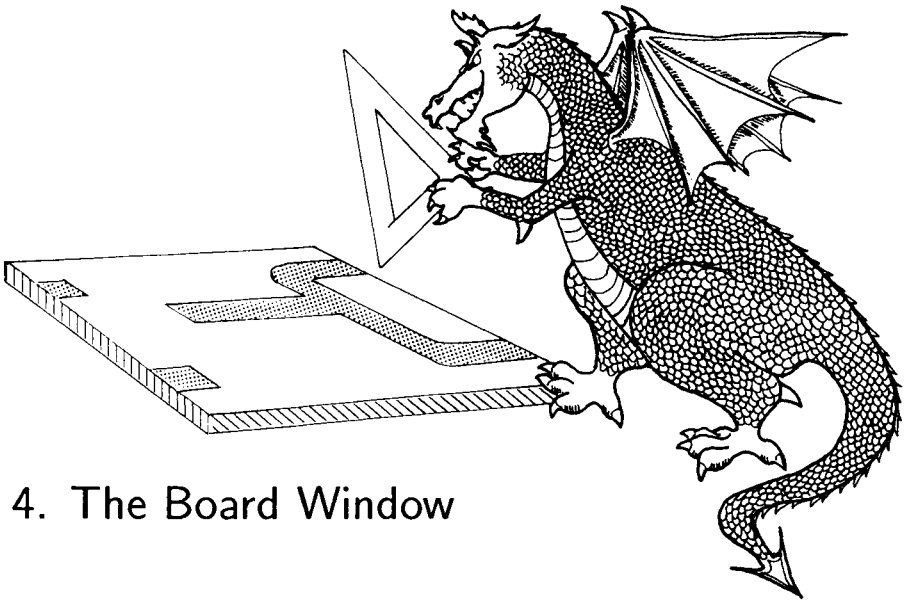
[4] M. V. Schneider, "Microstrip lines for microwave integrated circuits," *Bell System Technical Journal*, vol. 48, pp. 1421–1443, May/June 1969.

[5] S. B. Cohn, "Problems in strip transmission lines," *IRE Trans. on Microwave Theory and Tech.*, vol. MTT-3, pp. 119 126, March 1955.

[6] S. Akhtarzad, T. R. Rowbotham, and P. B. Johns, "The design of coupled microstrip lines," *IEEE Trans. on Microwave Theory and Tech.*, vol. MTT-23, pp. 486–492, June 1975.

[7] R. Garg and I. J. Bahl, "Characteristics of coupled microstriplines," *IEEE Trans. on Microwave Theory and Tech.*, vol. MTT-27, pp. 700–705, July 1979, and corrections *MTT-28*, pp. 272, Mar. 1980.

[8] S. B. Cohn, "Shielded coupled-strip transmission line," *IRE Trans. on Microwave Theory and Tech.*, vol. MTT-3, pp. 29–38, Oct. 1955.



4. The Board Window

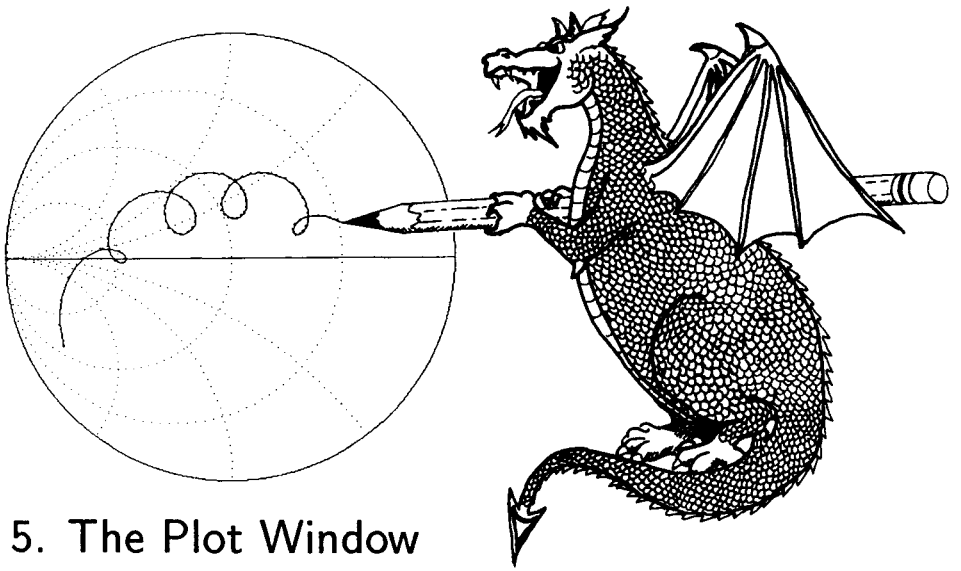
THE RELATIVE DIMENSIONS of the circuit board in the *Layout* window are specified from the *Board* window. These dimensions set the scale used to draw the distributed components on the screen. Figure 4.1 gives a brief description of each of the parameters available. Access the *Board* window by pressing function key *F4*. Edit the parameters using the same keys as in the *Parts* window. *F10* brings up a help window explaining the board parameters.

The normalizing impedance `zd` is used to calculate the scattering parameters. It also defines the normalized impedance and admittance values (`z` and `y`) that may appear in the *Parts* window. Paths to connectors are drawn with transmission lines with impedance `zd`. The design frequency `fd` is used to determine the physical lengths of distributed components entered in degrees. The Hz units for `fd` may use any prefix that appears in the chapter 2 table, although MHz and GHz are generally the most practical. The *Plot* window will inherit the frequency prefix. Change the prefix when you find yourself entering lots of zeros below the log-magnitude plot. Use caution. The prefixes are case sensitive. Also beware of frequency data in `device` and `indef` files that does not coincide with the `fd` prefix, e.g. *Puff* will give meaningless results if you use a device file with GHz frequency data when MHz is used for `fd`. Device scattering parameter data must also match the *Board* window's definition of `zd`.

Parameter	*Description*
`zd`	Normalizing or characteristic impedance. Used in the calculation of scattering parameters. Units are Ω's with optional prefix.
`fd`	Design frequency. Used to compute electrical length of parts entered in degrees. Also the frequency used for the component sweep. Prefix given is carried over into the *Plot* window.
`er`	Relative dielectric constant of substrate. Used to calculate dimensions for microstrip and stripline components. Unitless.
`h`	Substrate thickness. One of three parameters that specify the equivalent dimensions for the *Layout* window. Significant in transmission line calculations.
`s`	Board size. Specifies the equivalent length of each side of the square circuit board that makes up the *Layout* window.
`c`	Connector separation. Sets the spacing between ports 1 and 3, and ports 2 and 4 in the layout. Set to zero to create a centered two-port.
`Tab`	Circuit type. Use the `Tab` key to select a microstrip, stripline, or Manhattan layout.

Figure 4.1 Description of parameters that may be modified from the *Board* window.

The `Tab` key toggles between microstrip, Manhattan, and stripline layouts. Resort to the Manhattan mode when `tlines` or `clines` become too long or short. All distributed parts are then drawn with widths 1/20 the board size, and lengths 1/10 the board size. This mode also permits components with unrealizable values, such as `tline`'s with negative electrical lengths. However, if *Puff* requires an electrical length calculation, stripline models will be used, and physically unrealizable parameters may not be allowed. If enabled, Manhattan dimensions will appear in the artwork, and artwork corrections will be ignored.



5. The Plot Window

TO REACH THE *Plot* window, push function key *F2*. The circuit in the *Layout* window is analyzed by typing `p`. If you type *Ctrl*-`p`, the previous plot will be drawn before the new plot, allowing a comparison of results. After an analysis has been completed, the *Plot* window lists the values of the scattering coefficients at design frequency `fd`. Use the *PgUp* and *PgDn* keys to move the markers and show the scattering coefficients at the other frequencies. The ↑ and ↓ keys can be used to move the cursor to various parameters. They cycle through a loop that includes the *Plot* window and the x and y axes on the rectangular plot. You can type over any parameter to change it. *Puff* can plot up to four different scattering parameters simultaneously. To select an s-parameter, move the cursor down toward the bottom of the *Plot* window, and a marker will appear, together with the letter `S`. Then type in the port numbers for the desired s-parameters. If you leave a line blank, it will be erased when you move the cursor. Those with *EGA* and *VGA* graphics can use the `Tab` key to change the Smith chart from an impedance chart to an admittance chart. Typing *Alt*-`s` with *VGA* graphics toggles an enlarged Smith chart.

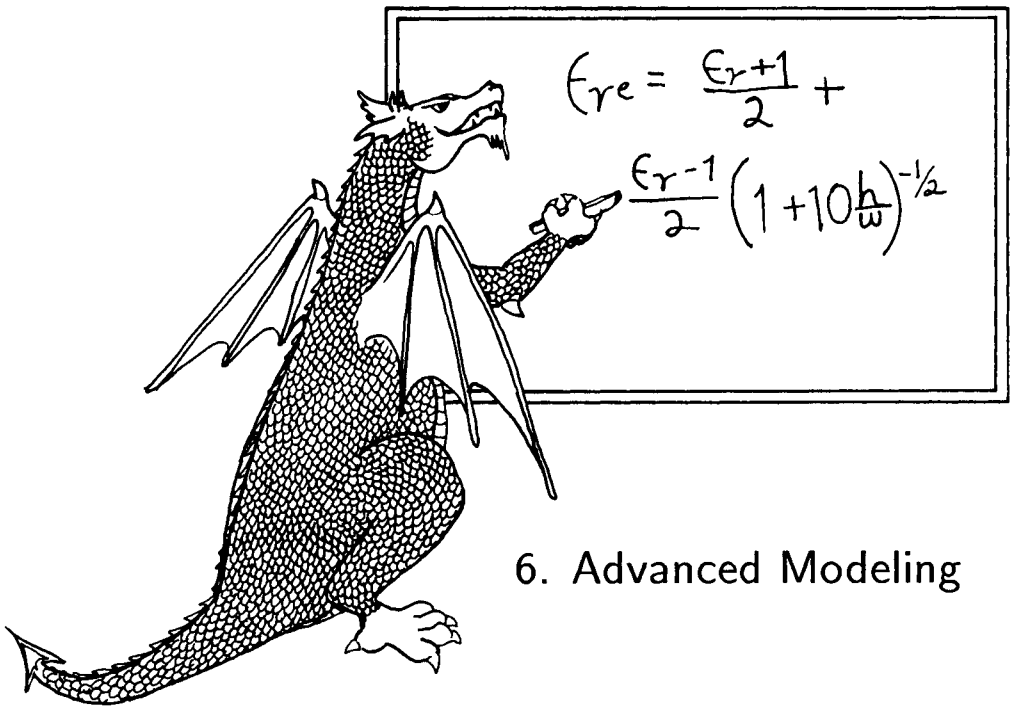
The *Plot* window allows you to select the number of frequency `Points` to analyze. This must be a positive integer no greater than 500, assuming you have a full complement of memory. *Puff* interpolates between calculated

points with a cubic spline. The interpolation is performed by splining the real and imaginary parts of the scattering coefficients separately. The independent variable for calculating the spline curve is the spacing on the Smith chart. This gives better results than using frequency as the splining parameter. If a curve kinks on the Smith chart, or gives erroneous ripples on the rectangular plot, it is an indication that the number of `Points` is too small.

