An Efficient Parallel MoM to Analyze Microstrip Structures *

F. Cabrera$^{(1)}$, *C.N. Ojeda-Guerra*$^{(2)}$, *E. Jiménez*$^{(1)}$, *J.G. Cuevas del Río*$^{(3)}$,
E.M. Macías-López$^{(2)}$, *A. Suárez*$^{(2)}$

$^{(1)}$Dpto. de Señales y Comunicaciones (GIC)
$^{(2)}$Dpto. de Ingeniería Telemática (GAC)
Campus de Tafira S/N, Las Palmas 35017, Spain.
$^{(3)}$E.T.S.I. Telecomunicación, U.P.M.
francis@cibeles.teleco.ulpgc.es, eugenio@cibeles.teleco.ulpgc.es

Abstract

The increasing use of microstrip technology require more accurate analysis methods like full wave method of moments. However, this involves a great computational effort. To reduce the computation time, an alternative parallel method to analyze irregular microstrip structures is presented in this paper. This method calculates the unknown surface current on the planar structure trough a irregular rectangular division using basis and weighted functions. The parallel algorithm performs the calculus of a $[Z]$ matrix and then solves the system using current densities as the unknowns. This parallel program was implemented in the IBM-SP2 using MPI library.

1 Introduction

The moment method is a numerical method to compute the current distribution of any arbitrary microstrip structure. This is accomplished transforming the EFIE 'Electrical Field Integral Equation' (1) into a linear algebraic equations system.

$$\overline{E}^{i} + j\omega\mu_0 L(\bar{J}_s) = 0 \tag{1}$$

Here *L* is a known linear (integral) operator, $\overline{E}^{i}$ is the incident field in the boundary and $\bar{J}_s$ is the surface current density to compute. The method of the moments solves this equation dividing the structure into M different entities called *patches*. Whichever it is the planar structure microstrip to analyze, it always can be splitted in non-uniforms cells following a vertical and horizontal pattern. With these cells or segments, it is formed an entity called *patch* composed by 4 cells called *subdomains*. Each subdomain is defined by 4 vertices with its coordenates (x,y) (figure 1).

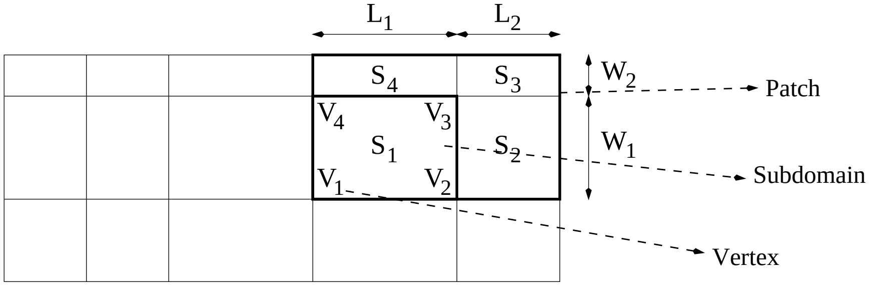

Figure 1: Discretization of the microstrip structure.

[*] This work has been partially supported by the spanish CICYT under project TIC 99-1172-C02-02 and by the european comunity FEDER initative under project 1FD97-1183-C02-02

The current distribution is expanded as a linear combination of two-dimensional *basis* functions which are defined over each subdomain (2).

$$\overline{J}_s = \sum_{i=1}^{M} I_{x_i} J_{x_i}(x',y')\hat{x} + \sum_{i=1}^{M} I_{y_i} J_{y_i}(x',y')\hat{y} \tag{2}$$

The *basis* function can be decomposed into two indepent variable functions $J_{x_i}(x',y') = T_i(x')Q_i(y')$. The longitudinal component has a piecewise sinusoidal or triangular behaviour (3) while the transversal component has a constant distribution (4). This function is an arch shape one like showed in the figure 2. The descomposition of $J_{y_i}(x',y')$ is similar.

$$T_i(x') = \begin{cases} \dfrac{\sin[k_e(l_{1i}+x'-x_i)]}{\sin(k_e l_{1i})} & \text{if } x_i - l_{1i} \le x' \le x_i \\ \dfrac{\sin[k_e(l_{2i}-x'+x_i)]}{\sin(k_e l_{2i})} & \text{if } x_i \le x' \le x_i + l_{2i} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

$$Q_i(y') = \begin{cases} \dfrac{1}{w_{1i}+w_{2i}} & \text{if } y_i - w_{1i} \le y' \le y_i + w_{2i} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Here k_e is a propagation effective constant which value is usually taken as $k_e = \frac{2\pi}{\lambda_0\sqrt{\frac{\epsilon_r+1}{2}}}$.

