INTERCONNECTING THE ELECTROMAGNETIC FIELD SIMULATOR Em WITH OSA90/hope

OSA-92-OS-3-R

April 7, 1992

I. ELECTROMAGNETIC SIMULATOR FROM SONNET SOFTWARE, INC.

The electromagnetic simulator *Em* from Sonnet Software, Inc. [1] is a "full-wave" analysis program. It performs an electromagnetic analysis of a microstrip circuit by solving for the current distribution in the microstrip metallization using the Method of Moments. The analysis takes into account all possible coupling mechanisms. It includes all dispersion, stray coupling, discontinuities, surface waves, moding, metallization loss and radiation loss. The user provides *Em* with the complete description of the material and geometry as well as with simulation instructions. *Em* calculates the *S* parameters very accurately. De-embedding of the microstrip-to-coaxial transition (connector) discontinuity and the length of interconnecting transmission line (reference plane shift) is provided as an option to the user. Important features of *Em* include simulation of shielding effects, resonance frequency, via hole, etc.

Em is implemented in a non-window environment, i.e., it is running under normal text mode. It is typically 100 times faster than the finite element technique for the class of predominantly planar circuits, according to Sonnet Software.

To prepare the material and geometry data file required by *Em*, Sonnet Software provides an accompanying program named *Xgeom*. It is a graphical editor running under X windows. The user can specify the substrate and graphically enter the geometrical structure of the circuit (component) to be simulated by *Em*.

II. INTERFACING Em TO OSA90/hope THROUGH Datapipe

Em's primary use is for design verification to validate large portions of the final circuit. So far, *Em* does not handle active devices. As stated in the *Em User's Manual*, "*Em* is designed to work with your CAE software, enhancing your capability, in a 'win-win' relationship".

Utilizing the special *Datapipe* feature of *OSA90/hope* [2], we have created a *Datapipe* interface called *Empipe*. It utilizes an inter-program communication mechanism for transferring data between *OSA90/hope* and certain external programs, for example, independent electromagnetic field

simulators. With *Empipe*, *OSA90/hope* can perform complete circuit simulation utilizing *Em* for passive planar structure analysis. The general structure of the connection is shown in Fig. 1. *OSA90/hope* is the parent program, the *Empipe* interface is the child program and *Em* is the grandchild.

The *Empipe* interface is realized using two *Datapipes* as described in Fig. 2. The first *Datapipe*, shown in Fig. 2(a), creates the data file for *Em*. We use the *COM Datapipe* to pass the component index and data file contents to the child program. The component index is needed to distinguish different structural components present in one *OSA90/hope* circuit file. The file contents are passed as a character string. The data file is then created and stored on the hard disk.

The second *Datapipe*, shown in Fig. 2(b), is a *LINEAR Datapipe*. As input to the child program, *OSA90/hope* passes the component index, frequency point and frequency range. The frequency point is the frequency specified for the current simulation. The frequency range (optional) corresponds to the frequency range specified in the sweep block of the circuit file. After entering the child program, we first create the simulation control file for *Em*. The control file describes the frequency unit and the actual frequency or frequencies at which simulation is desired. *Em* simulation is performed under two conditions, i.e., if the data file is new or if the frequency point (or range) was not covered in the previous runs with the same data file. If *Em* simulation is required, the *S* parameters will be extracted from the response file generated by *Em* and then saved in another file for later use. If *Em* simulation is not required, the *S* parameters are retrieved from an existing *S* parameter file. The child program ends by returning the *S* parameters at the specified frequency point to *OSA90/hope*.

Appendices A and B contain source listings of the two child programs described above.

III. CIRCUIT OPTIMIZATION EMPLOYING Empipe

Em simulation requires two input files as mentioned in Section II, i.e., a simulation control file and a data file. The data file contains the physical nature, geometrical conditions, port and

reference plane locations of the planar structure. The complex geometrical form of a microstrip structure is described by its two dimensional coordinates. Therefore *Em* is very flexible in handling complicated microstrip shapes. However, a change in a simple geometrical dimension can not be easily achieved by this coordinate description format.

To employ *Em* for simulation of common passive microstrip structures using typical descriptions, we have created *SIM Datapipes* for different simple microstrip components, e.g., single microstrip line, step and tee discontinuities. The general structure of such a *SIM Datapipe* is illustrated in Fig. 3. Instead of passing a whole data file generated by *Xgeom* as a character string as discussed in Section II, we pass parameters such as substrate thickness, length and width, etc., of the structure. The child programs determine the substrate size, cell size and translate the dimensions of the particular structure as well as port and reference plane locations to the corresponding planar coordinates. The data file is then created for *Em* simulation. Hence, the parameters of simple microstrip components become readily available within *OSA90/hope* for, for example, optimization.

The accuracy of *Em* simulation is largely controlled by the cell size specified in the child programs. We use at least 20 cells per wavelength as recommended in [1]. The wavelength is calculated from the maximum simulation frequency specified in a *Datapipe* input to the child program.

A child program for creating a single microstrip line data file is listed in Appendix C.

IV. APPLICATIONS

Microstrip Rectangular Structure

A microstrip rectangular structure simulation is chosen to demonstrate the communication between *OSA90/hope* and *Em*. The structure from Giannini *et al.* [3] is shown in Fig. 4. We first used *Xgeom* to create a data file for the structure. The data file was then included in an *OSA90/hope* circuit file listed in Appendix D. The simulation was from 2GHz to 12GHz with 0.1GHz step. A Sun SPARCstation 1 was used. The CPU time used by running *OSA90/hope* with *Empipe* connection to

Em was approximately the time used by running standalone *Em* plus the time *OSA90/hope* used to retrieve the *S* parameters from a file. Fig. 5 illustrates $|S_{21}|$ in dB. It is close to the measurements published in [3].

Small-Signal Amplifier

Fig. 6 depicts a small-signal amplifier. We use *Em* to simulate the input and output matching microstrip structures. The circuit file is listed in Appendix E. Two separate *Em* data files representing the input and output structures, respectively, are included in one *OSA90/hope* circuit file. Circuit responses using *OSA90/hope* built-in microstrip components and using *Em* simulated *S* parameters for the structures are shown in Fig. 7.

Design of an Impedance Transformer

To demonstrate the feasibility of circuit optimization employing *Em* simulator, we consider design of an impedance transformer. Two step discontinuities are used to realize the impedance transformer, as shown in Fig. 8. The source and load impedances are assumed 50 and 100 Ohms. We minimize the input and output reflection coefficients. The thickness and dielectric constant of the substrate are 0.635mm and 9.7, respectively. We fix W_1, L_1, L_2 and L_3, while W_2 and W_3 are chosen as optimization variables. The circuit is described by two microstrip/step components and one single line component which are simulated through *Empipe*. W_2 affects two steps and W_3 affects one step and the line. See the circuit file listed in Appendix F.

At the starting point $\{W_2, W_3\}$ = {0.3mm, 0.15mm} and the l_2 error is 0.242966. After 19 iterations, the error is reduced to 0.145583 with $\{W_2, W_3\}$ = {0.243mm, 0.145mm}. Each iteration uses about 6 CPU minutes on a SPARCstation 1 and most of the CPU time is consumed by *Em*.