You can plot an impulse response by typing `i`, or a step response by typing `s`. *Puff* will request a frequency interval specified by the ratio `fd/df`. It will then do a 256-point inverse fast Fourier transform of the scattering coefficients, and plot the results on a linear scale. The amplitude of the time-domain plot is the same as the Smith chart radius. The ratio `fd/df` will determine the time axis for the plot, which goes from `−1/8df` to `3/8df`. The upper and lower limits on the frequency axis of the magnitude plot are used for windowing the Fourier transform. The window is a raised cosine that goes to zero at the upper frequency limit. The scattering coefficients are set to zero outside the window.

A convenient way to see the impulse or step inputs is to draw an open-circuited connector path at an unused port. The reflection coefficient for this port is the input waveform. *Puff* normalizes the input waveforms so that the peak value is 1. The high-frequency limit controls the rise and fall times, and the low-frequency limit affects the ringing. Be aware that the time-domain waveforms are actually periodic, with period `1/df`, and aliasing from the previous pulse may affect the response. The step input is actually a square wave, and the response to the previous falling edge will affect the rising step that follows.

To save a network in a circuit file, type *Ctrl*-`s` and give a file name. The *Parts* window and the data in the *Plot* window will be saved along with the circuit. The `.puf` extension will be added to the filename if one is not specified. Typing *Ctrl*-`a` from the *Plot* window will activate *Puff*'s photographic artwork routines. The layout produced will be magnified by the photographic reduction ratio (`p`) in the circuit file. Only the `tlines`, `qlines`, and `clines` will appear in the artwork, and the corners will be mitered. The artwork output parameter (`o`) in the circuit file allows you to specify dot-matrix or HP LaserJet printouts, or the production of a Hewlett-Packard Graphics Language (HP-GL) file. *Puff* will prompt for titles to be placed atop the printout, or for an HP-GL file name. The HP-GL file will be created with the `.hpg` extension. The *Puff* diskette includes the program `hpg2com`. Use this to dump `.hpg` files to a serial plotter connected at port COM1.



6. Advanced Modeling

THE SINGLE AND COUPLED TRANSMISSION LINES represented by the `tline` and `clines` parts are modeled as *ideal* distributed components: their effective dielectric constants and impedances are assumed constant over frequency, and they are lossless. These models neglect dispersion and the many loss effects common in microwave circuits. When these effects must be included in the analysis, more accurate models may be invoked by the addition of an exclamation point (`!`) to the part's description. The `tline!` and `clines!` are therefore referred to as *advanced* models. An analysis with advanced models may be made for a microstrip or stripline circuit and will include effects due to finite strip thickness, conductor and dielectric losses, surface roughness, and dispersion. When using these models, be sure to first specify accurate values for metal thickness, surface roughness, loss tangent, and conductivity in your `.puf` file.

The ideal and advanced parts are not entirely distinct. When either type is entered into the *Parts* window, *Puff* will calculate their effective dielectric constants and physical layout dimensions using the quasi-static *synthesis* formulas of chapter 3. These calculations permit a scale drawing of the part in the *Layout* window, and artwork generation. A transmission line, for example, will therefore be the same size, be it specified a `tline!` or a `tline`. The

differences between ideal and advanced models arise when *Puff* is analyzing for scattering parameters. The ideal `tline` and `clines` are analyzed using the impedance values specified in the parts list and quasi-static effective dielectric constant values. These are held constant during the frequency analysis. Analyses using the advanced `tline!` and `clines!` models are based solely on the physical dimensions that were derived for the layout. Impedance and electrical length parameters entered in the parts list, although used to derive the dimensions, are ignored during the advanced frequency analysis. This is done to allow these parameters to vary over frequency due to dispersion. Often an advanced analysis based on a component's dimensions will result in different impedances and electrical lengths than were originally specified. This is because the *analysis* is generally more accurate than the *synthesis*.

Exercise 6.1

Go to the *Parts* window, move the cursor into a `tline` description, and hit the = key. If a valid part, the message box will reveal its layout dimensions. Change the part's specification to a `tline!` and again hit the = key. The message box now lists low frequency values for impedance and electrical length based on an accurate analysis of the dimensions. Note how the values differ from those in the part's description. Repeat this procedure with `clines`.

Exercise 6.2

Plot s_{11} and s_{21} for a single non 50Ω microstrip `tline` over several wavelengths. Change the `tline` to a `tline!` and re-plot by typing *Ctrl*-p from the *Plot* window. Note the differences caused by the addition of losses and dispersion.

Puff calculates the scattering parameters for distributed components using the formulas given in Fig. 6.1. The calculations require the physical length ℓ of the part, and its impedance(s) and propagation constant(s) (even and odd mode for coupled lines). The complex propagation constant γ is given by

$$\gamma = \alpha + j\beta \tag{6.1}$$

where α is the attenuation factor and β is the (real) propagation constant. The latter is given by

$$\beta = \sqrt{\epsilon_{re}(\omega)}k_0 \tag{6.2}$$

where $\epsilon_{re}(\omega)$ is the effective dielectric constant and k_0 is the TEM propagation constant $k_0 = \omega\sqrt{\epsilon_0\mu_0}$ for the equivalent transmission line with an air dielectric. For the ideal `tline`, the propagation constants simplify to

$$\alpha = 0 \qquad \beta = \sqrt{\epsilon_{re}}k_0 \tag{6.3}$$

Part	s_{ij}

`tline`
`tline!`
`qline`
	$\dfrac{(z-y)\sinh\gamma\ell}{2\cosh\gamma\ell+(z+y)\sinh\gamma\ell}$, for $i=j$;
	$\dfrac{2}{2\cosh\gamma\ell+(z+y)\sinh\gamma\ell}$, for $i\neq j$.

`clines` `clines!`	$\dfrac{(z_e-y_e)\sinh\gamma_e\ell}{4\cosh\gamma_e\ell+2(z_e+y_e)\sinh\gamma_e\ell}+\dfrac{(z_o-y_o)\sinh\gamma_o\ell}{4\cosh\gamma_o\ell+2(z_o+y_o)\sinh\gamma_o\ell}$, for $i=j$; $\dfrac{(z_e-y_e)\sinh\gamma_e\ell}{4\cosh\gamma_e\ell+2(z_e+y_e)\sinh\gamma_e\ell}-\dfrac{(z_o-y_o)\sinh\gamma_o\ell}{4\cosh\gamma_o\ell+2(z_o+y_o)\sinh\gamma_o\ell}$, coupled port; $\dfrac{1}{2\cosh\gamma_e\ell+(z_e+y_e)\sinh\gamma_e\ell}+\dfrac{1}{2\cosh\gamma_o\ell+(z_o+y_o)\sinh\gamma_o\ell}$, through port; $\dfrac{1}{2\cosh\gamma_e\ell+(z_e+y_e)\sinh\gamma_e\ell}-\dfrac{1}{2\cosh\gamma_o\ell+(z_o+y_o)\sinh\gamma_o\ell}$, isolated port.

Figure 6.1 Expressions used to calculate the scattering parameters for distributed components: the single transmission line (`tline`), transmission line with finite Q (`qline`), and coupled transmission lines (`clines`). In the table, z is the normalized impedance, y is the normalized admittance, γ is the complex propagation constant $\gamma=\alpha+j\beta$, and ℓ is the physical length of the component. The subscripts e and o refer to even and odd modes of the `clines`. For the `clines`, the ports are labeled in a way that is appropriate for a directional coupler: the through port is on the same line as the input port, but at the opposite end. The isolated port is the port diagonally across from the input, and the coupled port is at the same end as the input, but on the other line.

where, for microstrip, ϵ_{re} is the frequency independent quasi-static value given by (3.4). For stripline, ϵ_{re} is simply the relative dielectric constant ϵ_r of the substrate. If a `tline` specification is made using an electrical length in degrees at `fd`, then *Puff* will simply frequency scale the product $\theta=\beta\ell$ when computing the scattering parameters. For the ideal `clines`, (6.3) still applies, and is used for both even and odd mode propagation constants.

When a `tline!` or `clines!` model is invoked, *Puff* will calculate α at every analysis frequency for both microstrip and stripline circuits. It will also calculate β at each frequency for microstrip circuits. Dispersion can be a strong effect in microstrip transmission lines due to their inhomogeneity. Typically, as frequency is increased, ϵ_{re} increases in a non-linear manner, approaching an asymptotic value. For single and coupled microstrip trans-

mission lines, ϵ_{re} is modeled using Getsinger's expression [1,2]

$$\epsilon_{re_i}(\omega) = \epsilon_r - \frac{\epsilon_r - \epsilon_{re_i}(0)}{1 + F_i(\omega)} \tag{6.4}$$

where ϵ_r is the relative dielectric constant of the substrate material, $\epsilon_{re_i}(0)$ is the low frequency quasi-static value, and $F_i(\omega)$ is an increasing function of frequency. The subscript i is used to distinguish between the functions and values used for single microstrip, and for even and odd modes in coupled microstrip. Accurate closed form expressions for $F_i(\omega)$ are complicated. *Puff* uses the expressions derived for single and coupled microstrip transmission lines by Kirschning and Jansen [3,4]. Dispersion affects characteristic impedances in a similar way. It is modeled using Bianco's expression [5]

$$Z_{0_i}(\omega) = Z^s_{0_i} - \frac{Z^s_{0_i} - Z_{0_i}(0)}{1 + F_i(\omega)} \tag{6.5}$$

where $Z^s_{0_i}$ is twice the characteristic impedance of an equivalent (single or coupled) stripline with twice the thickness of the microstrip, and $Z_{0_i}(0)$ is the quasi-static impedance. *Puff* uses the same $F_i(\omega)$ functions for both (6.5) and (6.4).

Calculation of attenuation factors for the `tline!` and `clines!` includes contributions from a dielectric attenuation factor α_d and a conductor attenuation factor α_c where

$$\alpha = \alpha_c + \alpha_d. \tag{6.6}$$

The calculations for conductor attenuation have the form given by

$$\alpha^i_c = \frac{R_s}{Z_i W_t} K_i F_{sr} \tag{6.7}$$

where Z_i is the line impedance (even or odd mode for couplers), W_t is the effective width of the line (taking into account finite strip thickness), and K_i is the current distribution factor. The subscript and superscript i is again used to distinguish between the single line, and even and odd mode values for the coupled line. The surface resistance R_s is given by

$$R_s = \sqrt{\frac{\pi f \mu_0}{\sigma}} \tag{6.8}$$

at frequency f, free-space permeability μ_0, and conductivity σ. The substrate surface roughness factor F_{sr} has been evaluated by Hammerstad and

Bekkadal [6]. They find it well approximated by the expression

$$F_{sr} = 1 + \frac{2}{\pi} \arctan \left\{ 1.4 \left(\frac{\Delta}{\delta_s} \right)^2 \right\} \tag{6.9}$$

where Δ is the rms surface roughness and δ_s is the skin depth at the operating frequency. This factor is necessary to account for an asymptotic increase seen in the apparent surface resistance with decreasing skin depth. The expressions used for the current distribution factor K_i and effective width W_i have been given by Pucel, et al. [7] and Gupta, et al. [8] for single microstrip. Expressions for coupled microstrip have been given by Garg and Bahl [9] and Hammerstad and Jensen [10], while expressions for single and coupled stripline have been given by Gupta, et al. [11]. In most cases, K_i is derived from an application of Wheeler's incremental inductance rule [12].

