UDC 519.68:519.85:621.3
Original scientific paper
Received: 28.06.2004.

A new approach to spectral domain method: Functional programming

Bahadir Sevinc[(1)] and **Hasan H. Balik**[(2)]
[(1)]Department of Informatics, University of Firat, 23216 Elazig, TURKEY
e-mail: bahadirsevinc@firat.edu.tr
[(2)]Department of Electrical and Electronics Engineering, University of Firat, 23216 Elazig, TURKEY
e-mail: balik@firat.edu.tr

SUMMARY

The Spectral Domain Method (SDM) is a powerful technique to analyze planar microwave circuits. But available conventional programming languages used in the literature do not give enough speed to use the Spectral Domain Method to develop package analysis program. The functional approach to Spectral Domain Method gives a high level of programming and a variety of features which help to build elegant, powerful and general libraries of functions.

***Key words**: SDM, planar circuits, functional programming, Haskel, antennas.*

1. INTRODUCTION

For the last two decades, open microstrip structures have received special attention from the electromagnetic community because of their potential applications in the design of new devices and components. Meanwhile, high-speed computer has influenced the computation of electromagnetic problem to the point that most practical computations of the fields can be solved numerically by the computer. The reason why most of the analysis of the devices and components can be achieved numerically but is almost impossible to be solved analytically. A lot of efforts have still been done on improving numerical techniques because complexity of the problems always overstretch the speed of the processors. Moreover the operating frequency raised up for more available bandwidth, full-wave techniques which require more computer power and resources must be used.

A number of numerical full-wave techniques are reported in the literature for the analysis of microstrip antennas [1], resonators [2] and circuits [3]. All techniques reported in the literature have been either written by conventional programming languages such as Pascal and *C* or developed by using commercial analysis tools such as Matlab. In the author knowledge none of the papers can explore the idea of the way of programming such as functional or logic. This contribution presents a functional programming approach to Spectral Domain Method (SDM) which is one of the full wave numerical technique and widely used for the analysis of the microwave and millimeter wave devices and components. With this approach, Spectral Domain Method have gained a high level of programming giving its user a variety of features which help to build elegant yet powerfully and general libraries of functions. Numerical results have been also given and compared with published data to show the accuracy of the re-written Spectral Domain Method by Haskel which widely used functional programming language instead of conventional language such as Pascal used in Ref. [3].

2. FUNCTIONAL PROGRAMMING

2.1 Introduction

This section offers the readers the idea of the functional programming with variety of the general examples instead of the examples in electromagnetics to be understood clearly. Central part of functional programming is the idea of function, which computes a result that depends on the values of its inputs. An example of the power and generality of the language is the "map" function, which is used to transform every element of a list of objects in a specified way. For example, map can be used to double all the numbers in a sequence of numbers or invert the colors in each picture appearing in a list of pictures [4].

A simple illustration is the function "add" which adds two integers and doubles their sum. Its definition is:

add x y = 2(x+y)*

where *x* and *y* are the inputs and *2*(x+y)* is the result.

The model of functional programming is simple and clean; to work out the value of expression like:

? 3+add 4 5
~ 3+2 (4+5)*
~ ....
~ 21

This is how a computer would work out the value of expression, but it is also possible to do exactly the same calculation using pencil and paper, making transparent the implementation mechanism [4].

2.2 What is function?

A function is something which we can picture as a black box with some inputs and an output. Thus the function gives an output value which depends upon the input value. A simple example of a function is addition, + over numbers. Given input values *12* and *34* the corresponding output will be *46*.

2.3 Program structure

Programming is a form of problem solving. A program can be thought of as being developed in three stages. The first stage is specification, in which the initial informal requirements for the program are sharpened up so that the problem to be solved by the program is specified as precisely as possible [5].

The second stage is design, in which the problem to be solved is broken down into subproblems. Program components which solve the subproblems can be fitted together to solve the main problem.

The third stage is implementation in which the program components are written, debugged and fitted together.

A simple Haskell program:

--Find the square of an integer
square :: Integer ->Integer
*square n = n*n*

The first line of the program is a comment. In Haskell, any line beginning with "-" is a comment. The second line is a declaration. In Haskell, a declaration consists of a function name followed by the "::" symbol, followed by the type of each argument and the type of the result, separated by "- >" symbols. Type names begin with capital letters, whereas function names begin with lower case letters. The third line is a definition of the square function.

2.4 Data structure

Haskell has a mechanism for introducing new record-like data types using the data keyword. If a user wants to define a new type to hold information about a person for the purpose of storing personnel record in a database. This can be done in Haskell with a definition:

Data Personnel = Person String Integer Bool

This innocent looking definition does several things at the same time. First, it introduces a new type Personnel, second it introduces a new function called Person. Third, it implicitly specifies the type of the new function. Fourth, it provides a method called pattern matching for unpacking a personnel record and extracting its fields. The function Person is a special kind of function called a constructor. The name of the constructor always begin with a capital letter. A new mini database can be created from these definition:

people :: [*Personnel*]
people = [*Person "Sam" 25 True, Person "Merry" 40 False*

In this database the sex of a person as a boolean value is represented.