V. CONCLUSIONS

The efficiency of *Em* is quite impressive. For the structures we tested which have less than 120 subsections it takes just a few seconds to process a single frequency point computation.

We have noticed that the *S* parameters computed by *Em* may differ depending on whether the

frequency point considered is specified alone or is included in a frequency sweep. For instance, the response curve for the microstrip rectangular structure is different from Fig. 5 if *Em* is initiated at each frequency point separately, as illustrated in Fig. 9.

Em could not handle very low frequency simulation. This situation is encountered when we include *Em* in nonlinear small-signal or large-signal HB simulations where zero frequency responses are required for the structures.

Em assumes that all the coordinates specified in the data file are located exactly on the grid. If not, it will round the coordinates to their closest grid points. This situation normally does not occur when a data file is created by *Xgeom*. However, when using the option described in Section III, small discretization errors may occur in such situations.

By linking *Em* with *OSA90/hope* we take advantage of both systems and provide the user with a CAD system which has the versatility of a general CAD program with very accurate planar structure simulation capabilities.

Further investigations can be done on such aspects as an intelligent treatment of zero frequency simulation at the child program level, quadratic modeling for efficient use of the *Em* simulator, etc.

REFERENCES

[1] *Em User's Manual* and *Xgeom User's Manual*, Sonnet Software, Inc., 135 Old Cove Road, Suite 203, Liverpool, NY 13090-3774, Jan. 1991.

[2] *OSA90/hope User's Manual*, Optimization Systems Associates Inc., P.O.Box 8083, Dundas, Ontario, Canada L9H 5E7, 1991.

[3] F. Giannini, G. Bartolucci and M. Ruggieri, "Equivalent circuit models for computer aided design of microstrip rectangular structures," *IEEE Trans. Microwave Theory Tech.*, vol. 40, 1992, pp. 378-388.

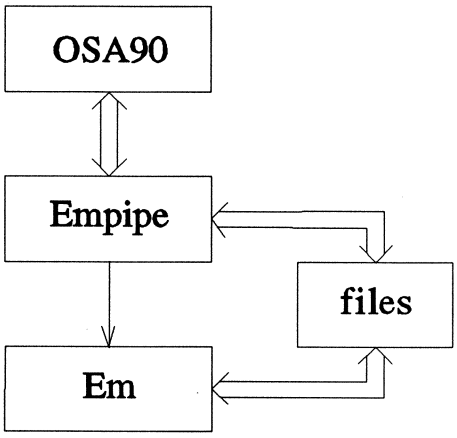
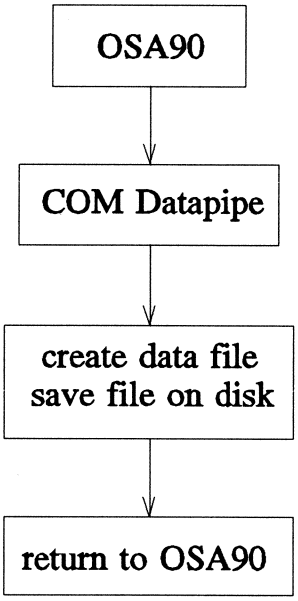
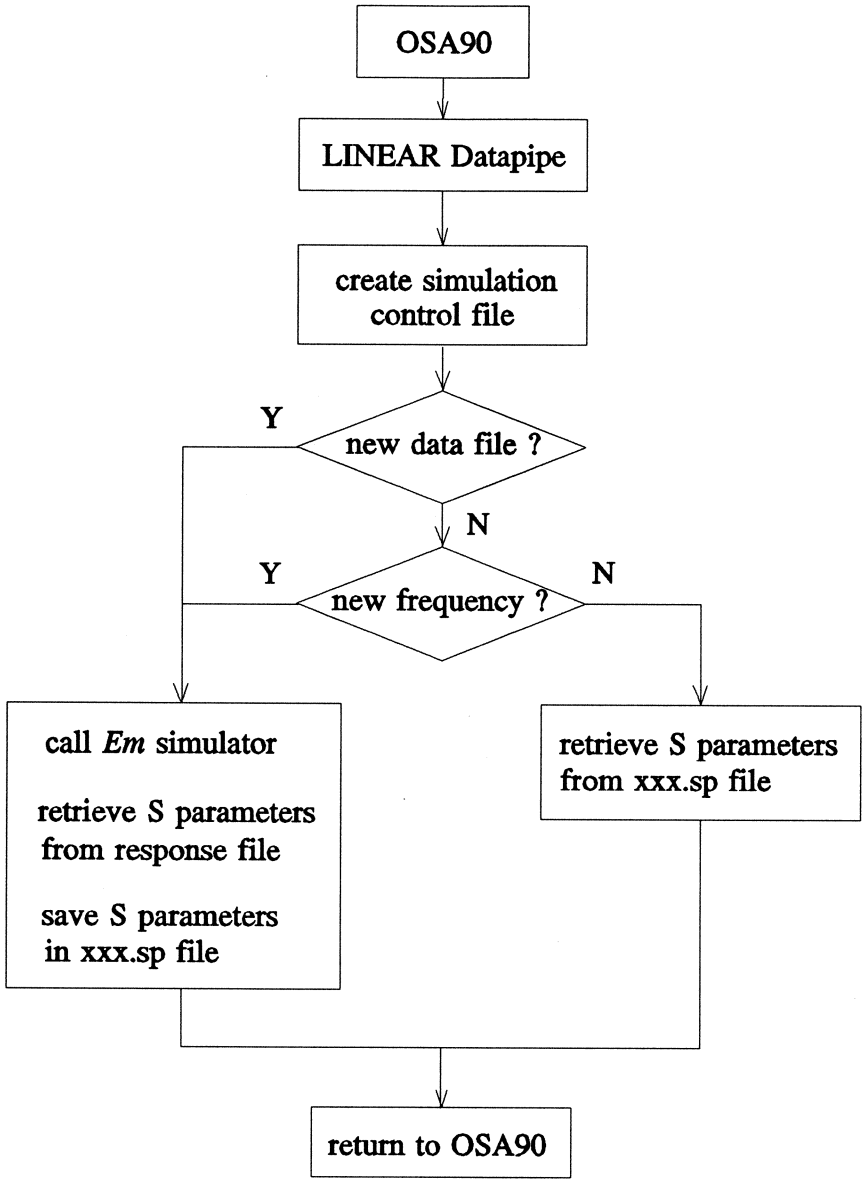


Fig. 1. General structure of the connection between *OSA90/hope* and *Em*.



(a)



(b)

Fig. 2. Flow diagram illustrating *Empipe* for interfacing *Em* simulator with *OSA90/hope*. (a) Creation of the data file for *Em* by *COM Datapipe*. (b) *S* parameter computation by *Em* and transferring to *OSA90/hope* by *LINEAR Datapipe*.