Dielectric loss is due to the effects of finite loss tangent, $\tan\delta$. For the inhomogeneous line, an effective dielectric filling fraction gives that proportion of the transmission line's cross section not filled by air. For microstrip lines, the result is [13,14]

$$\alpha_d^i = \frac{\pi \epsilon_r}{\epsilon_r - 1} \frac{\epsilon_{re_i} - 1}{\sqrt{\epsilon_{re_i}}} \frac{\tan\delta}{\lambda_0} \tag{6.10}$$

where again i has been used to distinguish between values for single and coupled lines. For homogeneous stripline transmission lines, (6.10) simplifies to

$$\alpha_d = \frac{\pi \sqrt{\epsilon_r} \tan\delta}{\lambda_0}. \tag{6.11}$$

The `tline!` is very useful when an accurate analysis of a microstrip or stripline transmission line is required. Often, however, one may only be interested in transmission line losses. For this occasion, *Puff* has a special transmission line model that is identical to the `tline`, but allows one to specify its loss in terms of quality factor, or Q. Called the `qline`, it is specified in the same way as the `tline`, but a value for Q may be included in addition to impedance and electrical length. The definition of the quality factor, or Q, for a transmission line is the same as with other components. It is the well known figure of merit for the ratio of stored energy to dissipated energy. In terms of group velocity, it is written [15]

$$Q = \omega \frac{\text{energy per unit length}}{\text{power loss per unit length}} = \frac{\omega P / v_g}{P_L} = \frac{\omega}{2 v_g \alpha} \tag{6.12}$$

where P_L is the power loss per meter, and power flow P equals the product of the energy density and the energy transport (group) velocity v_g. In the

absence of dispersion, Q becomes the widely used expression

$$Q = \frac{\beta}{2\alpha}. \tag{6.13}$$

In a line dominated by dielectric loss, both β and α will increase proportionately over frequency, and the Q is therefore constant. For the line dominated with conductor loss, both α and Q have $\sqrt{f}$ behavior. When using the `qline`, a conductor or dielectric loss model may be specified. A transmission line dominated by conductor loss with $Q = 100$, is specified as

```
qline 50Ω 90° 100Qc
```

The same line dominated by dielectric loss is specified as

```
qline 50Ω 90° 100Qd
```

Puff computes β for the `qline` using the same quasi-static formulas used for the `tline`. It then computes α at design frequency `fd` using the specified Q value and (6.13). At frequencies other than `fd`, *Puff* scales the α according to either the dielectric or conductor loss model.

Exercise 6.3
Edit a `.puf` file to include values for metal thickness, surface roughness, loss tangent, and conductivity for your favorite microwave substrate. In the *Layout* window, connect a `tline!` between ports 1 and 2, and a comparable `qline` between ports 3 and 4. While comparing dB plots for s_{21} and s_{43} over many wavelengths, find the best Q value to model the losses seen in the `tline!` Is it better to use the `Qc` or `Qd` model?

Exercise 6.4
Design a three section `clines` bandpass filter and plot its frequency response. For a simple dry-lab, change the `clines` to `clines!` and re-plot using *Ctrl*-p. Can you account for the differences?

References

[1] W.J. Getsinger, "Microstrip dispersion model," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-21, pp. 34-39, Jan. 1973.

[2] W.J. Getsinger, "Dispersion of parallel-coupled microstrip," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-21, pp. 144-145, Mar. 1973.

[3] M. Kirschning and R.H. Jansen, "Accurate model for effective dielectric constant of microstrip with validity up to millimeter wave frequencies," *Electron. Lett.*, vol. 18, pp. 272–273, Mar. 1982.

[4] M. Kirschning and R.H. Jansen, "Accurate wide-range design equations for the frequency dependent characteristics of parallel coupled microstrip lines," *IEEE*

Trans. Microwave Theory Tech., vol. MTT-32, pp. 83-90, Jan. 1984. Corrections: *IEEE Trans. Microwave Theory Tech.*, vol. MTT-33, p. 288, Mar. 1985.

[5] B. Bianco, A. Chiabrera, M. Granara, and S. Ridella, "Frequency dependence of microstrip parameters," *Alta Freq.*, vol. 43, pp. 413-416, July 1974.

[6] E.O. Hammerstad and F. Bekkadal, *A Microstrip Handbook*, ELAB Report, STF 44 A74169, N7034, University of Trondheim, Norway, 1975.

[7] R.A. Pucel, D.J. Masse, and C.P. Hartwig, "Losses in microstrip," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-16, pp. 342–350, June 1968.

[8] K.C. Gupta, R. Garg, and I.J. Bahl, *Microstrip Lines and Slotlines*, Artech House, Dedham, Mass., 1979.

[9] R. Garg and I.J. Bahl, "Characteristics of coupled microstriplines," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-27, pp. 700-705, July 1979.

[10] E. Hammerstad and O. Jensen, "Accurate models for microstrip computer-aided design," *IEEE MTT-S Int. Microwave Symp. Dig.* (Washington, D.C.), pp. 407-409, 1980.

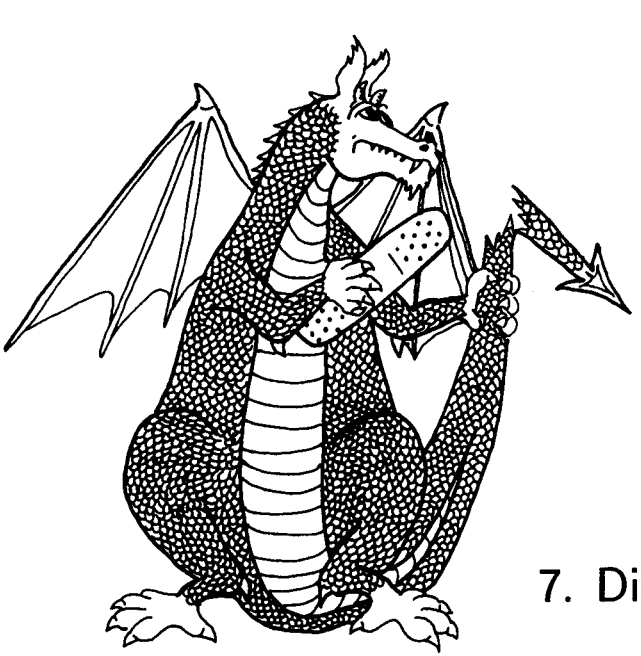
[11] K.C. Gupta, R. Garg, and R. Chadha, *Computer-Aided Design of Microwave Circuits*, Artech House, Dedham, Mass., 1981.

[12] H.A. Wheeler, "Formulas for the skin effect," *Proc. IRE*, vol. 30, pp. 412-424, Sept. 1942.

[13] M.V. Schneider, "Dielectric loss in integrated microwave circuits," *Bell System Technical Journal*, vol. 48, pp. 2325-2332, Sept. 1969.

[14] B. Ramo Rao, "Effect of loss and frequency dispersion on the performance of microstrip directional couplers and coupled line filters," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-22, pp.747-750, July 1974.

[15] R.E. Collin, *Field Theory of Guided Waves*, IEEE Press, New York, 1991.



7. Discontinuities

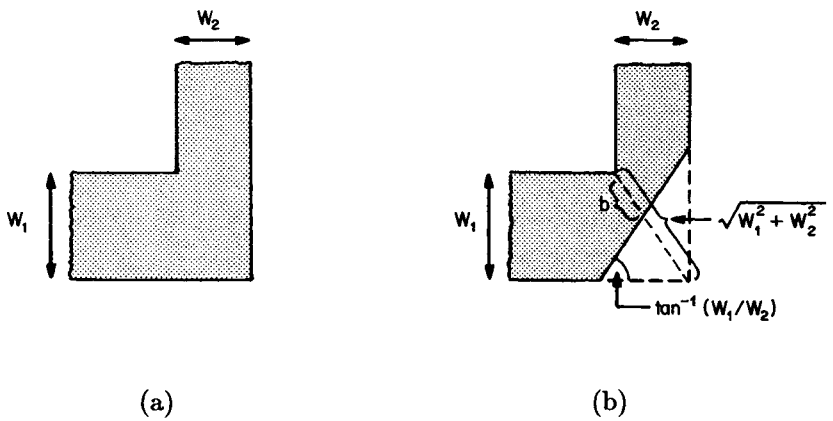
IN THIS CHAPTER the four dominant discontinuity effects in microstrip shall be considered: the excess capacitance of a corner, the capacitive end effect for an open circuit, the step change in width, and the length correction for the shunt arm of a tee. *Puff* will automatically miter corners in the artwork to reduce their effect. However, you must compensate for the other discontinuities yourself; they are neglected in the *Puff* analysis.

When a sharp right-angle bend occurs in a circuit (Fig. 7.1a), there will be a large reflection from the corner capacitance. *Puff* miters corners to reduce the capacitance and minimize this reflection, as shown in Fig. 7.1b. You can change the value of the miter fraction `m` defined to be

$$\mathtt{m} = 1 - b/\sqrt{W_1^2 + W_2^2}. \tag{7.1}$$

When the two line widths are equal, this formula reduces to the conventional definition [1]. In `setup.puf`, `m` is set to 0.6.

In an open-circuit, the electric fields extend beyond the end of the line. This excess capacitance makes the electrical length longer than the nominal length, typically by a third to a half of the substrate thickness. This will cause the design frequencies of patch antennas and filters to be shifted. To compensate for this effect in the artwork, a negative length correction can be



(a) (b)

Figure 7.1 Mitering a right-angle bend.

added to the parts list. Hammerstad and Bekkadal give an empirical formula for the length extension l in microstrip [2],

$$\frac{l}{h} = 0.412 \left(\frac{\epsilon_{re} + 0.3}{\epsilon_{re} - 0.258} \right) \left(\frac{W/h + 0.262}{W/h + 0.813} \right), \tag{7.2}$$

where h is the thickness of the substrate. This formula is plotted in Fig. 7.2 for several different dielectric constants.

A similar method may be used to compensate for a step change in width between high and low impedance lines. The discontinuity capacitance at the end of the low impedance line will have the effect of increasing its electrical length. Assuming the wider low impedance line has width W_2, and the narrower high impedance line has width W_1, compensate using the expression [1]

$$\frac{l_s}{h} \approx \frac{l}{h} \left(1 - \frac{W_1}{W_2} \right) \tag{7.3}$$

where l_s is the step length correction for line W_2, and l/h is the value obtained from (7.2) and Fig. 7.2.