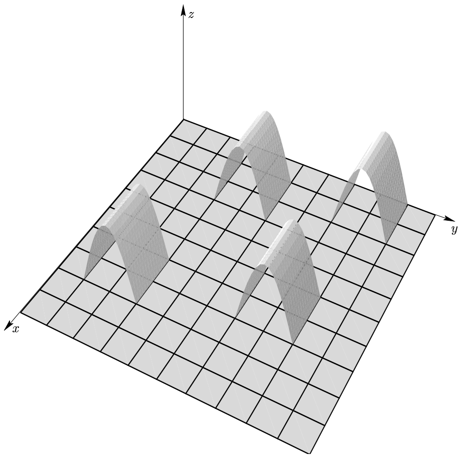

Figure 2: Several arch shapes in the structure.

The EFIE has two scalar functional equations that have M unknowns each one. These equations are not enough to determine the $2M$ unknown constants $[I_{x_i}, I_{y_i}]_{i=1,2,\ldots,M}$. In order to solve the $2M$ constants, it is necessary to compute $2M$ linearly independent equations. A way to solve this system is using a Galerkin's method [1]. This method called *weighted residuals* method minimize the residual error (5).

$$\triangle E_t = \overline{E}_t^i + \overline{E}_t^r \neq 0 \tag{5}$$

To minimize this residual, the method of weighted residuals is used in conjunction with the Hilbert inner product defined as:

$$\big\langle w(x,y), g(x',y') \big\rangle = \iint_S w(x,y) g(x',y')^* \, ds \tag{6}$$

The *basis* $g(x',y')$ and *weight* $w(x,y)$ functions are the same in this method. The final system of equations has $2M$ equations and 2M unknowns, where $1 \leq i,j \leq M$ and can be expressed:

$$\begin{pmatrix} Z_{xx}^{ij} & Z_{xy}^{ij} \\ Z_{yx}^{ij} & Z_{yy}^{ij} \end{pmatrix}_{2M\times 2M} \begin{pmatrix} I_x^i \\ I_y^i \end{pmatrix}_{2M\times 1} = \begin{pmatrix} V_x^j \\ V_y^j \end{pmatrix}_{2M\times 1} \qquad \begin{matrix} i = 1..M \\ j = 1..M \end{matrix} \tag{7}$$

The elements of the matrix Z_{xx}^{ij} and Z_{yy}^{ij} are the autoimpedances due to the couplings between the same components of the current densities. Z_{xy}^{ij} and Z_{yx}^{ij} are the couplings due to distinct components. The elements of the matrix $[Z]$ are composed by the sum of two quadruple integrals (integral 1 and integral 2) which general expression is collected in (8).

$$\int_{x_i-l_{1i}}^{x_j+l_{2j}} dx' \int_{x_j-l_{1j}}^{x_j+l_{2j}} dx \int_{y_i-w_{1i}}^{y_i+w_{2i}} dy' \int_{y_j-w_{1i}}^{y_j+w_{2j}} F_i(x',y')\; F_j(x,y)\; G_\alpha(x-x',y-y')dy \tag{8}$$

where G_α are Sommerfeld Integrals. The expression and computation of these integrals are collected in [4], where $F_i(x',y')$ and $F_j(x,y)$ are *basis* and *weight* functions. The table 1 shows the value of these functions for each integral (defined over rectangular shapes).