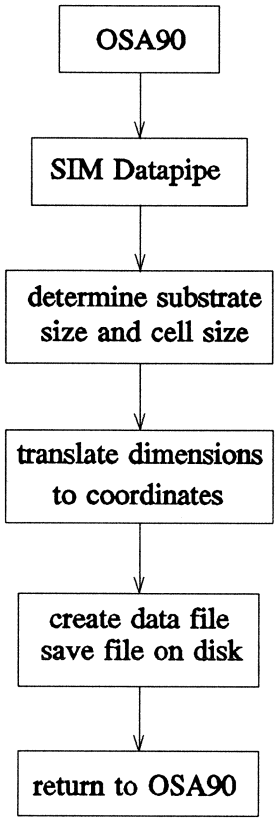


Fig. 3. General structure of the *SIM Datapipe* used to create *Em* data file from geometrical dimensions.

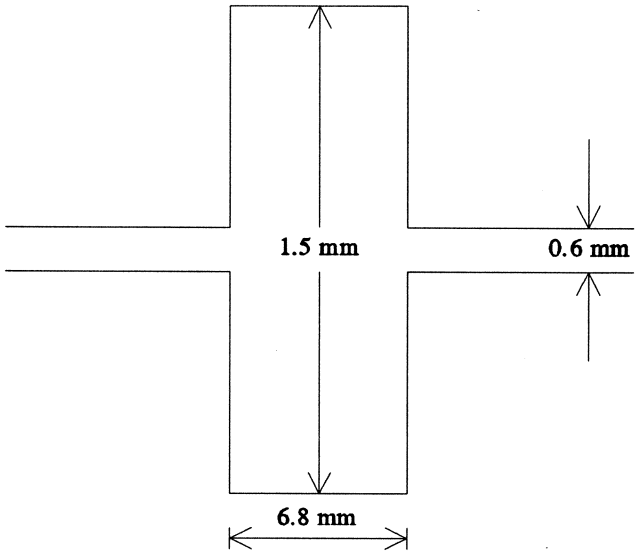


Fig. 4. A microstrip rectangular structure where the thickness and dielectric constant of the substrate are 0.635mm and 9.7, respectively.

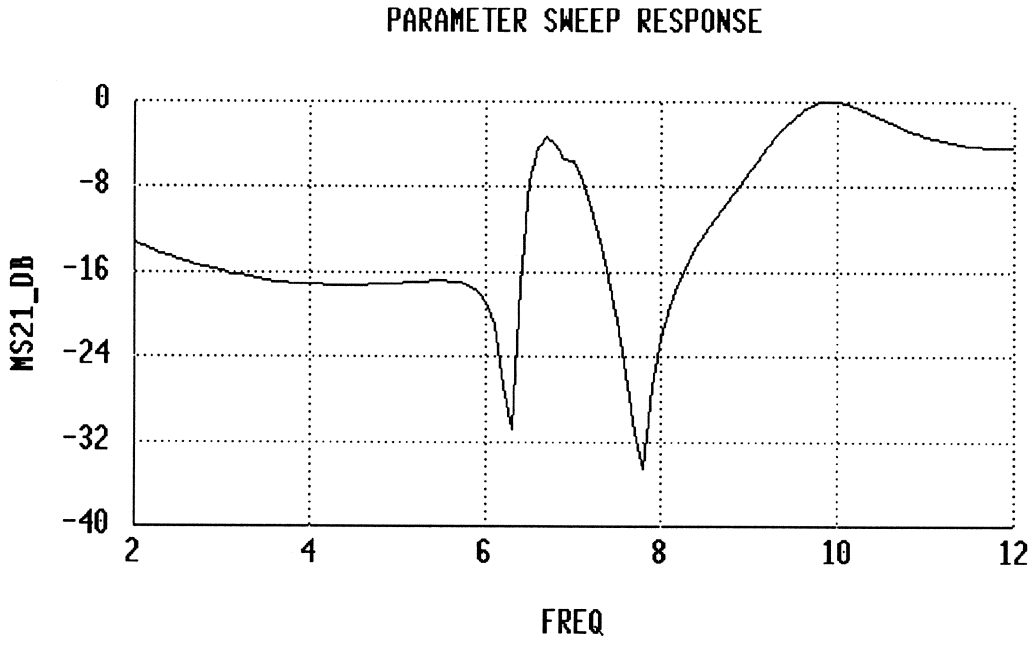


Fig. 5. $|S_{21}|$ in dB from the rectangular structure where the frequency is in GHz.

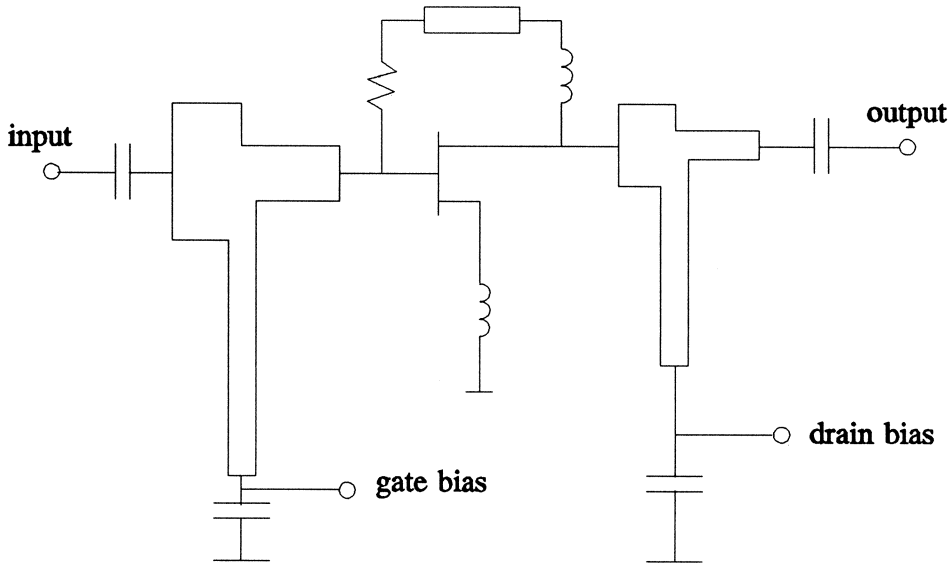
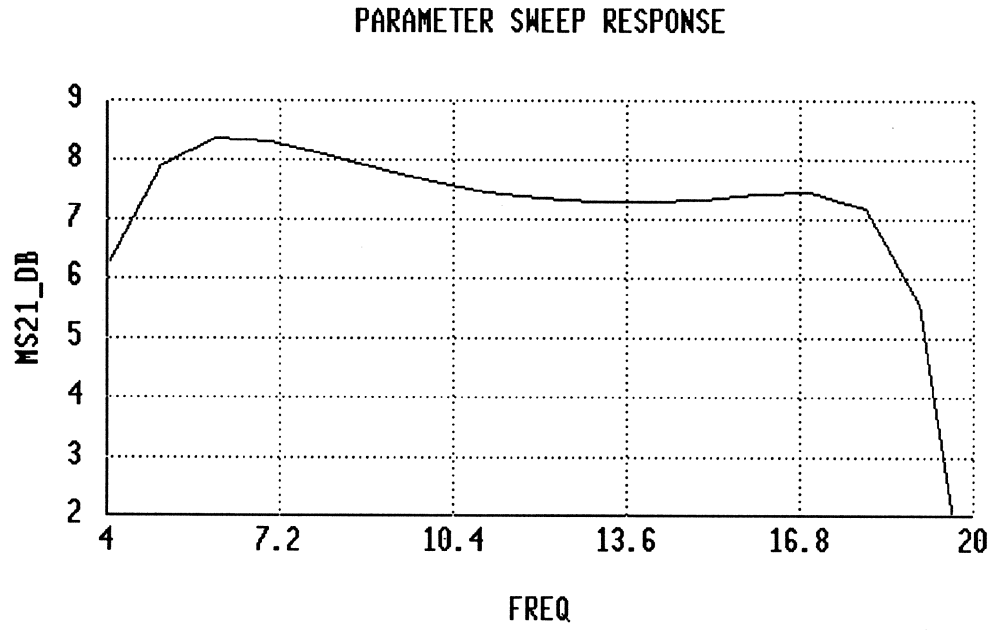
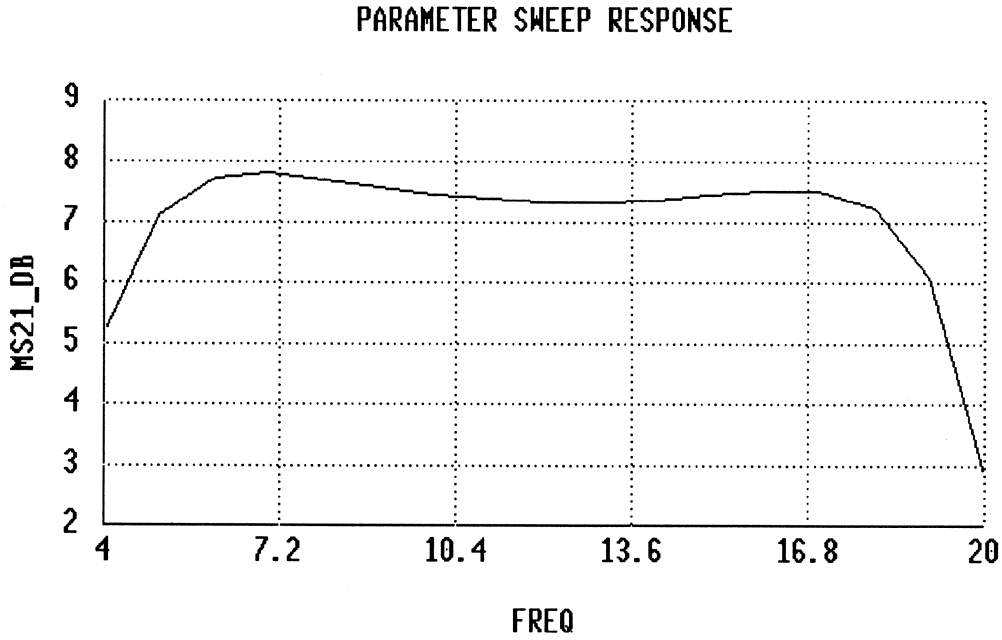