In the tee, shown in Fig. 7.3, the electrical length of the shunt arm is shortened by distance d_2. The currents effectively take a short cut, passing close to the corner. It is particularly noticeable in the branch-line coupler because there are four tees. Hammerstad and Bekkadal give an empirical

formula for d_2 in microstrip [2],

$$\frac{d_2}{h} = \frac{120\pi}{Z_1\sqrt{\epsilon_{re}^1}}\left(0.5 - 0.16\frac{Z_1}{Z_2}\left[1 - 2\ln(Z_1/Z_2)\right]\right), \tag{7.4}$$

where ϵ_{re}^1 is the effective dielectric constant of the through arm. Equation (7.4) is plotted in Fig. 7.3 for a 50-Ω through line. This correction can also be used as a first estimate for compensating a four-way cross. For additional help on discontinuity modeling for both microstrip and stripline, see Gupta, et al. [3].

References

[1] T. C. Edwards, *Foundations for Microstrip Engineering*, John Wiley, New York, 1981.

[2] E. O. Hammerstad and F. Bekkadal, *A Microstrip Handbook*, ELAB Report, STF 44 A74169, N7034, University of Trondheim, Norway, 1975.

[3] K.C. Gupta, R. Garg, and R. Chadha, *Computer-Aided Design of Microwave Circuits*, Artech House, Dedham, Mass., 1981.

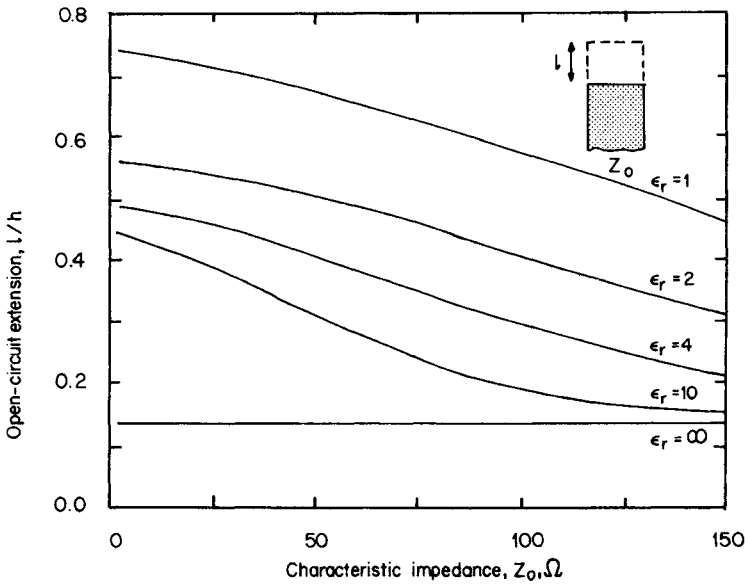


Figure 7.2 The open-circuit end correction in microstrip, plotted from (7.2). The artwork length correction in a parts list should be *negative*.

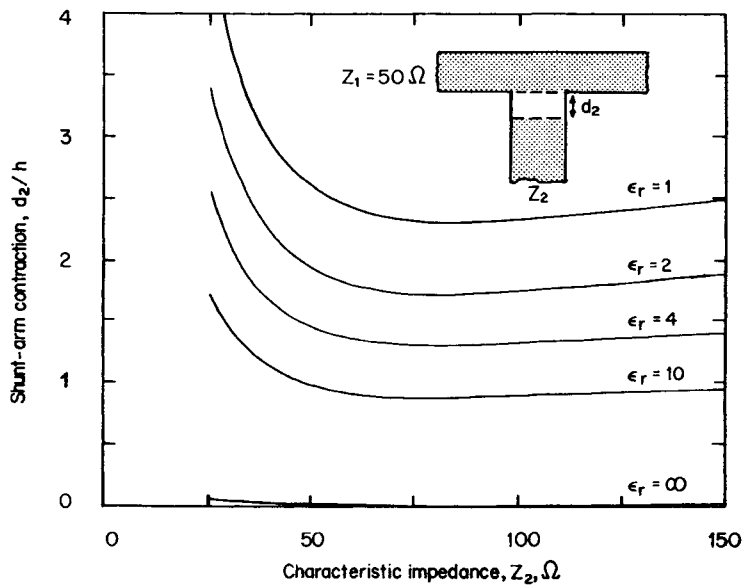
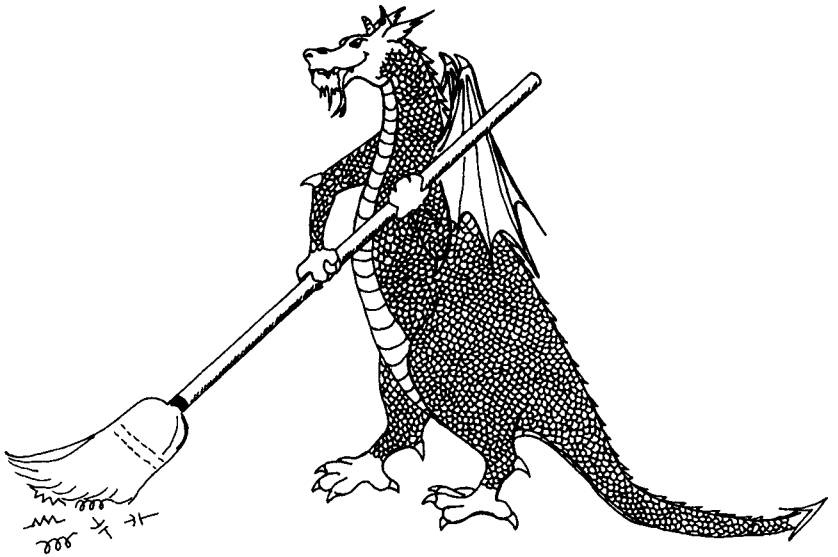


Figure 7.3 The shortening of the shunt arm of a microstrip tee, given by (7.4). The length correction for the shunt arm should be *positive*.



8. The Component Sweep

THE SIMPLE OPTIMIZER included in *Puff* is called the *component sweep.* Instead of sweeping with respect to frequency, a circuit's scattering coefficients may be swept with respect to a changing component parameter. This feature is invoked by placing a question mark (?) in front of the parameter to be swept in the appropriate position of a part's description. For example, to find the optimum value for a tuning capacitor, one could specify a part as `lumped ?5pF`. When plotting in the *Plot* window, the frequency will then be held constant at the design frequency `fd`, and the values specified in the x-axis of the rectangular plot will be substituted for (in the above example) capacitance values in picofarads. In this manner, any parameter used in the parts list may be designated as a sweep parameter, but only one parameter may be swept at a time. Swept lumped elements are restricted to single resistors, capacitors or inductors. A description such as `lumped ?1+5j-5j`Ω is not allowed since it is a series RLC circuit. In addition, the parallel sign || may not be present in the lumped specification. The unit and prefix given in the part's description (following the ?) is inherited by the component sweep.

As an example, we will use the component sweep to make a stub matching circuit for the Fujitsu FHX04 HEMT. We begin with the circuit of Fig. 8.1. The electrical length (in degrees) of the `tline` at part `b` has been swept from 0° to 100°. At 62°, the input impedance of the circuit is seen to cross the

upper half of the $g = 1$ normalized conductance circle (use the **Tab** key to toggle between impedance and admittance circles). This means we can then match with an open circuit stub. A stub has been added in Fig. 8.2. The length of the `tline` at part `b` is now held constant, and we sweep the electrical length of the stub at part `c`. With increasing stub length, the impedance slides down the $g = 1$ circle and becomes well matched at 70°. We thus have the matching circuit for operation at `fd`. Granted, this is a narrow bandwidth match, but this process can be repeated with more components and frequency domain checking to create a broader bandwidth circuit.

Be aware that values are easily encountered in the component sweep that cannot be physically realized. *Puff* must change the swept part to Manhattan dimensions to avoid problems. When specifying `tline` lengths in other than degrees, or when sweeping `clines`, this may cause the alternate parameter sweep to yield s-parameter values different from the standard *Puff* analysis. This occurs because electrical lengths are also a function of impedance via effective dielectric constant calculations. To reduce this disparity, specify electrical lengths in degrees whenever possible. When sweeping a `tline` or `clines` impedance, include a best guess value. For example, the specification `tline ?Ω 5mm` for a microstrip circuit is a poor one, because for *Puff* to compute the electrical length of a 5 mm length of line, it needs a W/h value from the impedance to compute the effective dielectric constant. When *Puff* encounters such a specification, it will compute an effective dielectric constant based on a characteristic impedance of `zd`. If it was known that the impedance would be near 70Ω, a better specification would be `tline ?70Ω 5mm`. Then *Puff* would use an effective dielectric constant calculation based on a 70Ω impedance and this value would be used for the impedance sweep.

Similar problems arise for `clines`. The description `clines ?Ω 90°` is ambiguous, because *Puff* cannot determine if the value to be swept is an even or odd mode impedance. In addition, `clines` require the calculation of both even and odd mode effective dielectric constants; the electrical length specified is the average of the two. For this reason it is always better to first enter a best guess component designation, and then add the question mark. If, for example, one wanted to sweep the even mode impedance with a 40Ω odd mode impedance, then by entering

```
clines ?60Ω 40Ω 90°
```

Puff has enough information to calculate the propagation constants, and can recognize that the even mode impedance is to be swept. Whenever possible, enter a valid best guess component specification, then add the question mark.

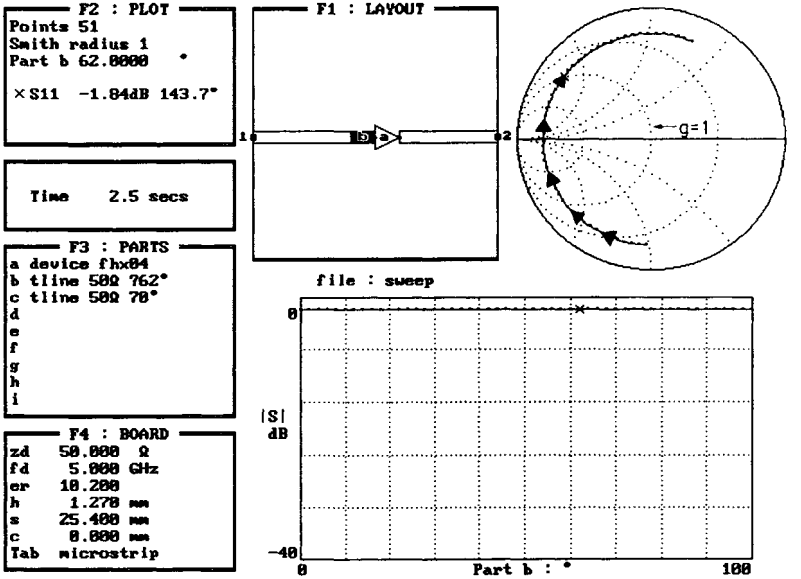


Figure 8.1 The `tline` at part `b` is swept from 0° to 100° while monitoring the input match of the Fujitsu FHX04 HEMT. At 62°, the locus intercepts the $g = 1$ normalized conductance circle.