	Integral 1		Integral 2	
	$F_i(x',y')$	$F_j(x,y)$	$F_i(x',y')$	$F_j(x,y)$
Z_{xx}^{ij}	$T_i(x')Q_i(y')$	$T_j(x)Q_j(y)$	$\frac{\partial T_i(x')}{\partial x'}Q_i(y')$	$\frac{\partial T_j(x)}{\partial x}Q_j(y)$
Z_{xy}^{ij}	$-$	$-$	$Q_i(x')\frac{\partial T_i(y')}{\partial y'}$	$\frac{\partial T_j(x)}{\partial x}Q_j(y)$
Z_{yx}^{ij}	$-$	$-$	$\frac{\partial T_i(x')}{\partial x'}Q_i(y')$	$Q_j(x)\frac{\partial T_j(y)}{\partial y}$
Z_{yy}^{ij}	$Q_i(x')T_i(y')$	$Q_j(x)T_j(y)$	$Q_i(x')\frac{\partial T_i(y')}{\partial y'}$	$Q_j(x)\frac{\partial T_j(y)}{\partial y}$

Table 1: $[Z]$ functions.

Since the functions F_i and F_j are defined within patches, the coupling between the patch i and j has to be computed by parts [2]. This computation will be accomplished through the sum of the coupling of each one of the subdomains the patch i is divided into, with each one of the subdomains of the patch j.

2 Numerical Integration

Each term of $[Z]$ can be divided into 16 quadruple integrals and computed directly through a Gauss-Laguerre quadrature but their cost would be large. In order to minimize this cost, a domain change transforming (8) to a double sum of double integrals is accomplished. Using the new transformation (9) and other similar for the domain (y,y'), the computation time reduces considerably.

$$u = \frac{1}{\sqrt{2}}(x-x') \qquad\qquad v = \frac{1}{\sqrt{2}}(x+x') \tag{9}$$

Each quadruple integral defined by parts is transformed in the sum of nine two-fold integrals. The total integral has the following expression:

$$\begin{aligned} &\int_{x_i-l_{1i}}^{x_j+l_{2j}} dx' \int_{x_j-l_{1j}}^{x_j+l_{2j}} dx \int_{y_i-w_{1i}}^{y_i+w_{2i}} dy' \int_{y_j-w_{1j}}^{y_j+w_{2j}} F_i(x',y')\; F_j(x,y)\; G_\alpha(x-x',y-y')dy \\ &= \sum_{L=1}^{12}\sum_{K=1}^{12} \int_{ULI_L}^{ULS_L} du \int_{VLI_K}^{VLS_K} f_{TT_L}(u)\cdot f_{TT_K}(v)\cdot G_\alpha(\sqrt{2}u + x_j - x_i, \sqrt{2}v + y_j - y_i)dv \end{aligned} \tag{10}$$

index	lower limit v VLI	upper limit v VLS	lower limit u ULI	upper limit u ULS
1	$\frac{-w_{1j}}{\sqrt{2}}$	$\frac{-(w_{1j}-w_{1i})}{\sqrt{2}}$	$\frac{-l_{1j}}{\sqrt{2}}$	$\frac{-(l_{1j}-l_{1i})}{\sqrt{2}}$
2	$\frac{-(w_{1j}-w_{1i})}{\sqrt{2}}$	0	$\frac{-(l_{1j}-l_{1i})}{\sqrt{2}}$	0
3	0	$\frac{w_{1i}}{\sqrt{2}}$	0	$\frac{l_{1i}}{\sqrt{2}}$
4	0	$\frac{w_{1i}}{\sqrt{2}}$	0	$\frac{l_{1i}}{\sqrt{2}}$
5	$\frac{w_{1i}}{\sqrt{2}}$	$\frac{w_{2j}}{\sqrt{2}}$	$\frac{l_{1i}}{\sqrt{2}}$	$\frac{l_{2j}}{\sqrt{2}}$
6	$\frac{w_{2j}}{\sqrt{2}}$	$\frac{w_{2j}+w_{2i}}{\sqrt{2}}$	$\frac{l_{2j}}{\sqrt{2}}$	$\frac{l_{2j}+l_{1i}}{\sqrt{2}}$
7	$\frac{-w_{2i}}{\sqrt{2}}$	0	$\frac{-l_{2i}}{\sqrt{2}}$	0
8	0	$\frac{w_{2j}-w_{1i}}{\sqrt{2}}$	0	$\frac{l_{2j}-l_{2i}}{\sqrt{2}}$
9	$\frac{w_{2j}-w_{2i}}{\sqrt{2}}$	$\frac{w_{2j}}{\sqrt{2}}$	$\frac{l_{2j}-l_{2i}}{\sqrt{2}}$	$\frac{l_{2j}}{\sqrt{2}}$
10	$\frac{-(w_{1j}+w_{2i})}{\sqrt{2}}$	$\frac{-w_{1i}}{\sqrt{2}}$	$\frac{-(l_{1j}+l_{2i})}{\sqrt{2}}$	$\frac{-l_{1j}}{\sqrt{2}}$
11	$\frac{-w_{1j}}{\sqrt{2}}$	$\frac{-w_{2i}}{\sqrt{2}}$	$\frac{-l_{1j}}{\sqrt{2}}$	$\frac{-l_{2i}}{\sqrt{2}}$
12	$\frac{-w_{2i}}{\sqrt{2}}$	0	$\frac{-l_{2i}}{\sqrt{2}}$	0