Fig. 6. Circuit diagram of the small-signal amplifier.



(a)



(b)

Fig. 7. The gain of the small-signal amplifier where the input and output microstrip matching structures are simulated by (a) using *OSA90/hope* built-in microstrip components and (b) using the *OSA90/hope Empipe* system connecting *Em*.

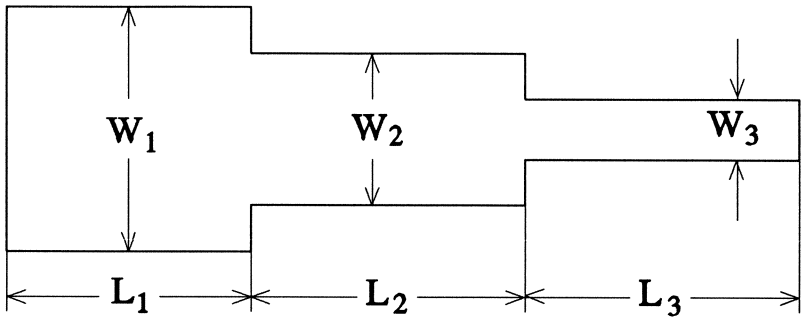


Fig. 8. Schematic diagram of an impedance transformer.

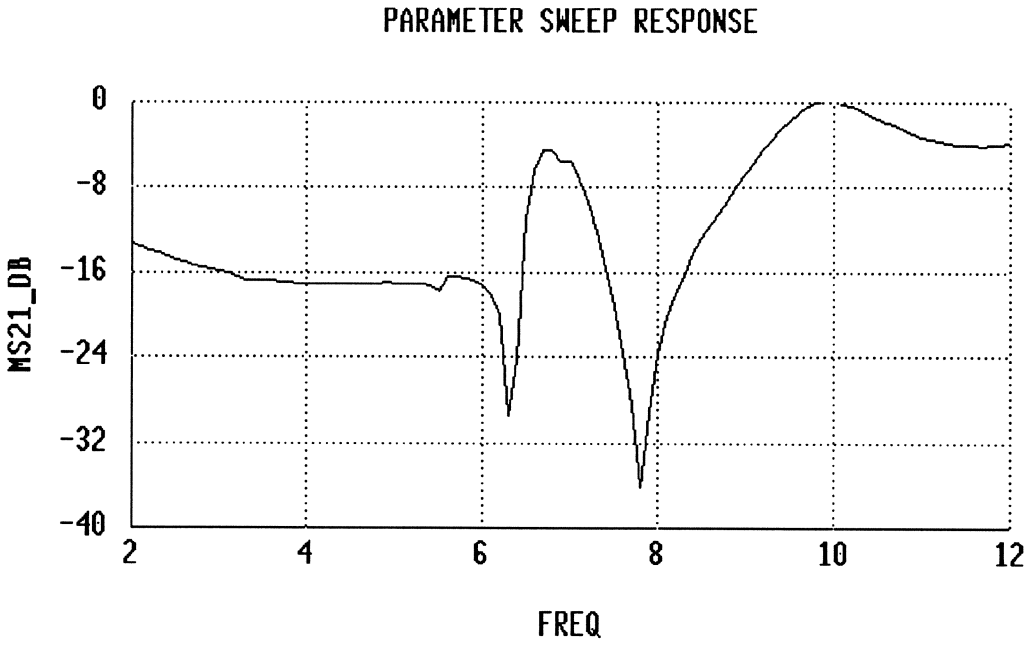


Fig. 9. $|S_{21}|$ in dB from the rectangular structure where the frequency is in GHz. The frequency step size is 0.1GHz. This curve is obtained by calling *Empipe* separately for each individual frequency point.