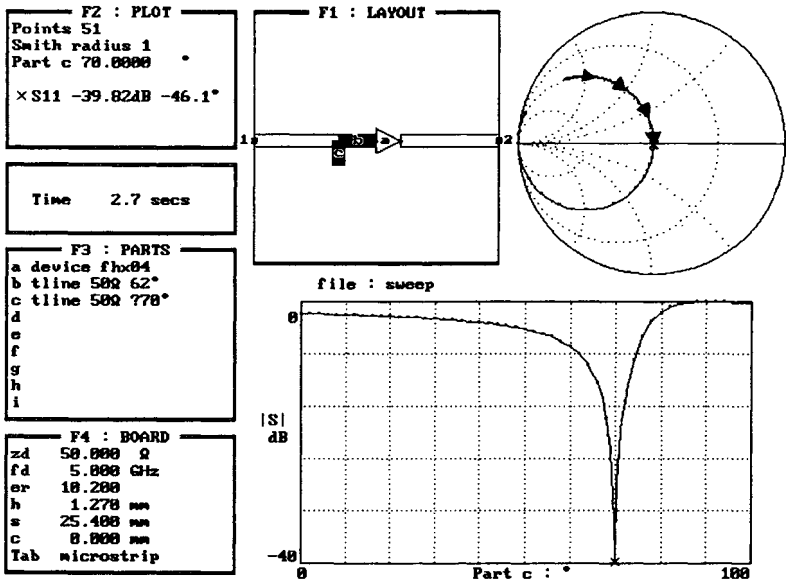
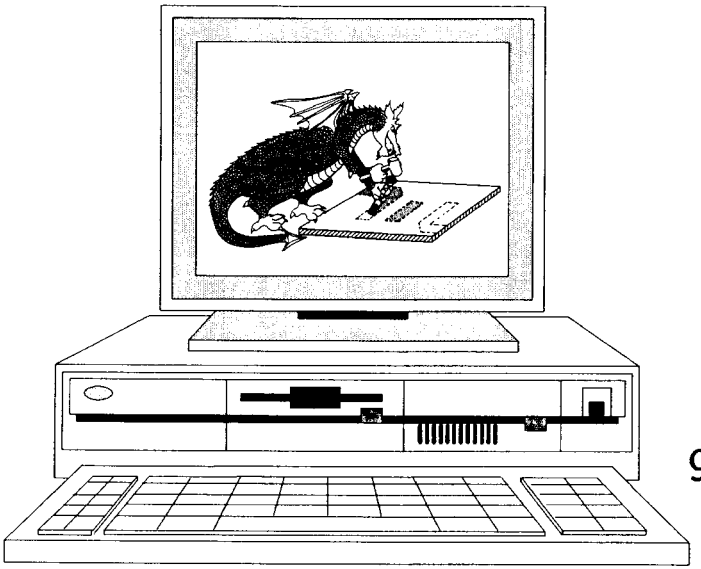


Figure 8.2 A stub is added at part `c` and swept from 0° to 100°. The locus slides down the $g = 1$ circle, and becomes well matched at 70°.



9. Using Puff

THE CALTECH MICROWAVE course has a weekly laboratory in which students use *Puff* to design, fabricate, and measure microwave integrated circuits [1]. Over the duration of the course, each student generates a minimum of eight different circuits. Six examples are given in Fig. 9.1. Quarter-wave low-pass filters (as in Fig. 9.1a) are used to study open-circuit end effects in low impedance transmission lines, periodicity in the frequency domain, and control of filter passband ripple. Students build either a branch-line (Fig. 9.1b) or rat-race 3 dB directional coupler to examine the properties of symmetrical four-port structures and to investigate the effects of tee discontinuities. A bandpass filter (Fig. 9.1c) is employed to study coupled transmission lines. Next, the optimum input and output matching circuits are generated for a GaAs FET low-noise amplifier (Fig. 9.1d), followed by the design of a broadband amplifier using resistive feedback. A GaAs FET is also used to build a microwave oscillator using either coupled-line feedback (Fig. 9.1e) or a common gate circuit. Students also use *Puff* to build patch antennas (Fig. 9.1f) for class projects [2].

The microwave lab procedure is streamlined so that it can be completed in a single two-hour period. Students begin by completing their circuit designs and then use *Puff* to produce the photographic artwork on an HP LaserJet printer. The artwork is then photographically reduced onto 2.5″ square glass

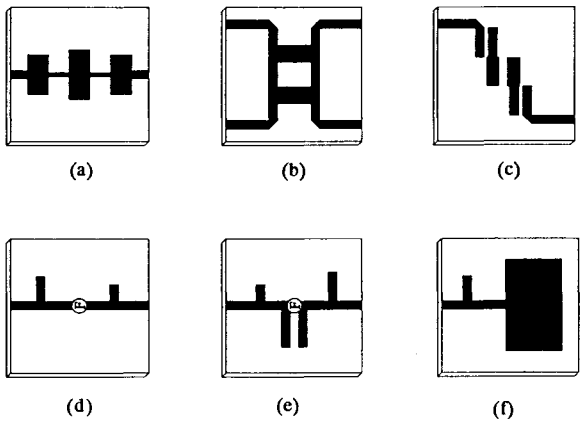


Figure 9.1 Example microstrip circuits designed with *Puff*: (a) a five-section low-pass filter; (b) a branch-line coupler; (c) a coupled-line bandpass filter; (d) a reflective matched GaAs FET low-noise amplifier; (e) an oscillator with coupled-line feedback; (f) a patch antenna with matching circuit. The circuits are typically fabricated on 0.635-mm (25-mil) and 1.27-mm (50-mil) *RT/duroid* substrates with $\epsilon_r = 10.2$ and one-ounce copper cladding.

emulsion plates. These are developed and used as masks to expose photoresist covered, copper clad *RT/duroid* substrates. The etching process leaves dimensions typically accurate to 25 μm. An alternative lower-cost fabrication procedure is also possible [3]. When fabrication is complete, the circuits are placed in a brass test fixture, as shown in Fig. 9.2. Scattering parameter measurements are made with an HP-8720 microwave network analyzer. A personal computer controls the process, placing the measured s-parameters into a device file readable by *Puff*. The students then compare *Puff*'s predictions alongside the measured data. Correlation between the two is usually not perfect, so the students must determine where the analysis goes wrong. *Puff* does not automatically compensate for discontinuities, and it is a good puzzle for the students to adjust their circuit models to include their effects.

We shall now look at several designs and compare *Puff* predictions with measurements. Discontinuities have been taken into account using the length corrections described in chapter 7. Figure 9.3a shows a low-pass filter with alternating sections of high and low impedance lines. This *Puff* analysis was performed using `tline` sections with length corrections included for the end-capacitance of the low impedance lines. Fig. 9.3b is a bandpass filter with three quarter-wave coupled line sections. This *Puff* analysis was performed using `clines!` to model the losses and dispersion. Each open circuit of the

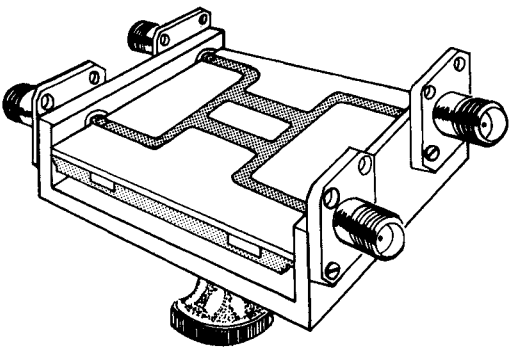


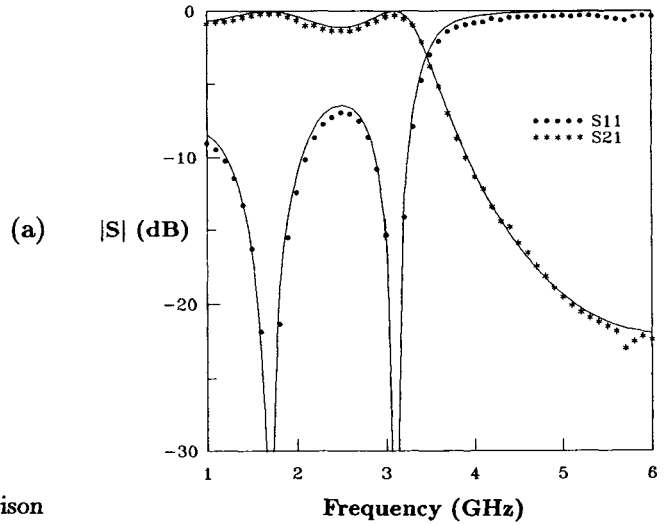
Figure 9.2 Brass test fixture for microstrip circuits. The plate underneath the circuit flexes and pushes against the sides to make a ground connection. The thumbscrew on the bottom locks the plate into place. The VSWR of the coax-to-microstrip transition is typically 1.1 or less for frequencies less than 6 GHz.

coupled lines was modeled using `tlines` with widths equal to the coupled line branches, and lengths equal to half the open-circuit end correction.

Figure 9.4 shows the design of a single-stage low-noise amplifier using a Fujitsu FSC10 field effect transistor. The gain of the amplifier was measured using an HP-8970A noise-figure meter, and the transistor bias was applied using two external bias tees. A maximum gain of 13.2 dB was obtained at 3.7 GHz with a noise figure of 1.9 dB.

We use an approach to oscillator design that differs from the traditional. With *Puff*, it is natural to analyze the circuit looking in from the external ports, whereas a conventional design looks at conditions inside the circuit. The oscillation condition is $s_{11} = 1/s_L$, where s_{11} is the reflection coefficient at the input of the transistor circuit and s_L is the reflection coefficient of the load. Fig. 9.5a shows how this condition can be achieved. The s_{11} curve loops around the point $1/s_L$ (Fig. 9.5a). It is assumed that the s_{11} curve circles clockwise as the frequency increases and that as the transistor saturates, the loop contracts. Oscillations build up in the circuit, the transistor saturates, and the s_{11} loop shrinks until the oscillation condition is met.