Table 2: Integral limits.

where the integral limits for each one of the integrals of both sums are collected in table 2. When the expression (10) has been obtained and its limits calculated, the elements of the $[Z]$ matrix are calculated numerically. The functions $f_{TT}(u), f_{TT}(v)$ are the $basis$ and $weight$ functions in the transformed domain.

In the transformed domain, there are 144 double integrals. If the time to compute a single integral by a 8 points Gauss-Laguerre quadrature is 8 time units, we can see in table 3 the computed time for an element of the $[Z]$ matrix.

Quadrature	Normal Method	Transformed Domain
8 pts.	65.536 u.t.	9.216 u.t.
32 pts.	16.777.216 u.t.	147.456 u.t.

Table 3: Computed Time.

The final system can be divided into 4 submatrices $M \times M$. The submatrices $[Z_{xx}]$ and $[Z_{yy}]$ compute $2M^2$ integrals and $[Z_{xy}]$ and $[Z_{yx}]$ compute M^2, so the sum of integrals is 6 M^2. The terms V_x^j and V_y^j are obtained using a delta-gap model. This model assumes that the voltage source V_m^t is within an infinitesimally small gap and across the extended ground plane. For this model, a physical port is used within a $patch$ o several $patchs$. The impressed field originated is:

$$\overline{E}^i(\bar{\rho}) = V_m^t \delta(\bar{\rho} - \bar{\rho}_m)\hat{n}_m \tag{11}$$

where $\bar{\rho}_m$ is the location of the port and $\hat{n}_m$ is a normal vector parallel to the feed-line. This voltage source originates induced currents which are modeled through half-basis function located at $\bar{\rho}_m$. Then, the $patchs$ having ports are composed of 2 physical rectangular subdomains and 2 of zero length. The choice of subdomains will depend on the position of the excitation.

The V_j coefficients are normalized to the unity in these $patchs$ and zero on the rest [3]. The total result is the linear superposition of each one of the excitations. It has supossed that a port is modeled through the discretation of a $patch$. However, the structure can be splitted in several $patches$ in function of the layout, the physical port can be discretized into various segments so several logic ports are conected in parallel to model a physical port.

3 Parallelization of the moment method

The moment method can be parallelized in two ways: computation of the $[Z]$ matrix and solving the system of equations. In this section, we discuss the parallelization of the computation of $[Z]$ and the computation of the LU factorization, involved in the linear system of equations. So:

$$[Z] \times [I] = [V] \qquad\qquad [L] \times [U] \times [I] = [V] \Longrightarrow [L] \times [Y] = [V], [Y] = [U] \times [I]$$

3.1 Computation of the [Z] matrix

In order to compute the $[Z]$ matrix , we need to analyze the microstrip structure. The shape of this structure is stored in a geometric file. This file has to be read in order to stablish the values of the dimensions and build a vector (this vector has a $2M$ dimension), which is used by the generation of the couplings among the patches of the structure. This computation involves the double sum of double integrals explained before. With this information, we compute the values of the complex matrix $[Z]$. The computation of each element of the matrix is independent, being only function of the couplings vector. The code, which computes these elements, is in table 4.

```
do j=1,N/2
    do i=1,N/2
        call coupling(coup,j,i,dimp(1,j),dimp(1,i),l0)
        Z(j,i)=(coup(1)+coup(2))/c4
        Z(j,i+N/2)=coup(3)/c4
        Z(j+N/2,i)=coup(4)/c4
        Z(j+N/2,i+N/2)=(coup(5)+coup(6))/c4
    end do
end do
```

Table 4: Algorithm to compute the $[Z]$ matrix ($M = N/2$).