APPENDIX A. CHILD PROGRAM FOR DATA FILE CREATION

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "ippcv2.h"

void* mymalloc();          /* used to allocate memory for input data */
char  error_str[128];      /* used to store the error string */

void main()
{
  int    input_no, /* number of input variables (=2) */
         output_no,/* number of output variables (=1, dummy) */
         group_no, /* number of groups (=2) */
         i,        /* loop counter */
         error=0,  /* stores the length of an error message (or -1) */
         data_type,/* stores the type of a group */
         data_size;/* stores the size of a group */
  void** input_data;/* contains addresses to the data groups */
  float  output=0; /* at least one output must be present */
  FILE*  data_file;/* sppac circuit file pointer */
  int    index;
  float  f_index;
  char cir_file[15], file_header[10], file_index[2];

  for (;;)
  {
    pipe_initialize2();
    pipe_read2(&input_no, sizeof(int), 1);
    pipe_read2(&output_no, sizeof(int), 1);
    pipe_read2(&group_no, sizeof(int), 1);
    input_data=mymalloc(group_no*sizeof(void*));
    for(i=0;i<group_no;i++)
    {
      pipe_read2(&data_type, sizeof(int), 1);
      pipe_read2(&data_size, sizeof(int), 1);
      if(data_type==IPPC_DATA_CHAR)
      {
        input_data[i]=mymalloc(data_size);
        pipe_read2(input_data[i], 1, data_size);
      }
      else  /* dat_type must be IPPC_DATA_FLOAT */
      {
        pipe_read2(&f_index, sizeof(float), data_size);
      }
    }

    /* create indexed circuit file for em simulation */
    index = (int) f_index;

     if ((index >= 0) && (index <= 26))
        file_index[0] = 'A' + index;
      /*  else error = 4; */

     file_index[1] = '\0';
     strcpy(file_header, "tmp00");
     strcat(file_header, file_index);
     strcpy(cir_file, file_header);
     strcat(cir_file, ".geo");
    /* write to the 'tmp00?.geo' */
    if((data_file=fopen(cir_file, "w"))!=NULL)
    {
      fprintf(data_file, "%s", (char*)input_data[1]);
      fclose(data_file);
    }
    else           /* opening error occurred */
```

```
    {
      strcpy(error_str, "Error in opening '");
      error=strlen(strcat(strcat(error_str, (char*)input_data[0]), "'."))+1;
    }
    free(input_data[1]);
    free(input_data);
    pipe_write2(&error, sizeof(int), 1);
    if(error)
      pipe_write2(error_str, 1, error);
    else
    {
      data_type=IPPC_DATA_FLOAT;
      pipe_write2(&data_type, sizeof(int), 1);
      pipe_write2(&output_no, sizeof(int), 1);
      pipe_write2(&output, sizeof(float), output_no);
    }
  }
}

void* mymalloc(malloc_size)
int malloc_size;
{
  void* pointer=NULL;
  int   error=0;

  if((pointer=(void*)malloc(malloc_size))==NULL)
  {   /* memory alloc. error occurred, return the error and terminate the child */
    error=strlen(strcpy(
          error_str, "Memory allocation error in 'create_cir'."))+1;
    pipe_write2(&error, sizeof(int), 1);
    pipe_write2(error_str, 1, error);
    pipe_initialize2();
  }
  return(pointer);
}
```

APPENDIX B. CHILD PROGRAM FOR S PARAMETER CALCULATION BY Em

```
#include <stdio.h>;
#include <string.h>
#include <unistd.h>
#include <math.h>
#include "ippcv2.h"

#define N_MAX  6
#define M_MAX  8

main()
{
  int input_no,
      output_no,
      error;
  float x[N_MAX], y[2 * M_MAX * M_MAX];
  double freq_start, freq_end, freq_step, freq_point;
  float freq0, freq1, mag, ang, scale;
  float deg_to_ang, ang0, ang1, tmp0, tmp1;
  float sp0[2 * M_MAX * M_MAX], sp1[2 * M_MAX * M_MAX];
  int i, status, index, diff;
  char *message, string1[81], command_line[75];
  char cir_file[12], ctl_file[12], rsp_file[12], sp_file[12];
  char cir_file_sav[16], ctl_file_sav[16], rsp_file_sav[16];
  char file_header[10], file_index[2];
  FILE *file1, *file2;

  input_no = output_no = error = 0;

  for (;;) {
     pipe_initialize2();  /* synchronization */
     pipe_read2(&input_no,  sizeof(int), 1);
     pipe_read2(&output_no, sizeof(int), 1);
     pipe_read2(x, sizeof(float), input_no);

     /* create the control file */
     index = (int) x[0];
     freq_point = x[2];
     if (input_no > 3) freq_start = x[3];
     else freq_start = freq_point;
     if (input_no > 4) freq_end = x[4];
     else freq_end = freq_start;
     if (input_no > 5) freq_step = x[5];
     else freq_step = 0.0;

     /* em does not alow 0 freq. */
     if (freq_start < 1.0E-8) {
        freq_start = 1.0E-8;
        freq_end   = freq_end + 1.0E-8;
     }

     if ((index >= 0) && (index <= 26))
        file_index[0] = 'A' + index;
      /*  else error = 4; */

     file_index[1] = '\0';
     strcpy(file_header, "tmp00");
     strcat(file_header, file_index);
     strcpy(cir_file, file_header);
     strcpy(ctl_file, file_header);
     strcpy(rsp_file, file_header);
     strcpy(sp_file,  file_header);
     strcat(cir_file, ".geo");
     strcat(ctl_file, ".an" );
     strcat(rsp_file, ".rsp");
     strcat(sp_file,  ".sp");
```

```
   strcpy(cir_file_sav, cir_file);
   strcpy(ctl_file_sav, ctl_file);
   strcpy(rsp_file_sav, rsp_file);
   strcat(cir_file_sav, ".sav");
   strcat(ctl_file_sav, ".sav");
   strcat(rsp_file_sav, ".sav");

   /* create the simulation control file for em */
   file1 = fopen(ctl_file, "w");
   fprintf(file1, "VER 2.1 \n");
   fprintf(file1, "GHZ \n");

   if (freq_start == freq_end)
      fprintf(file1,"FRE %15.8E \n", freq_start);
   else if (freq_step == 0.0)
      fprintf(file1,"FRE %15.8E %15.8Ef \n", freq_start, freq_end);
   else
      fprintf(file1,"FRE %15.8E %15.8E %15.8E \n",
              freq_start, freq_end, freq_step);
   fclose(file1);

   diff = 1;
   /* check whether this circuit has been simulated already */
   if ( (access(rsp_file_sav, R_OK) == 0) &&
        (access(cir_file_sav, R_OK) == 0) &&
        (access(ctl_file_sav, R_OK) == 0) ) {
      /* compare cir_file with cir_file_sav */
      strcpy(command_line, "diff -b -i ");
      strcat(command_line, cir_file);
      strcat(command_line, " ");
      strcat(command_line, cir_file_sav);
      strcat(command_line, " > tmp");
      diff = system( command_line );

      /* compare ctl_file with ctl_file_sav */
      if (diff == 0) {
         strcpy(command_line, "diff -b -i ");
         strcat(command_line, ctl_file);
         strcat(command_line, " ");
         strcat(command_line, ctl_file_sav);
         strcat(command_line, " > tmp");
         diff = system( command_line );
      }
   }
   if (diff != 0) {
      /* call em to simulate the component */
      strcpy(command_line, "em -Qdm ");
      strcat(command_line, cir_file);
      strcat(command_line, " -a ");
      strcat(command_line, ctl_file);
      strcat(command_line, " -r ");
      strcat(command_line, rsp_file);
      strcat(command_line, " > tmp");

      if (fork() == 0)
         execl("/bin/sh", "sh", "-c", command_line, NULL);
      wait(&status);