We can now build records, but as things stand so far we have no way to unpack them again to get at their fields. One way to do this would have a collection of functions for extracting the individual fields, like the head and tail functions which can be used to extract the two fields of a non-empty list.

However, Haskell provides a different mechanism called pattern-matching which is very convenient and expressive. A constructor such as Person can be re-used to break a record down into its fields, as well as to build new records up. The way this works is that the constructor is used on the left hand side of a definition. For example testing a person is an adult or not:

--Test whether a person is an adult
adult :: Personnel ->Bool
adult (*Person s n b*) = (*n>=18*)

The idea is that (*Person s n b*) acts as a pattern, with unknowns s, n and b when the adult function is applied to an argument expression is evaluated and then compared against the pattern. The pattern

matching technique allows you to extract all the fields at once and make them all available:

```
-- Exctract the fields from a personnel record
name :: Personnel -> String
name (Person s n b) = s
age :: Personnel -> Integer
age = (Person s n b) = n
sex :: Person -> Bool
sex (Person s n b) = b
```

A further feature of data definitions in Haskell is the use of alternatives. For example to carry out a search to find out where an item appears in a list. It is tempting to use some values such as (*-1*) which can not represent a position in the list to mean that the search fails. This might lead you to a definition such as:

```
search :: Eq a => a -> [a] -> Integer
search x xs =
if null xs then -1 else
if x == head xs then 0 else
      1+search * (tail xs)
```

In this function the use of the arithmetic result is not a good solution. We want two alternatives; if the search succeeds, we want to return the position of the item. If it fails, we just want to return the fact of failure:

```
-- Introduce a new type Answer with to represent
-- the result of a search for a position
data Answer = Yes Integer -> No
```

with the "*Answer*" type, there are two constructors. The "*Yes*" constructor builds a record with just one field, which is an integer, the "*No*" constructor builds a record with no fields at all. If the search example is - rewrite:

```
search :: Eq a => a -> [a] -> Answer
search x xs =
if null xs then No else
if x == head xs then Yes 0 else
increment (search x (tail xs))
increment :: Answer -> Answer
increment (Yes n) = Yes (n+1)
increment No = No
```

The constant constructor and alternatives define enumerated types.

```
--Introduce a new type Sex to represent the sex of
person
data Sex = Female | Male
```

Another example of pattern-matching is:

```
-- data Day = Mon | Tue | Wed | Tru | Fri | Sat | Sun
-- find the next day
next :: Day -> Day
next Mon = Tue
next Tue = Wed
next Wed = Thu
next Thu = Fri
next Fri = Sat
next Sat = Sun
next Sun = Mon
```

In some applications, the result is the same for many of the cases, and it is boring to write them all down. There is a default mechanism for catching all remaining cases. For example:

```
-- Test whether a day is a weekend day.
weekend Sat = True
weekend Sun = True
weekend d = False
```

When dealing with the problem of searching for the position of an item in a list, we introduce the answer type defined by:

```
data Answer = Yes Integer | No
```

Haskell deals with this by allowing types to be polymorphic, and then the constructors they introduce can also be polymorphic. For example the Answer type can be generalized:

```
data Answer a = Yes a | No
```

The word "*Answer*" is now a type constructor, it is a function which takes a type as its argument and constructs a new type as its results. The original search and increments functions would now have to change their declarations to specify that they act on one particular type of answer, namely Answer Integer:

```
-- search for an item in a list
search :: Eq a => a -> [a] -> Answer Integer
.....
-- Check a result and add one if it succeeded
increment :: Answer Integer -> Answer Integer
increment (Yes n) = Yes (n+1)
increment No = No
```

Polymorphic data types once more increase the expressive power and generality of the language, allowing facilities to be used in a wide variety of contexts.

One of the aims in the design of Haskell was to provide tools for the programmers which are as powerful as the tools needed to implement the standard data types, functions and notations.

```
-- Standard type : boolean values for testing and
choosing
data Bool = False | True
```

This explains why the boolean constants start with capital letters; they are constructors. It also means that they can be used for pattern-matching. For example:

```
xor :: Bool -> Bool -> Bool
xor False False = False
xor False True = True
xor True True = True
xor True True = False
```

On Integer and Char data types can be used pattern-matching for example:

```
-- Test the characters is blank or not
isSpace :: Char -> Bool
isSpace '\n'= True
isSpace c = False
-- Find the factorial of an integer
fact :: Integer -> Integer
fact 0 = 1
fact n = n*f act(n-1)
```

Pattern matching is used on the list:

```
myLength :: [a] -> Integer
myLength [ ] = 0
myLength (x:xs) = 1+myLength xs
```

As well as introducing new data types, it is also possible to define type synonyms. These have no other effect than to provide more readable abbreviations for existing types, to help make your own type declarations more readable:

```
type Position = (Integer, Integer)
```

This allows you to use pairs of integers as positions, which looks more natural then using a new data type for positions. At the same time, your own data declarations can use the Position type freely to document where pairs are being used as positions.