The algorithm, which computes the $[Z]$ matrix, has not data dependencies [5] in the computation of the $[Z]$ terms ($c4$ is a complex constant), so these elements can be calculated in different processors, at the same time. In order to compute the matrix $[Z]$, we use one dispatcher-worker processor and several workers (figure 3). In any case, the data distribution is depending on the later computation.

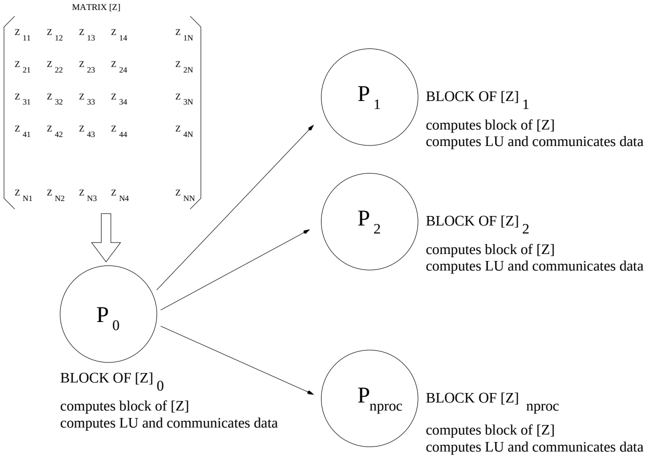
MATRIX [Z]
z_{11} z_{12} z_{13} z_{14} z_{1N}
z_{21} z_{22} z_{23} z_{24} z_{2N}
z_{31} z_{32} z_{33} z_{34} z_{3N}
z_{41} z_{42} z_{43} z_{44} z_{4N}
z_{N1} z_{N2} z_{N3} z_{N4} z_{NN}
P_1
BLOCK OF $[Z]_1$
computes block of [Z]
computes LU and communicates data
P_2
BLOCK OF $[Z]_2$
computes block of [Z]
computes LU and communicates data
P_0
BLOCK OF $[Z]_0$
computes block of [Z]
computes LU and communicates data
P_{nproc}
BLOCK OF $[Z]_{nproc}$
computes block of [Z]
computes LU and communicates data

Figure 3: Dispatcher-Worker scheme.

3.2 LU factorization

In order to distribute the $[Z]$ terms in the local memories of the processors, it's necessary a previous study of the LU sequential algorithm [6] (table 5) and its data dependencies. We solve this problem using our semi-automatic data distribution tool that carries out this study and builds one coloured graph by each dependence [9]. These graphs represent the distribution of the different vectors among the processors, by assigning a colour by processor. In figure 4, we can see the data distribution designs by our tool, using an example with a matrix of 4×4 dimension. In LU sequential algorithm, there are five data dependencies ($A_{i,j} \leftarrow A_{i,j}$ (d_1), $A_{i,j} \leftarrow A_{i,k}$ (d_2), $A_{i,j} \leftarrow A_{k,j}$ (d_3), $A_{i,k} \leftarrow A_{i,k}$ (d_4) y $A_{i,k} \leftarrow A_{k,k}$ (d_5)). Any dependence shows how the data have to be distributed. In this way, the elements enclosed with a line (figure 4), can be distributed in different processors and computed in parallel. But, this is not really true, because communications are needed.

```
do $k = 0, n-2$
    Search for the pivot
    do $i = k+1, n-1$
        $A_{i,k} = A_{i,k}/A_{k,k}$
        do $j = k+1, n-1$
            $A_{i,j} = A_{i,j} - A_{i,k} \times A_{k,j}$
        enddo
    enddo
enddo
```

Table 5: LU factorization.