/*    copy circuit file to a bak file */
      strcpy(command_line, "cp ");
      strcat(command_line, cir_file);
      strcat(command_line, " ");
      strcat(command_line, cir_file_sav);
      i = system(command_line);
/*    copy control file to a bak file */
      strcpy(command_line, "cp ");
      strcat(command_line, ctl_file);
      strcat(command_line, " ");
      strcat(command_line, ctl_file_sav);
```

```
      i = system(command_line);
/*    copy rsp file to a bak file */
      strcpy(command_line, "cp ");
      strcat(command_line, rsp_file);
      strcat(command_line, " ");
      strcat(command_line, rsp_file_sav);
      i = system(command_line);
/*    remove rsp file */
      strcpy(command_line, "rm ");
      strcat(command_line, rsp_file);
      i = system(command_line);
   }
/*    remove ctrl file */
   strcpy(command_line, "rm ");
   strcat(command_line, ctl_file);
   i = system(command_line);

/*    check if sp_file is available */
   if ( (diff != 0) || (access(sp_file, R_OK) != 0) ) {
   /* extract sp_file from rsp_file_sav */
      if ( ((file1 = fopen(rsp_file_sav, "r")) != NULL) &&
           ((file2 = fopen(sp_file, "w")) != NULL)) {
         for( ; !feof(file1) ; ) {
            fgets(string1, 80, file1);
         /* search for the first non-blank character */
            for (i=0; i<strlen(string1); i++)
               if (string1[i] != ' ') break;
         /* check if the first non-blank character is a number */
            if ( (string1[i] >= '0') &&
                 (string1[i] <= '9') )
               fprintf(file2, "%s", string1);
         }
         fclose(file1);
         fclose(file2);
      }
      /* else error */
   }

   /* read the s parameters from sp_file */
   file1 = fopen(sp_file, "r");
   /* read in the first line : freq + s-parameters */
   fscanf(file1, "%f", &freq0);
   for (i=0; i<output_no; i++) fscanf(file1, "%f", &sp0[i]);

   if (freq_point > freq0) {
      for ( ; !feof(file1) ; ) {
         fscanf(file1, "%f", &freq1);
         for (i=0; i<output_no; i++) fscanf(file1, "%f", &sp1[i]);
         if (freq_point <= freq1 ) break;
         freq0 = freq1;
         for (i=0; i<output_no; i++) sp0[i] = sp1[i];
      }
   }
   else {
      freq1 = freq0;
      for (i=0; i<output_no; i++) sp1[i] = sp0[i];
   }
   fclose(file1);

   deg_to_ang = 3.1415926 / 180.0;
   /* convert s-parameters from polar to rectangular */
   if (freq0 != freq1) {
      scale = (freq_point - freq0) / (freq1 - freq0);
      for ( i=0; i<output_no; i=i+2) {
         ang0 = sp0[i+1] * deg_to_ang;
         ang1 = sp1[i+1] * deg_to_ang;
         tmp0 = sp0[i] * cos(ang0);
         tmp1 = sp1[i] * cos(ang1);
         y[i] = tmp0 + scale * (tmp1 - tmp0);
```

```
            tmp0 = sp0[i] * sin(ang0);
            tmp1 = sp1[i] * sin(ang1);
            y[i+1] = tmp0 + scale * (tmp1 - tmp0);
         }
      }
      else {
         for ( i=0; i<output_no; i=i+2) {
            mag = sp0[i  ];
            ang = sp0[i+1] * deg_to_ang;
            y[i] = mag * cos(ang);
            y[i+1] = mag * sin(ang);
         }
      }

      i = system("rm tmp");
      /* return to osa90 */
      pipe_write2(&error, sizeof(int), 1);
      if (!error)
          pipe_write2(y, sizeof(float), output_no);
      else
          pipe_write2(message, 1, error);
   }
}
```

APPENDIX C. CHILD PROGRAM FOR SINGLE MICROSTRIP LINE DATA FILE CREATION

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <sys/time.h>
#include "ippcv2.h"
#define  N_MAX 10
char  error_str[128];      /* used to store the error string */
time_t now;

void main()
{
   float f_input[N_MAX];
   int    input_no, /* number of input variables (=2) */
          output_no,/* number of output variables (=1, dummy) */
          error;  /* stores the length of an error message (or -1) */
   float  output; /* at least one output must be present */
   int    index;
   char   cir_file[15], file_header[10], file_index[2];
   int    n_e, m_e, n, m, n_box, m_box, i;
   float  w, l, h, er, fmax;
   float  xcell_e, ycell_e, xcell, ycell, x_box, y_box;
   float  x[5], y[5];
   float  lamda, C0 = 3.0E8;
   FILE   *file1;

  for (;;)
  {
    pipe_initialize2();
    pipe_read2(&input_no, sizeof(int), 1);
    pipe_read2(&output_no, sizeof(int), 1);
    pipe_read2(f_input, sizeof(float), input_no);

    error = output = 0;

    if (input_no != 6) {
       if (input_no < 6) strcpy(error_str, "not enough parameters to MSL-pipe");
       else strcpy(error_str, "too many parameters to MSL-pipe");
       error = strlen(error_str) + 1;
    }

/*   create indexed circuit file for em simulation */
    index = (int) f_input[0];
    if ((index >= 0) && (index <= 26))
       file_index[0] = 'A' + index;
       /*  else error = 4; */
     file_index[1] = '\0';
     strcpy(file_header, "tmp00");
     strcat(file_header, file_index);
     strcpy(cir_file, file_header);
     strcat(cir_file, ".geo");

/*  extract W, L, Er, H, fmin and fmax from the input string */
    w    = f_input[1];
    l    = f_input[2];
    h    = f_input[3];
    er   = f_input[4];
    fmax = (f_input[5] > 0.1) ? f_input[5] * 1.0e9 : 0.1e9;

    if ( (w <= 0.0) || (l <= 0.0) || (h <= 0.0) || (er < 1.0) ) {
       strcpy(error_str, "invalid parameter(s) to MSL-pipe");
       error = strlen(error_str) + 1;
    }

    if (error == 0) {
/*  calculate the wavelength and determine the approximate cell size */
    lamda = C0 / (sqrt(er) * fmax);
```

```
    xcell_e = ycell_e = lamda / 20.0;

/*  determine xcell from L and xcell_e */
    n_e = l / xcell_e;
    if (n_e < 2) n_e = 2;
    n = (n_e / 2) * 2;
    xcell = l / n;

/*  similarly determine ycell from W and ycell_e */
    m_e = w / ycell_e;
    if (m_e < 2) m_e = 2;
    m = (m_e / 2) * 2;
    ycell = w / m;

/*  box size */
    n_box = n;
    x_box = n_box * xcell;
    m_box = 9*m;
    y_box = m_box * ycell;

/*  convert all dimensions to mm */
    h     *= 1.0e3;
    xcell *= 1.0e3;
    ycell *= 1.0e3;
    x_box *= 1.0e3;
    y_box *= 1.0e3;

/*  arrange coordinates */
    x[0] = 0.0;   y[0] = 4*m * ycell;
    x[1] = 0.0;   y[1] = 5*m * ycell;
    x[2] = x_box; y[2] = y[1];
    x[3] = x_box; y[3] = y[0];
    x[4] = 0.0;   y[4] = y[0];

/*  create the data file for em simulator */
    /* write to the 'tmp00?.geo' */
    if((file1=fopen(cir_file, "w"))!=NULL)
    {
       fprintf(file1, "VER 2.2a\n");
/*
       now = time(NULL);
       fprintf(file1, "DAT %s", ctime(&now));
*/
       fprintf(file1, "LEN mm 1.0e-3\n");
       fprintf(file1, "TOP 377 0 0\n");
       fprintf(file1, "TON 0 Top Cover\n");
       fprintf(file1, "SYM\n");
       fprintf(file1, "REF 0.0  0.0  0.0  0.0\n");
       fprintf(file1, "BOX 1 %15.7E %15.7E %d %d 20\n",
                       x_box,y_box,2*n_box,2*m_box);
       fprintf(file1, "      %f 1.0  1.0  0  0\n", 200*h);
       fprintf(file1, "      %f %f  1.0  0  0\n", h, er);
       fprintf(file1, "POR 0 0 0 1  50.0  0.0  0.0  0.0\n");
       fprintf(file1, "POR 0 2 0 2  50.0  0.0  0.0  0.0\n");
       fprintf(file1, "NUM 1\n 0 5 -1 N  0 1 1 100 100\n");
       for (i = 0; i < 5; i++)
       fprintf(file1, "%15.7E  %15.7E\n", x[i], y[i]);
       fprintf(file1, "END\n");
       fclose(file1);
    }
    else           /* opening error occurred */
    {
      strcpy(error_str, "Error in opening '");
      error=strlen(strcat(strcat(error_str, cir_file), "'."))+1;
    }

    }

    pipe_write2(&error, sizeof(int), 1);
```

```
    if(error)
      pipe_write2(error_str, 1, error);
    else
    {
      pipe_write2(&output, sizeof(float), output_no);
    }
  }

}
```