However, an interesting problem arises when we try to follow this procedure with *Puff*. The load is the matched port, so that $1/s_L = \infty$. It is not clear how we go about drawing a clockwise loop about the point at infinity. The solution is to add circuit elements that produce a *counterclockwise* loop. Because of the properties of bilinear transforms, a region *inside* the previous clockwise loop is mapped to the *outside* of the counterclockwise loop. This means that when the transistor saturates the loop expands until it intersects the point at infinity, and the oscillation condition is satisfied (Fig. 9.5b). It is easy to demonstrate this effect by reducing the transistor gain to simulate



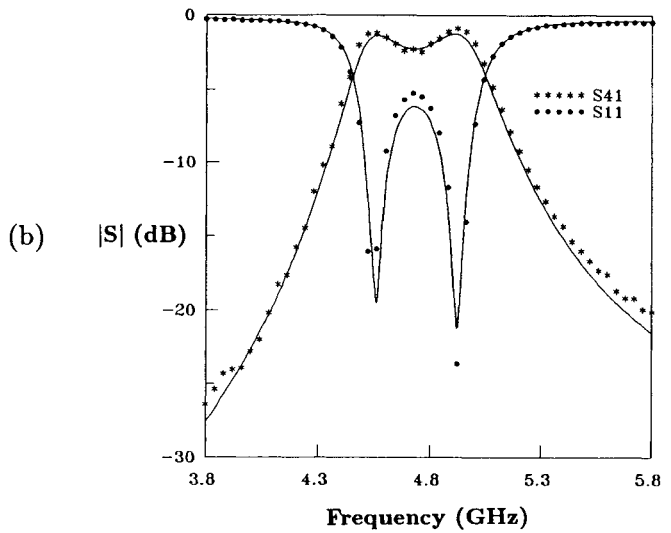


Figure 9.3 Comparison between *Puff*'s predictions (solid lines) and HP-8720 measurements (• and ∗) for a low-pass filter (a), and a bandpass filter (b).

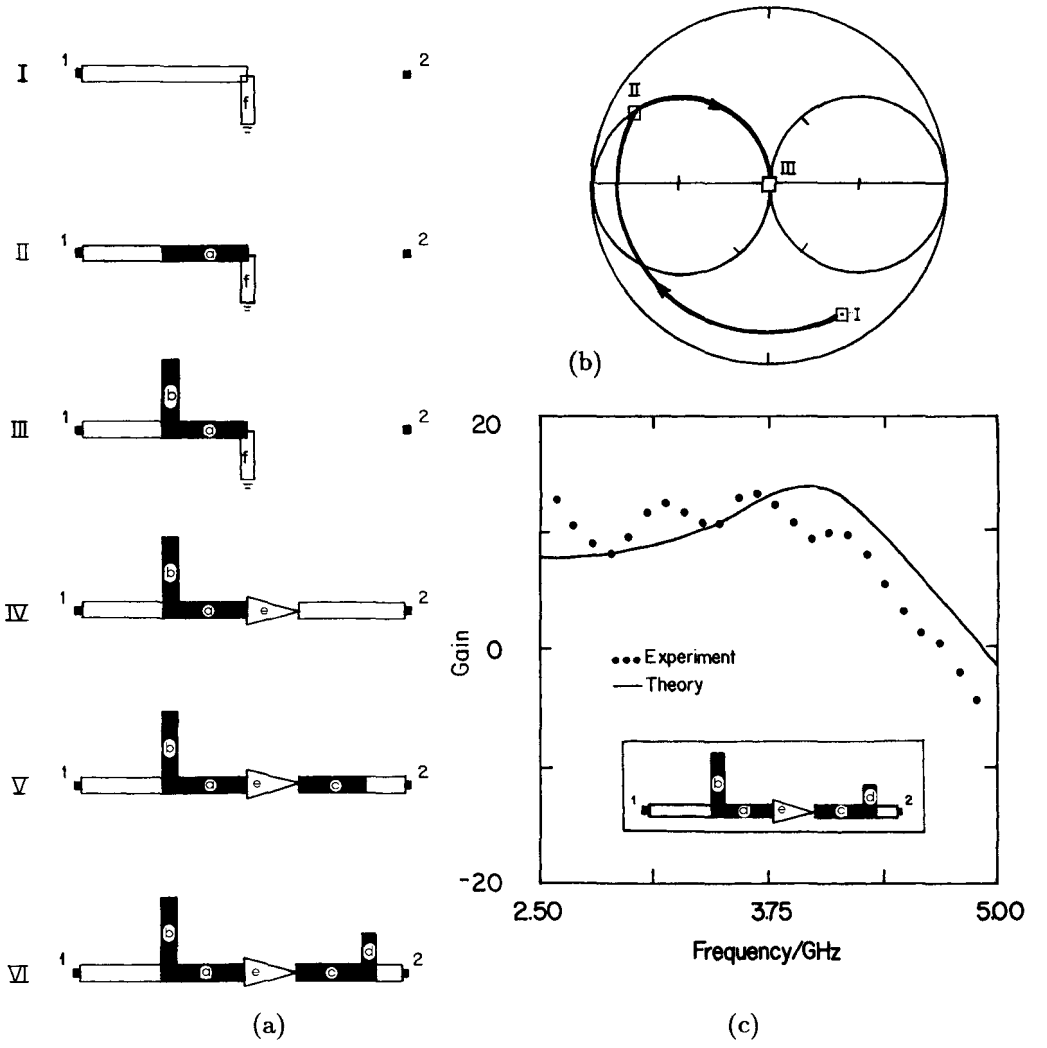


Figure 9.4 Designing a low-noise FET amplifier. Successive views of the *Layout* window are on the left (a), and the Smith chart (b) and the *Plot* window (c) are on the right. We start with a noise match of the input circuit in (b). In (I), `lumped f` is the conjugate of the optimum source impedance, given in the manufacturer's data sheet. In (II), `tline a` moves s_{11} around to the $g = 1$ circle and in (III) a stub, `tline b`, brings it into the center of the Smith chart. In a similar way, we power match the output circuit. In (IV–VI), `tline c` and `tline d` bring s_{22} into the center. The rectangular plot, (c), compares *Puff*'s predictions with gain measurements.

saturation. This design procedure has been used to build several oscillators.

The final example shows how to calculate the voltage waveforms in circuits that are not terminated in the normalizing impedance. On your disk are a voltmeter, `vmeter.dev`, and voltage source, `vsource.dev`. The voltmeter is connected with the wide end, port 1, at some point in the network. The other end is joined to a connector port. The voltmeter s_{11} is 1, an open, so that it does not load the network, and the s_{21} is 2, because the total voltage at an open circuit is twice the incident voltage. The voltage source is also connected with the wide end at some point on the circuit and the narrow end to another connector port. The source s_{11} is -1, a short, and the source s_{12} is 1. The source s_{22} is 1; this allows us to see the input waveform. With these devices in the circuit, we can interpret scattering coefficients as voltage-transfer parameters. Fig. 9.6 shows the step response at 50-Ω line section that is terminated with a 150-Ω resistor. This approach can be extended to make a wide variety of probes and sources, and to many different kinds of plots.

References

[1] S.W. Wedge and D.B. Rutledge, "Computer-aided design for microwave education," *Electrosoft*, vol. 2, pp. 13-20, Mar. 1991.

[2] R.A. York, R.C. Compton, M. Kim, S. Wedge, and D.B. Rutledge, "An interactive approach for designing microwave circuits using a personal computer," *IEEE AP-S Int. Symp. Dig.*, pp. 6-9, 1988.

[3] R.C. Compton and R.A. York, "A hands-on microwave laboratory course using microstrip circuits," *IEEE Trans. Educ.*, vol. E-33, pp. 161-163, Feb. 1990.

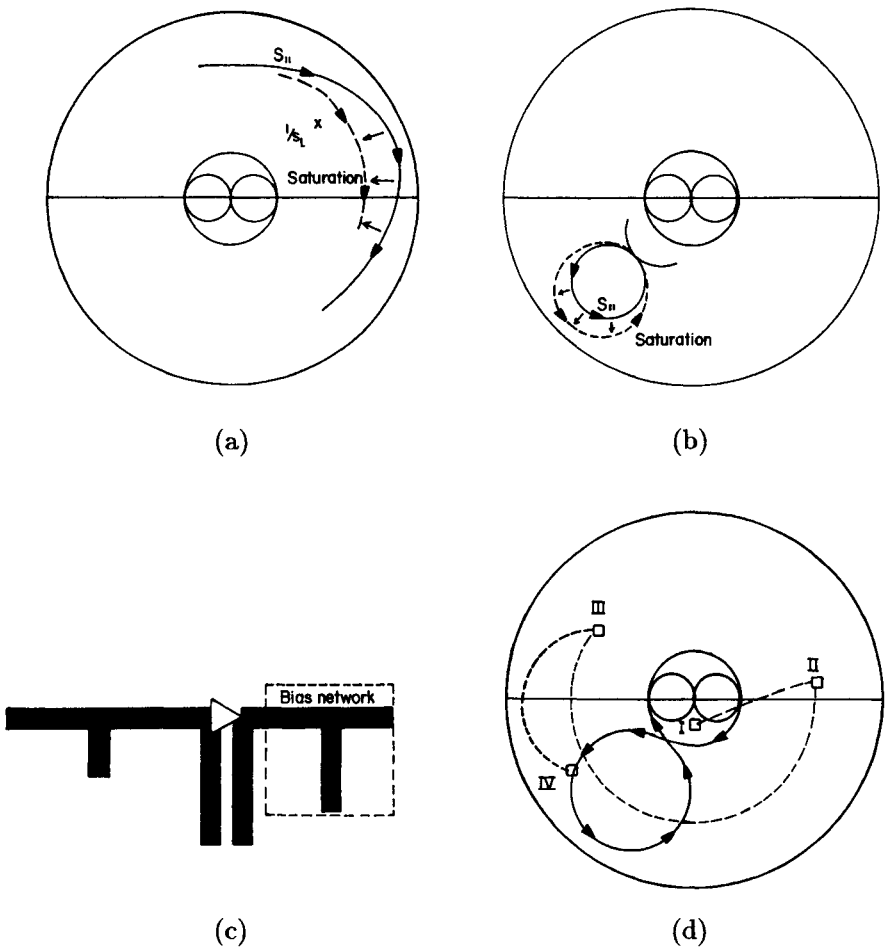


Figure 9.5 Expanded Smith chart of radius 4 for the FET oscillator design. A clockwise loop enclosing the point $1/s_L$ results in oscillations (a). A counterclockwise loop gives oscillations for a matched load, where $1/s_L = \infty$ (b). Layout for a microstrip FET oscillator (c). The coupled-line feedback loop increases $|s_{11}|$ to 3 (I–II) and a stub is positioned to produce a counterclockwise loop (III–IV) (d). This circuit generated 10 mW at 5.6 GHz.