Type synonyms can be polymorphic, but they can not be recursive. There is one built-in synonym which have been used frequently; *String* is a synonym for "[*Char*]"

```
-- Standard type synonym : String
type String = [Char]
```

2.5 Recursion

The idea behind recursion is that a problem often breaks down into subproblems which are similar to the original. The simplest example is the loop, where a problem breaks down into a sequence of simpler subproblems of the same kind.

For example, take the problem of adding up a list of numbers. There is a standard function sum for doing this, and we know it can be implemented using folding. However, suppose we want to implement it directly.

When solving list problems the most obvious approach is usually to run along the list dealing with the items one by one. In the summing problem it is usual to run along the list which has been dealt with, the subproblem left it to add up the remaining list of items and so on:

```
subProblems :
mySum [5,4,3,2,1]
mySum [4,3,2]
mySum [3,2]
mySum [2]
mySum []
```

An example of a recursion:

```
mySum :: [Integer] ->[Integer]
mySum ns =
if null ns then 0 else
      head ns + mySum (tail ns)
```

2.6 Modules

Using modules to structure a large program has a number of advantages. These can be itemized as follows:

- Parts of the system can be built separately from each other;
- Part of a system can be compiled separately; this is a great advantage for a system of any complexity;
- Libraries of components can be reused, by importing the appropriate modules containing them.

In the definition of Haskell, there is no identification between modules and files. Nevertheless, we chose here to write one module per file.

3. SPECTRAL DOMAIN METHOD

Although it would be impossible to include all works concerning the Spectral Domain Method (SDM), even when there is a restricting interest to implementations for planar microwave circuits, it is hoped to present a guide to the historical background which inspired the development of this contribution.

The analysis in the Fourier transform domain was first introduced by Yamashita and Mittra [6] in 1968 for the computation of the characteristic impedance and the phase velocity of an open microstrip. It was based on a quasi-TEM approximation which is a low frequency simplification that neglects the longitudinal electric and magnetic fields supported by the microstrip.

Increasing the operating frequency has led numerical techniques towards full-wave analysis, because the effects such as coupling due to surface waves and discontinuities become significant at high frequencies. Denlinger [7] solved the integral equation using the Fourier transform technique. The solution by his method, however, is strongly dependent on the assumed current distribution on the strip. To avoid this difficulty and to allow systematic improvements of the solution for the current components to a desired degree of accuracy, Itoh and Mittra [8] introduced the Spectral Domain Approach, otherwise called the Spectral Domain Method in 1973 for open microstrip lines and enhanced for shielded microstrip lines in Ref. [9]. The unknown current distribution on the strip is defined as a set of known current basis functions with unknown coefficients and Galerkin's Procedure is used to yield a homogeneous system of equations to determine the propagation constant and the amplitude of current distribution.

The application of the method to a general three-dimensional structure required the introduction of a more advanced formulation. The move in this direction is presented by Itoh [10] with the modelling of a microstrip resonator. In 1980, Itoh introduced a formulation process called the immitance approach [11] for easy formulation of multilayer structures. This approach is based on the transverse equivalent circuit concept as applied in the spectral domain. An application of the immitance approach was presented by Itoh and Menzel [12] which allowed the analysis of planar resonator antennas for the complex resonant

frequency. The historical progress of the Spectral Domain Method (SDM) until 1984 was summarized by Jansen [13].

A full-wave analysis of microstrip open-end and gap discontinuities in SDM was presented by Jackson [14]. In his analysis, the results were presented for the reflection and transmission coefficients derived with a source formulation of SDM. A time-harmonic electromagnetic field analysis method related to SDM, which allows the modelling of shielded arbitrary shaped planar discontinuities, was presented by Rautio and Harrington [15]. Similarly, Jackson [16] formulated SDM for open irregular microwave circuit discontinuities with semi infinite feedlines, although Jackson calls his technique as Finite Element Method.