APPENDIX D. CIRCUIT FILE FOR MICROWAVE RECTANGULAR STRUCTURE

```
! cross01.ckt
! 19-MAR-1992
! test of a microstrip cross section:
! substrate size: 17x26 mm, with min 20 subsections per wavelength.
!
#define sub_circuit10 7
Model
   Datapipe: LINEAR FILE="em_sim3" NAME = new1
             N_INPUT = 6 INPUT=(sub_circuit10, out10, FREQ, 2, 12, 0.05)
             N_PORT  = 2  FORMAT=S
             timeout = 0;

   DATAPORT @data1 @data2 DATA = new1;
   port @data1 @ground;
   port @data2 @ground;

   CIRCUIT;

   ms21_db = 20*log10(ms21);

end

Sweep
   AC: freq: from 2ghz to 12ghz step 0.1ghz ms ps ms21_db;
end

Expression
char cir_contents10[] =
"
VER 2.2d
LIC none
DAT Wed Mar 18 10:20:20 1992
LEN mm 1.0000000000e-03
TOP 377 0 0
TON 0 Top Cover
SYM
REF       0.0       0.0       0.0       0.0
BOX 1 17.000000 24.000000 80 160 20
      100.00000 1.0000 1.0000 0 0
      0.6350000 9.7000 1.0000 0 0
POR 0 11 0 1 50.000000 0.000000 0.000000 0.000000
POR 0 5 0 2 50.000000 0.000000 0.000000 0.000000
NUM 1
0 13 -1 B  0 1 1 100 100
      0.0 11.700000
5.1000000 11.700000
5.1000000 4.5000000
11.900000 4.5000000
11.900000 11.700000
17.000000 11.700000
17.000000 12.300000
11.900000 12.300000
11.900000 19.500000
5.1000000 19.500000
5.1000000 12.300000
      0.0 12.300000
      0.0 11.700000
END
";
   Datapipe:      COM           FILE = "create_file1"
                  N_INPUT = 2   INPUT = (sub_circuit10, cir_contents10)
                  N_OUTPUT= 1   OUTPUT= (out10);
end
```