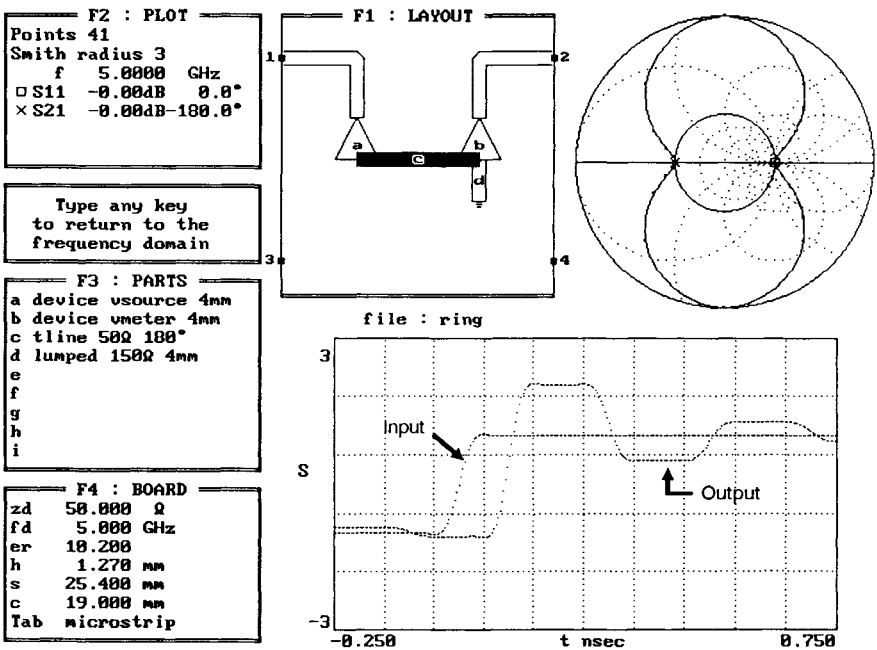
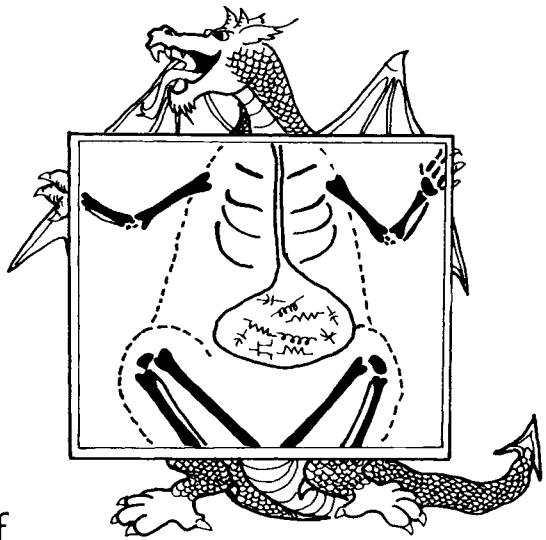


Figure 9.6 Step response for an ideal voltage source driving a 150-Ω load through a section of 50-Ω transmission line. The lower frequency limit is 0, and the upper limit is 20 GHz. Part a is the source, part b is the meter, s_{11} is the input, and s_{21} is the output. The ringing occurs because neither the source nor the load is matched.



10. Inside Puff

YOU MAY HAVE STUDIED how to analyze a circuit with an admittance or impedance matrix, and solved the matrix by Gaussian elimination. This approach has the advantage of being easy to program, but it is relatively slow for microwave circuit analysis. One problem is that microwave networks are characterized by scattering parameters. If we want to analyze a microwave circuit in terms of an admittance matrix, we must first convert the s-parameters of the components to admittance matrices, then find the admittance matrix of the entire circuit, and finally convert back to s-parameters. Usually these conversions take so much time that it is better to work with scattering coefficients throughout. We can also improve on the simple Gaussian elimination. A matrix that describes a complete microwave circuit is usually large. For example, a branch-line coupler requires the solution of a system of 20 linear equations with complex coefficients. The important thing to realize is that the matrix is sparse, that is to say, it has a large number of zeros. In the branch-line coupler, for example, about two-thirds of the 400 coefficients in the matrix are zero. In an ordinary Gaussian elimination, most of the time would be spent multiplying, adding, or storing zeros.

Puff computes scattering matrices using a fast and memory efficient algorithm called *subnetwork growth.* It is applicable to arbitrarily interconnected components, and involves dividing the network to be solved into subcircuits

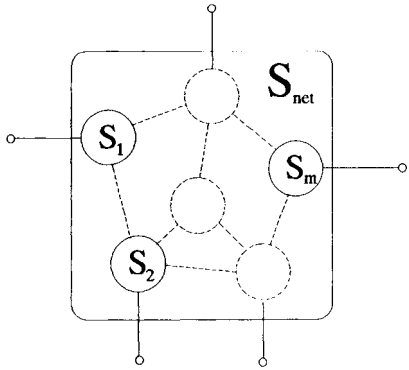


Figure 10.1 Schematic diagram of an aggregate network, $\mathbf{S_{net}}$, consisting of many interconnected components, each characterized by a scattering matrix. Solid lines depict external terminals; dashed lines depict internal connections.

to simplify the calculations. The approach was first described by Murray-Lasso [1] and applied to scattering matrices by Monaco and Tiberio [2]. The calculations use small and dense scattering matrices, and conversions to impedance or admittance matrices are unnecessary.

Puff must solve the network analysis problem illustrated in Fig. 10.1. The components, represented by scattering matrices $\mathbf{S_1}$, $\mathbf{S_2}$, ... $\mathbf{S_m}$, have been combined to form an aggregate network with unknown scattering matrix $\mathbf{S_{net}}$. The analysis begins with a calculation of the scattering matrices for each component that appears in the circuit. Chapter 6 describes the calculations for the transmission lines and coupled lines. The remaining components are calculated using the expressions given in Fig. 10.2. *Puff* keeps track of the connections present in the circuit and inserts open circuits, grounds, tee's, and crosses, as needed. The open circuit is a one-port that occurs when the end of a part is left hanging, unconnected to anything. The one-port ground connection appears when = is typed. Tees occur where three parts join at an ungrounded node, and crosses, where four parts meet.

Associated with each component scattering matrix $\mathbf{S_k}$ are input and scattered wave variable vectors $\mathbf{a_k}$ and $\mathbf{b_k}$, respectively. Organizing the scattering matrices for all components into block diagonal form results in the matrix equation

$$\begin{pmatrix} \mathbf{b_1} \\ \mathbf{b_2} \\ \vdots \\ \mathbf{b_m} \end{pmatrix} = \begin{pmatrix} \mathbf{S_1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{S_2} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{S_m} \end{pmatrix} \begin{pmatrix} \mathbf{a_1} \\ \mathbf{a_2} \\ \vdots \\ \mathbf{a_m} \end{pmatrix} \tag{10.1}$$

Apparent in Fig. 10.1, the aggregate network will have internal connections,

part	s_{ij}
open	1
ground	-1
tee	$-1/3$, for $i = j$; $2/3$, for $i \neq j$.
cross	$-1/2$, for $i = j$; $1/2$, for $i \neq j$.
`atten`	0, for $i = j$; $10^{-\alpha/20}$, for $i \neq j$.
`lumped`	$\frac{z}{z+2} = \frac{1}{1+2y}$, for $i = j$; $\frac{2}{z+2} = \frac{2y}{1+2y}$, for $i \neq j$.
`xformer`	$\pm\frac{n^2-1}{n^2+1}$, for $i = j$; $\frac{2n}{n^2+1}$, for $i \neq j$.
`device`	Values specified in the file; 0 if not specified.
`indef`	Values specified in the file; $\mathbf{S}_n \Rightarrow \mathbf{S}_{n+1}$.

Figure 10.2 Expressions used to calculate the scattering parameters for most *Puff* components. In the table, α is the attenuation in dB, z is the normalized impedance, y is the normalized admittance, and n is the transformer turns ratio. $\mathbf{S}_n \Rightarrow \mathbf{S}_{n+1}$ refers to the `indef`'s conversion of an n port scattering matrix into an $n+1$ port scattering matrix. Scattering parameters for the `tline`, `qline`, and the `clines` are calculated using the equations given in Fig. 6.1.

as well as external terminals. By separating the internal and external wave variables, a partitioned version of (10.1) is formed:

$$\begin{pmatrix} \mathbf{b_e} \\ \mathbf{b_i} \end{pmatrix} = \begin{pmatrix} \mathbf{S_{ee}} & \mathbf{S_{ei}} \\ \mathbf{S_{ie}} & \mathbf{S_{ii}} \end{pmatrix} \begin{pmatrix} \mathbf{a_e} \\ \mathbf{a_i} \end{pmatrix} \tag{10.2}$$

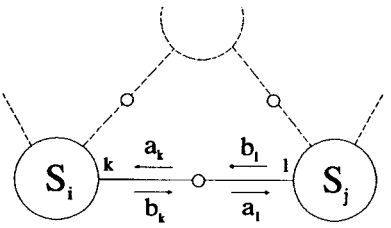


Figure 10.3 Detail showing an incident and scattered waves for connected components $\mathbf{S_i}$ and $\mathbf{S_j}$. The connection requires equalities $a_k = b_l$ and $a_l = b_k$.

where the external waves are denoted by subscript **e**, and those internal by subscript **i**. The internal connections of the network impose constraints on the wave variables. Namely, where a connection exists, there will be an equality established between incident and scattered waves. This is apparent upon examination of the single connection shown in Fig. 10.3. The waves in the figure must satisfy

$$a_k = b_l \qquad a_l = b_k. \tag{10.3}$$

These relations exist at every internal connection allowing the construction of a *connection matrix* Γ that satisfies

$$\mathbf{b_i} = \mathbf{\Gamma}\, \mathbf{a_i}. \tag{10.4}$$

Since Γ designates equality between internal incident and scattered waves, its elements are either 1 or 0 based on the circuit topology. This is sufficient information to calculate the scattering matrix for the connected network. With $\mathbf{S_{net}}$ defined as the ratio of external incident and scattered waves:

$$\mathbf{b_e} = \mathbf{S_{net}}\, \mathbf{a_e} \tag{10.5}$$

solving (10.2-10.5) yields the connection equation:

$$\mathbf{S_{net}} = \mathbf{S_{ee}} + \mathbf{S_{ei}}(\mathbf{\Gamma} - \mathbf{S_{ii}})^{-1}\mathbf{S_{ie}}. \tag{10.6}$$

A single application of connection equation (10.6) is all that is needed to carry out the scattering parameter analysis. Unfortunately, for a circuit with many components, the matrices involved become very large, and the computer time and memory required for the $(\Gamma - \mathbf{S_{ii}})$ matrix inversion are substantial.

These computational requirements are reduced with the *subnetwork growth* process. In subnetwork growth, a large circuit is analyzed by first calculating the s-parameters of smaller intermediate subnetworks [3]. The subnetworks are then connected to form the overall network. Many applications of connection equation (10.6) are required, but the smaller subnetworks require simpler matrix inversions. *Puff* builds up networks by making one connection at a time, effectively reducing the size of the $(\mathbf{\Gamma} - \mathbf{S_{ii}})$ matrix to a 2×2. The low order connection calculations continue until the complete network is formed.

As an example of the subnetwork growth process, consider how *Puff* analyzes the four-port branch-line coupler shown in Fig. 10.4a. *Puff* interprets the coupler to be a network of eight parts: two quarter-wave 50-Ω transmission lines, two quarter-wave 35-Ω lines, and four tees. These components are labeled A through H in the figure, and the external ports are numbered from 1 to 4. In the first step, the parts are joined in pairs (Fig. 10.4b) to make four new three-ports shown in Fig. 10.4c. These are combined by pairs to form two four-ports, $ABCD$ and $EFGH$, as shown in Fig. 10.4d. *Puff* makes a connection between the four-ports to create the six-port shown in Fig. 10.4e. Two remaining terminals of the six-port are connected to complete the analysis.