A generalised spectral domain Green's function was described by Das and Pozar [17] in terms of suitable components of the vector electric and magnetic potentials. The works by Railton et al who described the asymptotic form of the Green's function in Ref. [18] and re-arranged the characteristic equation by making use of its asymptotic properties in 1992 in Ref. [19] are found out by the author to be particularly significant. Precalculated current basis functions have been used by Meade [20] to reduce the number of current basis functions required and to include a priori knowledge of the current distribution at discontinuities. It can be noticed that the historical development of SDM has been summarized until 1994 which is the starting date of this project. The existing SDM has been improved by the enhancements introduced in this project. Meanwhile Tsai et al [21] has developed the Mixed Potential Integral Equation Method (MPIEM) in the spectral domain and Kuo et al [22] has formulated the hybrid-mode Spectral Domain Approach to investigate the dispersion nature of multiple coupled microstrip lines with arbitrary metallization thickness. General analytical solutions of static Green's functions for shielded and arbitrarily opened multilayered media have been presented by Li et al in 1997.

Numerous authors have contributed to the development of SDM. Balik introduced compensation function to remedy the deficiency, presented in Ref. [16,] and to model correctly the excitation over the whole microwave frequency region. Kaichida et al., Ref. [23], introduced a novel technique in SDM to evaluate the loss in the transmission line at millimeter wave frequency.

3.1 General formulation

A general formulation of the Spectral Domain Method for the passive, open planar microwave circuits of interest is presented. The following is based on references quoted above. Before formulating this process, the types of equations obtained by SDM and those obtained by a typical space domain formulation are compared. In the space domain, a microwave circuit can be analyzed by solving the coupled integral equation given as:

$$\begin{aligned}\iint\left[G_{zz}(x-x',y,z-z')J_z(x',z')+G_{zx}(x-x',y,z-z')J_x(x',z')\right]dx'\,dz' &= E_z(x,z)\\ \iint\left[G_{xz}(x-x',y,z-z')J_z(x',z')+G_{xx}(x-x',y,z-z')J_x(x',z')\right]dx'\,dz' &= E_x(x,z)\end{aligned} \tag{1}$$

where:

G_{st} is the dyadic Green's function (dependent on frequency, ω),

E_s is the electric field at the air-dielectric interface,

Js is the current distribution on the metallization.

Equation (1) can be solved if G_{st}, etc., are given. The Green's functions G_{zz}, etc., however, are not available in closed form for the nonhomogeneous structures. The following algebraic Eq. (2), instead of the coupled integral Eq. (1), are obtained in the spectral domain formulation. These equations are Fourier transforms of the coupled integral Eq. (1):

$$\begin{aligned}\boldsymbol{G}_{zz}(k_x,d,k_z,w)\boldsymbol{J}_z(k_x,k_z)+\boldsymbol{G}_{zx}(k_x,d,k_z,w)\boldsymbol{J}_x(k_x,k_z)&=\boldsymbol{E}_z\,(k_x,d,k_z)\\ \boldsymbol{G}_{xz}(k_x,d,k_z,w)\boldsymbol{J}_z(k_x,k_z)+\boldsymbol{G}_{xx}(k_x,d,k_z,w)\boldsymbol{J}_x(k_x,k_z)&=\boldsymbol{E}_x(k_x\ d,k_z)\end{aligned} \tag{2}$$

where the bold quantities are the Fourier transform of corresponding quantities and d is the thickness of the dielectric substrate. The two dimensional Fourier transform is defined as:

$$\phi(k_x,k_z)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\phi(x,z)e^{j(k_x x+k_z z)}dx\,dz \tag{3}$$

Equation (2) can also be written in a matrix form as follows:

$$\begin{bmatrix}\boldsymbol{G}_{zz} & \boldsymbol{G}_{zx}\\ \boldsymbol{G}_{xz} & \boldsymbol{G}_{xx}\end{bmatrix}\begin{bmatrix}\boldsymbol{J}_z\\ \boldsymbol{J}_x\end{bmatrix}=\begin{bmatrix}\boldsymbol{E}_z\\ \boldsymbol{E}_x\end{bmatrix} \tag{4}$$

where:

$\boldsymbol{G}(k_x,d,k_z,\omega)$ is the dyadic Green's function in the spectral domain,

$\boldsymbol{J}_z$ is the Fourier transform of the *z*-directed surface current density,

$\boldsymbol{J}_x$ is the Fourier transform of the *x*-directed surface current density,

$\boldsymbol{E}$ is the Fourier transform of the electric field.

Spectral Fourier transforms have been taken in the *x* and *z* directions, therefore the dielectric must extend to infinity in both the *x* and *z* directions. This is a disadvantage of the SDM formulation, but a layered dielectric structure is considered to be general enough to fit a wide variety of applications.

Derivations of the spectral domain forms of the Green's functions ($\boldsymbol{G}_{zz}$, etc. in Eqs. (2) and (4)) are widely available in the literature for single dielectric layer [8, 9].