APPENDIX E. CIRCUIT FILE FOR A SMALL-SIGNAL AMPLIFIER

```
! msl_amp.ckt
! 19-Mar-1992
!  6-18GHz microstrip amplifier stage with drain to gate feedback
#define SUBS EPSR=9.9  H=10MIL T=0.15MIL ROC=0.0 RHS=0
#define sub_circuit03 18
#define sub_circuit04 19
Model

! parameters for the input matching structure
! simulation by Em through LINEAR Datapipe
   Datapipe: LINEAR FILE="em_sim3" NAME = match_in
             N_INPUT = 6 INPUT=(sub_circuit03, out3, FREQ, 4, 20.0001, 1)
             N_PORT  = 3  FORMAT=S
             timeout = 0;
! parameters if simulation by OSA90/hope microstrip components
   W1=24MIL;
   L1=15MIL;
   W2=8MIL;
   L2=29.98MIL;
   WSC1=3.33MIL;
   LSC1=108MIL;

! parameters for the output matching structure
! simulation by Em through LINEAR Datapipe
   Datapipe: LINEAR FILE="em_sim3" NAME = match_out
             N_INPUT = 6 INPUT=(sub_circuit04, out4, FREQ, 4, 20.0001, 1)
             N_PORT  = 3  FORMAT=S
             timeout = 0;
! parameters if simulation by OSA90/hope microstrip components
   W3=9.9MIL;
   L3=6.25MIL;
   W4=3.3MIL;
   L4=11.25MIL;
   WSC2=3.75MIL;
   LSC2=49.5MIL;
!
   CX=6PF;
   RFB=1800;
   WFB=2MIL;
   LFB=11MIL;

   CAP  1 2  C=CX;

   DATAPORT 2 5 31 DATA = match_in;

!   MSL  2 3  W=W1 L=L1 SUBS;
!   MTEE  3 4 30  W1=W1 W2=W2 W3=WSC1 SUBS;
!   MSL  4 5  W=W2 L=L2 SUBS;
!   MSL  30 31  W=WSC1 L=LSC1 SUBS;

   CAP  31 0  C=10PF;
   RES  5 20  R=RFB;
   MSL  20 21  W=WFB L=LFB SUBS;
   CAP  21 6  C=CX;

   DATAPORT 6 9 41 DATA = match_out;

!   MSL  6 7  W=W3 L=L3 SUBS;
!   MTEE  7 8 40  W1=W3 W2=W4 W3=WSC2 SUBS;
!   MSL  8 9  W=W4 L=L4 SUBS;
!   MSL  40 41  W=WSC2 L=LSC2 SUBS;

   CAP  41 0  C=10PF;
   CAP  9 10  C=CX;
```

```
   EXTRINSIC2  @gate @drain @source 5 6
       RG=2  RD=2.4  RS=3.7
       CDS=0.07pF  GDS=(1 / 1250)  CX=10pF  LS=0.083NH;

   SRC  @drain @source  R=250 C=3NF;

   FETC @gate @drain @source
       A0=0.065  A1=0.115  A2=0.059  A3=0.009
       GAMMA=2  BETA=0.04  VDS0=2  IS=1.0e-9
       CGD0=0.06pF  CGS0=0.37pF;

   VSOURCE  31 0  NAME=GATE_BIAS VDC=-0.3V;
   VSOURCE  41 0  NAME=DRAIN_BIAS VDC=3V;

   PORT  1 0  NAME=INPUT; !P=PIN;

   PORT  10 0  NAME=OUT;

   CIRCUIT;

   MS21_DB = 20 * log10 ( ms21 );
   MS11_DB = 20 * log10 ( ms11 );
   MS22_DB = 20 * log10 ( ms22 );
end

Sweep
    AC: FREQ: from 4ghz to 20ghz step 1ghz MS11_DB MS22_DB MS21_DB ms ps
end

Expression
! Em data file for input microstrip structure
char cir_contents03[] =
"
VER 2.2a
LIC none
DAT Wed Mar 18 15:05:07 1992
TOP 377 0 0
TON 0 Top Cover
LEN mils 2.5400000000e-05
REF       0.0       0.0       0.0       0.0
BOX 1 48.300000 160.00000 58 80 20
      1000.0000 1.0000 1.0000 0 0
      10.000000 9.9000 1.0000 0 0
POR 0 7 0 1 50.000000 0.000000 0.000000 0.000000
POR 0 3 0 2 50.000000 0.000000 0.000000 0.000000
POR 0 0 0 3 50.000000 0.000000 0.000000 0.000000
NUM 1
0 11 -1 N  0 1 1 100 100
14.943333 160.00000
18.276667 160.00000
18.276667 44.000000
48.220000 44.000000
48.220000 36.000000
16.610000 36.000000
16.610000 28.000000
      0.0 28.000000
      0.0 52.000000
14.943333 52.000000
14.943333 160.00000
END
";
   Datapipe:      COM           FILE = "create_file1"
                  N_INPUT = 2   INPUT = (sub_circuit03, cir_contents03)
                  N_OUTPUT= 1   OUTPUT= (out3);

! Em data file for output microstrip structure
char cir_contents04[] =
"
VER 2.2a
```

```
LIC none
DAT Wed Mar 18 16:02:02 1992
TOP 377 0 0
TON 0 Top Cover
LEN mils 2.5400000000e-05
REF       0.0       0.0       0.0       0.0
BOX 1 21.250000 79.200000 34 96 20
      1000.0000 1.0000 1.0000 0 0
      10.000000 9.9000 1.0000 0 0
POR 0 8 0 1 50.000000 0.000000 0.000000 0.000000
POR 0 4 0 2 50.000000 0.000000 0.000000 0.000000
POR 0 0 0 3 50.000000 0.000000 0.000000 0.000000
NUM 1
0 12 -1 N  0 1 1 100 100
6.2500000 79.200000
10.000000 79.200000
10.000000 29.700000
10.000000 26.400000
21.250000 26.400000
21.250000 23.100000
10.000000 23.100000
10.000000 19.800000
0.0000000 19.800000
0.0000000 29.700000
6.2500000 29.700000
6.2500000 79.200000
END
";
   Datapipe:      COM           FILE = "create_file1"
                  N_INPUT = 2   INPUT = (sub_circuit04, cir_contents04)
                  N_OUTPUT= 1   OUTPUT= (out4);

end
```

APPENDIX F. CIRCUIT FILE FOR IMPEDANCE TRANSFORMER

```
#define step1 4
#define step2 5
#define line  6
#define subs epsr=9.7 t=0 tand=0 roc=0 rhs=0
Expression
    w1   = 0.0006;
    w2   = ?0.0003?;
    w3   = ?0.00015?;
    l1   = 0.004;
    l2   = 0.004;
    l3   = 0.004;
    h    = 0.635e-3;
    er   = 9.7;
    fmax = 15ghz;
Datapipe:      SIM           FILE = "step_pipe"
               N_INPUT = 7   INPUT = (step1, w1, w2, l1, h, er, fmax)
               N_OUTPUT= 1   OUTPUT= (out1)
               timeout = 0;
Datapipe:      SIM           FILE = "step_pipe"
               N_INPUT = 7   INPUT = (step2, w2, w3, l2, h, er, fmax)
               N_OUTPUT= 1   OUTPUT= (out2)
               timeout = 0;
Datapipe:      SIM           FILE = "msl_pipe"
               N_INPUT = 6   INPUT = (line, w3, l3, h, er, fmax)
               N_OUTPUT= 1   OUTPUT= (out3)
               timeout = 0;
end
Model
   Datapipe: LINEAR FILE="em_sim3" NAME = step01
             N_INPUT = 5 INPUT=(step1, out1, FREQ, 4, 5)
             N_PORT  = 2  FORMAT=S
             timeout = 0;
   Datapipe: LINEAR FILE="em_sim3" NAME = step02
             N_INPUT = 5 INPUT=(step2, out2, FREQ, 4, 5)
             N_PORT  = 2  FORMAT=S
             timeout = 0;
   Datapipe: LINEAR FILE="em_sim3" NAME = line01
             N_INPUT = 5 INPUT=(line, out3, FREQ, 4, 5)
             N_PORT  = 2  FORMAT=S
             timeout = 0;

!   msl   @data1  @data20 w=w1  l=l1  h=h subs;
!   mstep @data20 @data2  w1=w1 w2=w2 h=h subs;
!   msl   @data2  @data30 w=w2  l=l2  h=h subs;
!   mstep @data30 @data3  w1=w2 w2=w3 h=h subs;
!   msl   @data3  @data4  w=w3  l=l3  h=h subs;
   DATAPORT @data1 @data2 DATA = step01;
   DATAPORT @data2 @data3 DATA = step02;
   DATAPORT @data3 @data4 DATA = line01;
   port @data1 @ground name = input  R = 50;
   port @data4 @ground name = output R = 50;

   CIRCUIT;

   Z0 = 50;
   ZL = 100;
   ZS = 50;
   gamma_out = (ZL - Z0) / (ZL + Z0);
   gamma_in  = (ZS - Z0) / (ZS + Z0);

!  calculate the input reflection coef.
   m1 = ms12 * ms21 * gamma_out;
   p1 = ps12 + ps21;
   r2 = 1.0 - rs22 * gamma_out;
   i2 =      -is22 * gamma_out;
```

```
   RI2MP(r2, i2, m2, p2);
   m3 = m1/m2;
   p3 = p1 - p2;
   MP2RI(m3, p3, r3, i3);
   r4 = rs11 + r3;
   i4 = is11 + i3;
   RI2MP(r4, i4, m4, p4);

!  calculate the output reflection coef.
   m10 = ms12 * ms21 * gamma_in;
   p10 = ps12 + ps21;
   r5 = 1.0 - rs11 * gamma_in;
   i5 =      -is11 * gamma_in;
   RI2MP(r5, i5, m5, p5);
   m6 = m10/m5;
   p6 = p10 - p5;
   MP2RI(m6, p6, r6, i6);
   r7 = rs22 + r6;
   i7 = is22 + i6;
   RI2MP(r7, i7, m7, p7);

end

Sweep
   AC: freq: from 2ghz to 6ghz step 0.25ghz
                  m4 m7 ms ps;
end
specification
   AC: freq: 4ghz 5ghz m4 m7;
end
```