In order to reduce the number of communication (and reduce the total time of execution), there is a new data distribution: by row or by column. This redistribution is based on the more restrictive distributions that can be seen in figure 4. Although according to the dependency d_1, the elements in a row of the matrix can be stored in different processors, according to the dependency d_2 these elements have to be stored in only one processor. The same happens with the elements in a column of the matrix.

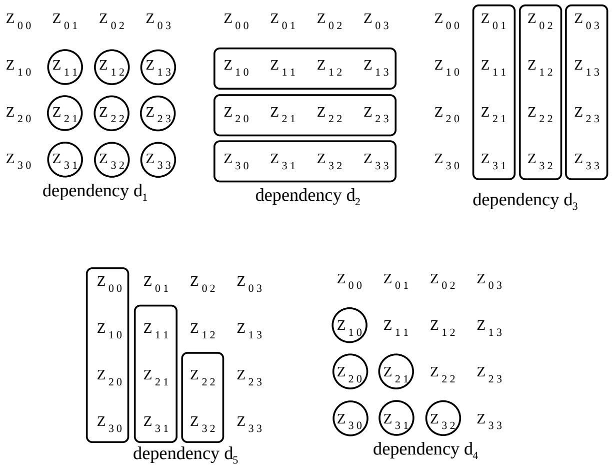

Figure 4: Data distribution of LU factorization ($n = 4$) according to the dependencies.

In this work, we have chosen the data distribution by column, because in any iteration of the algorithm, we have to search for the pivot (maximum absolute value in a column). In order to reduce the number of communications, it's necessary that all the elements in a column of the matrix, are stored in the memory of one specific processor, so that, the processors have not to communicate each other. So, $[Z]$ matrix is distributed in groups of $G = \frac{M}{nproc}$ consecutive columns in a cyclic way. With this data distribution, the LU factorization of submatrices is possible (groups of columns) and then, this block LU is communicated to the remaining processors, in parallel with the computations of the $[Z]$ terms (overlapping of computations and communications).

4 Experimental result

The parallel algorithm was implemented using C and Fortran77 languages and MPI library to communicate the data. The kernel of the computation was developed in Fortran77, because in the code, there are many routines using complex data types. This routines read the geometric files and build the vectors used in the computation of the double integrals. The main program and the LU factorization program were written in C language with some routines in Fortran77. These programs calls Fortran77 routines, interchanging data.

The execution time has been measured in three different architectures: SUN Ultra 5 (UltraSparc II 270MHz/64Mb), DualPentium (300MHz/128Mb) and IBM-SP2[1]. We implemented the parallel program in the IBM-SP2 [7] usign MPI library [8], and measured the sequential time in the three architectures. In the measurement, we took the access to disk into account (read the geometric files and build the final files with the results). In this way, we used double precision (in complex and real numbers) in the computation of the $[Z]$ terms, because the order of the number is between $10e^{-10}$ and $10e^{4}$. In table 6, we can see the studied structures, with their characteristics. The shape of these structures are in figures 5 and 6 The experimental results can be analyzed in table 7.

	Dstub2	*Hybrid*	*Stub2*	*Meander*
Patches	66	44	25	32
Subdomains	92	68	40	62
Vertices	134	96	64	96
Excitations	4	4	2	2
Ports	2	4	2	2
No. Frecuencies	20	10	81	10

Table 6: Structures and characteristics.

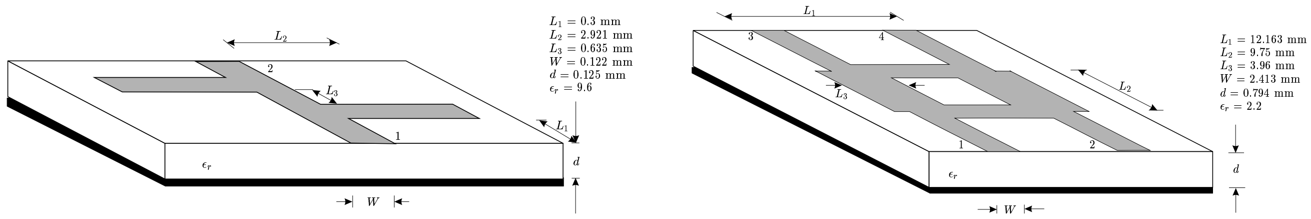

Figure 5: *Dstub2* and *Hybrid* structures.