Equations (2) and (4) contain four unknowns which are $\boldsymbol{J}_z$, $\boldsymbol{J}_x$, $\boldsymbol{E}_z$ and $\boldsymbol{E}_x$. At least two unknowns must be eliminated to solve the equations. The Method of Moments [24] is applied to eliminate the Fourier transformed electric fields from the formulation. The first step to the solution is to expand the unknown surface current as a set of known basis functions with unknown coefficients:

$$\boldsymbol{J}_s(k_x,k_z)=\sum_{n=1}^{N} a_{sn}\boldsymbol{J}_{sn}(k_x,k_z) \quad s=x,z \qquad (5)$$

where $\boldsymbol{J}_{sn}$ is the n^{th} basis function with coefficient a_{sn}.

The basis functions ($\boldsymbol{J}_{sn}$) must be chosen to approximate the unknown current distribution on the metallization of the circuit. The choice of basis functions is crucial to the efficiency of the technique. If they are not chosen to represent the actual current distribution closely, then a large number of functions will be required for convergence.

To eliminate the Fourier transformed electric fields ($\boldsymbol{E}_z$, $\boldsymbol{E}_x$ in Eq. (4)), a set of weighting functions (w_t) are required. The weighting functions (w_t) are chosen to be identical to the set of current basis functions. Two more weighting functions are also required for the problem with excitation.

After application of the Method of Moments the Eq. (4) becomes:

$$\begin{bmatrix} \boldsymbol{Z}_{zz} & \boldsymbol{Z}_{zx} \\ \boldsymbol{Z}_{xz} & \boldsymbol{Z}_{xx} \end{bmatrix} \begin{bmatrix} a_z \\ a_x \end{bmatrix} = \begin{bmatrix} \boldsymbol{V}_z \\ \boldsymbol{V}_x \end{bmatrix} \qquad (6)$$

where the elements of the impedance matrix $\boldsymbol{Z}$ are given by:

$$\boldsymbol{Z}_{st} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \boldsymbol{w}_t(k_x,k_z)\boldsymbol{G}(k_x,k_z,d,\omega)\boldsymbol{J}_s(k_x,k_z)dk_x dk_z$$
$$s,t=x,z \qquad (7)$$

and the excitation vector is given by:

$$\boldsymbol{V}_t = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \boldsymbol{w}_t(k_x,k_z)\boldsymbol{E}_t(k_x,k_z)dk_x dk_z \quad t=x,z \qquad (8)$$

By solving Eq. (6) for the unknown current coefficients, the unknown current distribution of the metallization of the circuit can be determined. For the sourceless eigenvalue problem, the excitation vector ($\boldsymbol{V}_t$) is identically zero. This can be verified by using Parseval's theorem, see page 341 in Ref. [25].

Note that the method is formulated for spot frequencies, that is the final matrix equation is repeatedly solved for each frequency of interest. Thus the minimization of the order of the matrix equation is very important for the efficiency of the technique. For the Method of Moments, the order of the matrix is twice the total number of basis functions plus two, therefore the critical choice of the basis functions is highlighted.

4. FUNCTIONAL APPROACH TO SDM

4.1 Introduction

In this paper, five fundemantal modules are rewritten by using functional approach instead of conventional programming language such as Pascal to show the applicability of the approach. All of the modules are described in the sections below.

4.2 Input functions

In this module the input parameters are taken and passed to other modules. Operating frequency, substrate layer parameters, k_x, k_z which are Fourier transform variables in *x* and *z* directions respectively, *n* which is number of layers, l_x and l_z which are dimensions of rooftop function are used as input values. The rewritten module becomes as follows:

```
type Ind = Double
type D = Double
type M = Double
type E = Double
type Layer = [(Ind,D,M,E)
type OneLayer = (Ind,D,M,E)
type Kx = Complex Double
type Kz = Complex Double
type Lx = Double
w :: Double
w = 2*3.1456*saveF
saveLayer :: IO - Layer
saveKx :: IO - Kx
saveKz :: IO - Kz
saveLx :: IO - Lx
saveN :: IO - Double
saveF :: IO - Double
```

```
nx=saveKx/sgrt(saveKx*saveKx+saveKz*saveKz)
nz=saveKz/sgrt(saveKx*saveKx+saveKz*saveKz)
```