When making one connection at a time, matrix equation (10.6) is equivalent to applying two distinct connection operations. These depend on whether the ports to be joined are on different networks or on the same network (Fig. 10.5). We can apply Mason's theory of signal flow-graphs [4] to calculate the s-parameters for each case. Fig. 10.6a shows the signal flow graphs when the two ports are on different networks. We call the original pair of networks S and T, and the resulting single network S'. The ports to be joined are k and l. We let the input port be j, and consider an output port i, in S, and another output port m, in T. Using Mason's rule, we can write down two formulas for the scattering coefficients:

$$s'_{ij} = s_{ij} + \frac{s_{kj} t_{ll} s_{ik}}{1 - s_{kk} t_{ll}} \tag{10.7}$$

and

$$s'_{mj} = \frac{s_{kj} t_{ml}}{1 - s_{kk} t_{ll}}. \tag{10.8}$$

The signal-flow graph for the more complicated internal connection is shown in Fig. 10.6b. Mason's rule has to be applied carefully, because there are three first-order loops and a second-order loop. The result is

$$s'_{ij} = s_{ij} + \frac{s_{kj} s_{il}(1 - s_{lk}) + s_{lj} s_{ik}(1 - s_{kl}) + s_{kj} s_{ll} s_{ik} + s_{lj} s_{kk} s_{il}}{(1 - s_{kl})(1 - s_{lk}) - s_{kk} s_{ll}}. \tag{10.9}$$

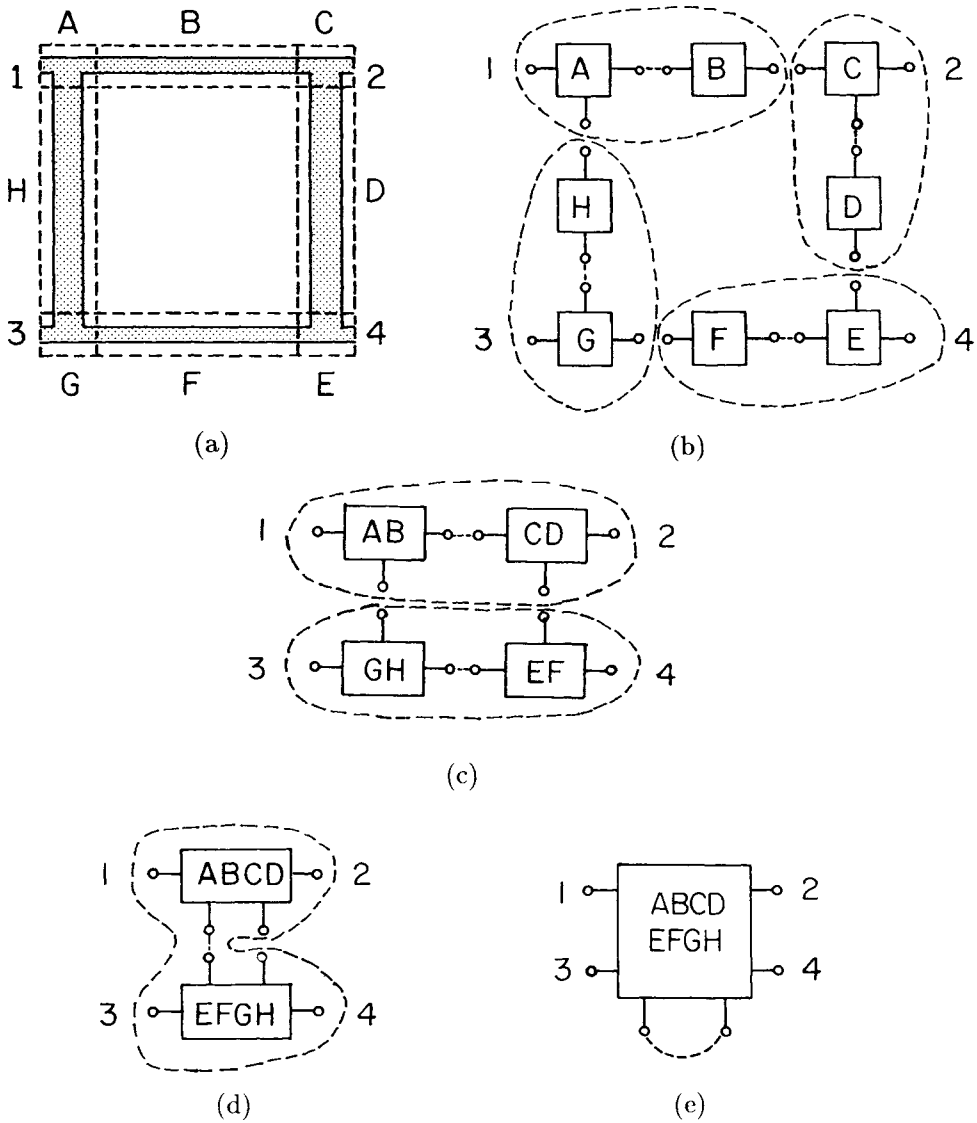


Figure 10.4 Analysis of a branch-line coupler. Sketch of the coupler, showing the different parts (a). Representing the eight parts by unconnected boxes (b). Joining the eight parts by pairs to make four three-ports (c), and joining these to make two four-ports (d). Then one connection is made to a single six-port. Finally the last two internal ports are connected (e).

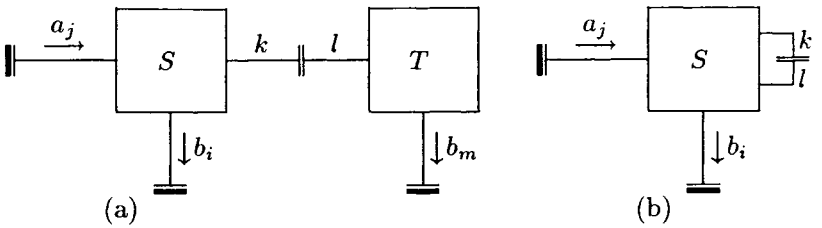


Figure 10.5 The two kinds of joints. A joint between ports k and l on different networks (a), and on the same network (b).

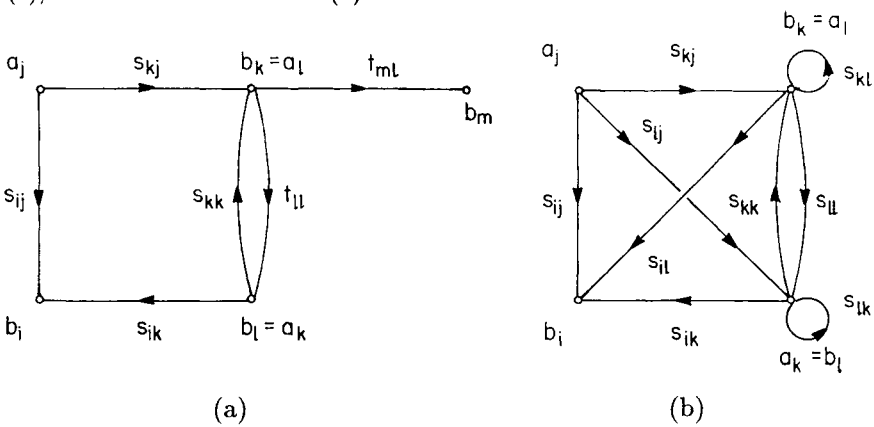


Figure 10.6 The signal flow graphs for joining ports k and l on different networks S and T (a), and on the same network (b).

The execution time of this algorithm is sensitive to the order in which the joins are made. How do we choose the next pair of ports to combine? We do not know of any proofs that answer this question, but it is easy to see by experimenting that it does make a difference. *Puff* uses a simple approach that searches for a connection that will result in the new subnetwork with the smallest number of s-parameters to calculate [5].

There are two classes of singularities that must be avoided when calculating the s-parameters. One type occurs when the s-parameters become infinite. For example, the reflection coefficient of a resistor connected to ground with a resistance *exactly* equal to $-$`zd` is infinite. To avoid this problem, *Puff* multiplies all negative resistances by a factor exceedingly close to one: $1 - (1.235 \times 10^{-12})$. This is equivalent to adding a tiny resistor, which does not affect ordinary calculations. This permits the scattering parameters for the singular resistor, although large, to be carried through further calculations

without overflows.

The other singularity occurs when a denominator in (10.7-10.9) goes to zero, but the numerator also goes to zero at the same time, in such a way that the scattering parameter remains finite. This is associated with resonant loops in specific circuits involving tees, crosses, opens, and shorts. *Puff* avoids this singularity by multiplying the scattering parameters for these parts by the same factor in the previous paragraph. Physically, this is equivalent to adding tiny attenuators at each port, and it is numerically equivalent to applying l'Hôpital's rule.

Puff is written in *Turbo Pascal* and stores circuits in dynamic variables that are addressed with pointers. The process of laying-out a network consists of building up linked lists of subnetworks and connections. In the analysis, *Puff* collapses these linked lists by applying the reduction formulas (10.7-10.9) until every connection is made. Computation time is reduced by using routines which exploit the power of the math coprocessor. If a coprocessor is not present, it is emulated.

References

[1] M.A. Murray-Lasso, "Black-box models for linear integrated circuits," *IEEE Trans. Educ.*, vol. E-12, pp. 170-180, Sept. 1969.

[2] V. A. Monaco and P. Tiberio, "Automatic scattering matrix computation of microwave circuits," *Alta Freq.*, vol. 39, pp. 59-64, Feb. 1970.

[3] G. Filipsson, "A new general computer algorithm for S–matrix calculation of interconnected multiports," *Proc. 11th Euro. Microwave Conf.*, pp. 700-704, 1981.

[4] S. J. Mason, "Feedback theory–further properties of signal flow graphs," *Proc. IRE*, vol. 44, pp. 920-926, July 1956.

[5] V. A. Monaco and P. Tiberio, "Computer-aided analysis of microwave circuits," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-22, pp. 249-263, Mar. 1974.

Index

Disclaimer of Warranty

Puff is distributed at-cost by Caltech as a public service. It is provided "as is" without warranty of any kind, express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to the software and the accompanying written materials. The entire risk as to the quality and performance of *Puff* will lie with the user. In no event shall Caltech, the authors, or any other person, institution, or establishment be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use *Puff*, regardless of the possibility of such damages. There is no guarantee that the operation of the program will be uninterrupted or error free.

PUFF

Computer Aided Design for Microwave Integrated Circuits

Version 2.0

Scott W. Wedge, California Institute of Technology

Richard Compton, Cornell University

David Rutledge, California Institute of Technology

PUFF is a program for laying out and analyzing microstrip and stripline circuits on IBM-compatible personal computers. The program is fast and easy to use. Circuits are laid out on the screen using cursor keys, and may be analyzed for scattering parameters in the frequency domain, or impulse and step responses in the time domain. PUFF's element library includes several lumped elements, transmission lines, and coupled lines. Multiport scattering parameter files may be read into the program to represent transistors, to create hypothetical components, or to compare theory and experiment. PUFF also produces photographic artwork on dot-matrix and HP LaserJet printers. Support for VGA, EGA, and CGA graphics is included.

Over 10,000 copies distributed.