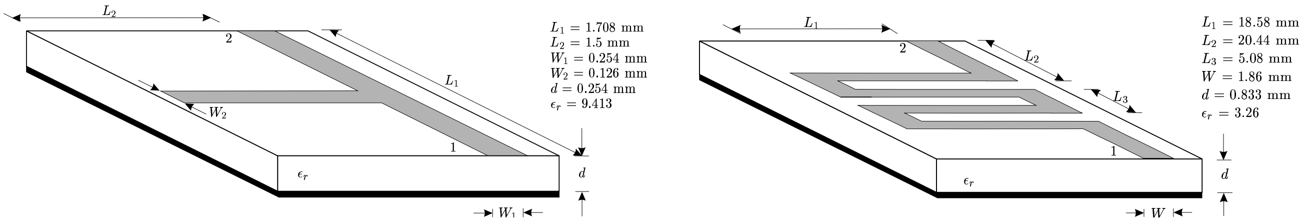

Figure 6: *Stub2* and *Meander* structures.

[1] We gratefully acknowledge C4 (Centre de Computació i Comunicacions de Catalunya) for providing access to the SP-2 and for the technical support.

	Dstub2	*Hybrid*	*Stub2*	*Meander*
SP-2 Parallel	$\simeq 27$ (6 proc)	$\simeq 8$ (4 proc)	$\simeq 30$ (5 proc)	$\simeq 8$ (4 proc)
	$\simeq 25$ (11 proc)	$\simeq 6$ (11 proc)	$\simeq 9$ (25 proc)	$\simeq 5$ (8 proc)
	$\simeq 14$ (22 proc)	$\simeq 4$ (22 proc)		$\simeq 3$ (16 proc)
SP-2 (seq.)	$\simeq 131$	$\simeq 30$	$\simeq 75$	$\simeq 15$
DualPentium	$\simeq 170$	$\simeq 39$	$\simeq 82$	$\simeq 19$
Sun	$\simeq 200$	$\simeq 43$	$\simeq 97$	$\simeq 21$

Table 7: Execution time (minutes).

5 Conclusions

It has been showed the parallelization of the moment method to analyze irregular microstrip structures. There are two steps to parallelize the sequential method, the computation of the $[Z]$ matrix which has not data dependencies and the LU factorization. This method has been applied over several structures such as stubs, dstubs, hybrids in which the computed time has been decreased with the number of proccesors. The data distrubution is important in this method to avoid the communications between proccesors.

References

[1] D. I. Wu, D. C. Chang, B. L. Brim, “Accurate Numerical Modeling of Microstrip Junctions and Discontinuities”, *Int. J. of MIMICAE*, vol. 1, n.1, pp. 48-58, 1991.

[2] E. Jiménez, F. Cabrera, J. G. Cuevas, “Análisis de Onda Completa por el Método de los Momentos de estructuras microtiras eléctricamente grandes”, *Proc. X Symposium Nacional URSI*, Valladolid, pp. 679-682, Sep. 1995.

[3] G. V. Eleftherides, J. R. Mosig, “On the Network Characterization of Planar Passive Circuits Using the Method of Moments”, *IEEE Trans. on Microwave, Theory Tech.*, vol. 44, pp. 438-445, Mar. 1996.

[4] E. Jiménez, F. Cabrera, J. G. Cuevas, “Sommerfeld: A Fortran Library for Computing Sommerfeld Integrals”, *IEEE Antennas and Propagat. Society International Symposium*, Baltimore, pp. 966-969,Jul. 1996.

[5] U. Banerjee U, “Dependence Analysis for Supercomputing”, *Kluwer Academic Publishers*, 1988.

[6] G. H. Golub, C. F. Van Loan, “Matrix Computations. Second edition”, *The Johns Hopkins University Press*, 1989.

[7] Information available in: http://ibm.tc.cornell.edu/ibm/pps/spe/sp2.html.

[8] Information available in: http://netlib.org/mpi.

[9] C. N. Ojeda-Guerra, “Methodology of Parallelization of Algorithms Based on Coloured Graphs”, PhD Thesis, May 2000 (in Spanish).