4.3 Impedance functions

In this module the Green Function (given in Eq. (2)) in the spectral domain has been computed by using a functional approach. Mathematical formulation of the Green function can be found in the literature such as Ref. [24, Chapter 4].

```
findlayer :: Ind -> OneLayer
findlayer indx = head [(ind,d,m,e) | (ind,d,m,e) <-
saveLayer , ind==indx]
layparD :: OneLayer -> D
layparD (ind,d,m,e) = d
layparM :: OneLayer -> M
layparM (ind,d,m,e) = m
layparE :: OneLayer -> E
layparE (ind,d,m,e) = e
ztm :: Double -> Complex Double
ztm i = (gama i)/(w*(layparE(findlayer i)):+0)
gama :: Double -> Complex Double
gama i = sgrt((saveKz*saveKz)+(saveKx*saveKx)-
((w*w*(layparM (findlayer i))*(layparE(findlayer
i))):+0))
zte :: Double -> Complex Double
zte i = (w*(layparM(findlayer i)):+0) /gama i
zelist :: Double -> (Complex Double,Double)
zelist n = ((zeN n) ,(n-1) )
zeN :: Double -> Complex Double
zeN n = ztm n / atanh((gama n)*((layparD
(findlayer n)):+0))
zhN :: Double -> Complex Double
zhN n = zte n / atanh((gama n)*((layparD
(findlayer n)):+0))
zhlist :: Double -> (Complex Double,Double)
zhlist n = ((zhN n) ,(n-1) )
coth :: Double -> Complex Double
coth i = atanh((gama i)*((layparD (findlayer
i)):+0))
ze2 :: (Complex Double,Double) -> (Complex
Double,Double)
ze2 (n,2) = (n,2)
ze2 (n,s) =
ze2
(
(
(ztm s * (n*(coth s ))+ztm s)/
(ztm s *(coth s)+n)
)
, (s-1)
)
zh2 :: (Complex Double,Double) -> (Complex
Double,Double)
zh2 (n,2) = (n,2)
zh2 (n,s) =
zh2
(
(
(zte s * (n*(coth s ))+zte s)/
(zte s *(coth s)+n)
)
, (s-1)
)
ze1 = ztm 1
zh1 = zte 1
ze :: Double -> Complex Double
ze n =
1/
(
((1:+0)/ze1)
+
((1:+0)/fst((ze2 (zelist n))))
)
zh :: Double -> Complex Double
zh n=
1/
(
((1:+0)/zh1)
+
((1:+0)/fst((zh2 (zelist n))))
)
gzz n = nz*nz*(ze n) + nx*nx*(zh n)
gzx n = nx*nz*(-(ze n)+(zh n))
gxz n = gzx n
gxx n = nx*nx*(ze n) + nz*nz*(zh n)
```

4.4 Current functions

In this module current basis functions, which are roottop functions [24, Chapter 3], are computed by a functional approach.

```
jz :: Double ->Complex Double
jz n = (2/saveKx)* sin(saveKx*(saveLx:+0))*
exp(saveKx*(n*(saveLx):+0))
jx :: Double -> Complex Double
jx n = (2/(saveKx*saveKx))*(1-cos(saveKx*
(saveLx:+10)*exp(saveKx*(n*(saveLx):+0))
makeNpar :: Double -> Double -> [Double]
makeNpar 0 n = []
makeNpar n a =(-((n-1)-a)):(makeNpar (n-1) a)
jzn :: Double -> Double -> [Complex Double]
jzn n a = map jz (makeNpar n a)
jxn :: Double -> Double -> [Complex Double]
jxn n a = map jx (makeNpar n a)
mux :: [Complex Double] -> [(Complex Double)]
mux xs = concat (map (fun xs) xs)
fun :: [Complex Double] ->Complex Double->
[(Complex Double)]
fun as a = [(a*b)I b<-as]
```

4.5 Integral functions

This module is used to calculate the each element of the impedance matrix given in Eq. (7).

```
makeMat :: Integer -> Integer-> Double ->
[(Complex Double)] -> [(Complex Double)]
makeMat a c k [] = []
makeMat a c k (n:ns)
  |(a>O &&a<=(1*c)) =(n*(gzz k)):(makeMat
(a+1) c k ns)
  |(a>(1*c)&& a<=(2*c))  (n*(gzx k)):(makeMat
(a+1) c k ns)
  |(a>(2*c)&& a<=(3*c))  (n*(gzz k)):(makeMat
(a+1) c k ns)
  |(a>(3*c)&& a<=(4*c))  (n*(gzx k)):(makeMat
(a+1) c k ns)
  |(a>(4*c)&& a<=(5*c))  (n*(gxz k)):(makeMat
(a+1) c k ns)
  |(a>(5*c)&& a<=(6*c))  (n*(gxx k)):(makeMat
(a+1) c k ns)
  |(a>(6*c)&& a<=(7*c))  (n*(gxz k)):(makeMat
(a+1) c k ns)
  |(a>(7*c)&& a<=(8*c))  (n*(gxx k)):(makeMat
(a+1) c k ns)
```

5. NUMERICAL RESULTS

These sections below include several analyzed example microwave structures to show the accuracy of the re-written program using Spectral Domain Method in Haskell which is one of the functional programming languages. Total program code has been optimized *55%* compared to Pascal code. As a result runtime has been reduced *50%* compared to Pascal code run on the computer. The computer has Intel P4 *2.4 GHz* with 1 GB RD RAM. The operating system is Redhat Linux *8.0*.

5.1 Simple low-pass filter

Measurement results are available for the microstrip low-pass filter [26] shown in Figure 1. The dimensions and parameters of the dielectric substrate are given in Figure 1.

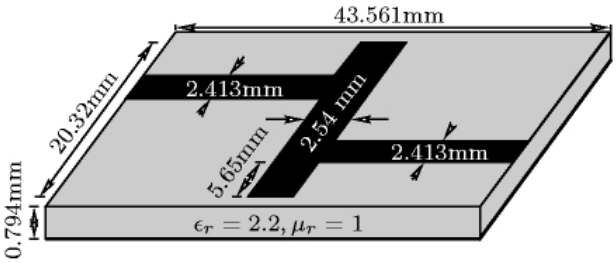

Fig. 1 Low pass filter detail

The S-parameter results are plotted in Figure 2 where it can be seen that the computed results and measurements are in a very good agreement.

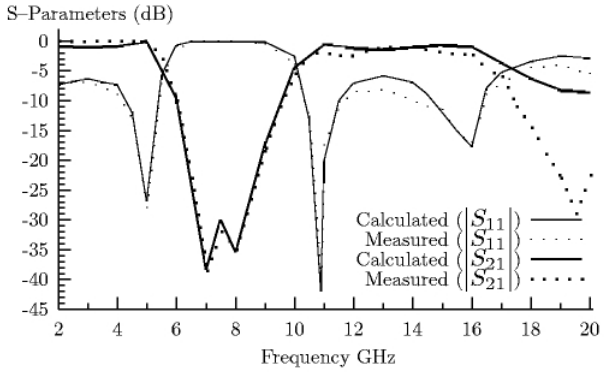

Fig. 2 Plot of S-parameters' magnitude for the low-pass filter

5.2 Edge-coupled filter

In order to further prove the accuracy of the rewritten program, the analysis of the microstrip edge-coupled filter, shown in Figure 6 in Ref. [19], is considered. The measurements were performed by Shibata et al [27] for this filter.

As it can be seen in Figure 3, there is a clear agreement between the newly written program and measured data.

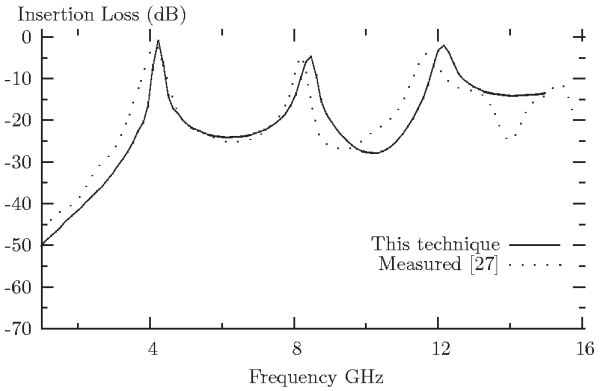

Fig. 3 Magnitude of S-parameters for the edge-coupled filter

6. CONCLUSION

We have shown that realistically complex microstrip circuits can be rigorously analyzed by rewritten program which uses a functional programming approach to Spectral Domain Method. The accuracy of the program is obvious. The code size and runtime reduction are *55%* and *50%* recpectively on ordinary computers. By this approach a model which retains the accuracy of the full-wave anaylsis technique as well as the speed of the package programs has been introduced.

7. REFERENCES

[1] A. Gharsallah, A. Mami, R. Douma, A. Gharbi and H. Baudrand, Analysis of microstrip antenna with fractal multilayer substrate using iterative method, *International Journal of RF and Microwave Computer-Aided Engineering*, Vol. 11, pp. 212-218, 2001.

[2] T. Fukusako and M. Tsutsumi, Microstrip superconducting resonators loaded with yttrium iron garnet single crystals, *Eletronics and Communication in Japan*, Vol. 81, No. 5, pp. 44-50, 1998.

[3] H.H. Balik and C.J. Railton, New compensation functions for efficient excitation of open planar microwave circuits in SDM, *IEEE Transaction on Microwave Theory and Technique*, Vol. 47, pp. 106-108, January 1999.

[4] S. Thomson, *The Craft of Functional Programming*, 2nd Edition, Addison Wesley Press, 1999.

[5] I. Holyer, *Functional Programming with Haskel*, Bristol University Press, 1996.

[6] E. Yamashita and R. Mittra, Variational method for the analysis of microstrip lines, *IEEE Transaction on Microwave Theory and Technique*, Vol. 16, pp. 251-255, April 1968.

[7] E.J. Denlinger, A frequency dependent solution for microstrip transmission lines, *IEEE Transaction on Microwave Theory and Technique*, Vol. 19, pp. 30-39, January 1971.

[8] T. Itoh and R. Mittra, Spectral domain approach for calculating the dispersion characteristics of microstrip lines, *IEEE Transaction on Microwave Theory and Technique*, pp. 496-499, July 1973.

[9] T. Itoh and R. Mittra, A technique for computing dispersion characteristics of shielded microstrip lines, *IEEE Transaction on Microwave Theory and Technique*, pp. 896-898, October 1974.

[10] T. Itoh, Analysis of microstrip resonators, *IEEE Transaction on Microwave Theory and Technique*, Vol. 22, pp. 946-952, November 1974.

[11] T. Itoh, Spectral domain immitance approach for dispersion characteristics of generalized printed transmission lines, *IEEE Transaction on Microwave Theory and Technique*, Vol. 28, pp. 733-736, July 1980.

[12] T. Itoh and W. Menzel, A full-wave analysis method for open microstrip structures, *IEEE Transactions on Antennas and Propagation*, Vol. 29, pp. 63-67, January 1981.

[13] R.H. Jansen, The spectral domain approach for microwave integrated circuits, *IEEE Transaction on Microwave Theory and Technique*, Vol. 33, pp. 1043-1056, October 1985.

[14] R.W. Jackson and D.M. Pozar, Full-wave analysis of microstrip open-end and gap discontinuities, *IEEE Transaction on Microwave Theory and Technique*, Vol. 33, pp. 1036-1042, October 1985.

[15] J.C. Rautio and R.F. Harrington, An electromagnetic time harmonic analysis of shielded microstrip circuits, *IEEE Transaction on Microwave Theory and Technique*, Vol. 35, pp. 726-729, August 1987.

[16] R.W. Jackson, Full-wave, finite element analysis of irregular microstrip discontinuities, *IEEE Transaction on Microwave Theory and Technique*, Vol. 37, pp. 81-89, January 1989.

[17] N.K. Das and D.M. Pozar, A generalised spectral domain Green's function for multilayer dielectric substrates with application to multilayer transmission lines, *IEEE Transaction on Microwave Theory and Technique*, Vol. 35, pp. 326-335, March 1987.

[18] C.J. Railton and T. Rozzi, Complex modes in boxed microstrip, *IEEE Transaction on Microwave Theory and Technique*, Vol. 36, pp. 865-873, May 1988.

[19] C.J. Railton and S.A. Meade, Fast rigorous analysis of shielded planar filters, *IEEE Transaction on Microwave Theory and Technique*, Vol. 40, pp. 978-985, May 1992.

[20] S.A. Meade and C.J. Railton, Efficient implementation of the spectral domain method including precalculated corner basis functions, *IEEE Transaction on Microwave Theory and Technique*, Vol. 42, pp. 1678-1684, September 1994.

[21] M.-J. Tsai, D. Flaviis, O. Fordham, and N.G. Alexopoulos, Modelling planar arbitrary shaped microstrip elements in multilayered media, *IEEE Transaction on Microwave Theory and Technique*, Vol. 45, pp. 330-337, March 1997.

[22] J. Kuo and T. Itoh, Hybrid-mode computation of propagation and attenuation characteristic of parallel coupled microstrips with finite metalisation thickness, *IEEE Transaction on Microwave Theory and Technique*, Vol. 45, pp. 274-280, February 1997.

[23] T. Kaichida, N. Ishii, M. Yamamoto, T. Nishimura and K. Itoh, Analysis of dual frequency operating microstrip antenna fed by coplanar waveguide using method of moments in spectral communications in Japan, Vol. 84, No. 3, pp. 21-28, 2001.

[24] H.H. Balik, Passive open planar circuit analysis by enhanced spectral domain method, PhD. Thesis, University of Bristol, December 1997.

[25] T. Itoh, *Numerical Techniques for Microwave and Millimeter-Wave Passive Structures*, John Willey and Sons, 1989.

[26] D.M. Sheen, S.M. All, M.D. Abdouzahra and J.A. Kong, Application of the three-dimensional finite-difference time-domain method of the analysis of planar microstrip circuits, *IEEE Transaction on Microwave Theory and Technique*, Vol. 38, pp. 849-857, July 1990.

[27] T. Shibata, T. Hayashi and T. Kimura, Analysis of microstrip circuits using three-dimensional full-wave electromagnetic field analysis in the time domain, *IEEE Transaction on Microwave Theory and Technique*, Vol. 36, pp. 1064-1070, June 1988.

NOVI PRISTUP METODI PRIJENOSNOG PODRUČJA: FUNKCIONALNO PROGRAMIRANJE

SAŽETAK

Metoda prijenosnog područja predstavlja snažnu tehniku za analizu ravninskih mikrovalnih strujnih krugova. Dostupni konvencionalni jezici programiranja koji se koriste u literaturi ne omogućavaju dovoljno brzo korištenje Metode prijenosnog područja da se razvije analiza programskog paketa. Funkcionalni pristup Metodi prijenosnog područja omogućava visoku razinu programiranja kao i višestruke mogućnosti koje pomažu da se izgrade elegantne, dapače jake i općenite knjižnice funkcija.

***Ključne riječi**: SDM, ravninski strujni krugovi, funkcionalno programiranje, Haskel, antene.*