Progress In Electromagnetics Research, PIER 56, 93–108, 2006

UNSUPERVISED SYNTHESIS OF MICROWAVE COMPONENTS BY MEANS OF AN EVOLUTIONARY-BASED TOOL EXPLOITING DISTRIBUTED COMPUTING RESOURCES

M. Donelli, R. Azaro, and A. Massa

Department of Information and Communication Technologies
University of Trento
Via Sommarive 14, 38050 Trento, Italy

M. Raffetto

Department of Biophysical and Electronic Engineering
University of Genoa
Via Opera Pia 11a, 16145 Genoa, Italy

Abstract—A parallel implementation of an automatic CAD tool based on the parallel virtual machine software package, genetic algorithms and finite element simulators is presented. It is shown that the parallel implementation can be obtained by developing just a few hundred lines of code and a pseudocode description is provided. Finally, selected numerical results are provided in order to show the effectiveness and the reliability of the proposed approach.

1. INTRODUCTION

Microwave CAD tools are aimed at providing the designer with an environment where a given component can be characterized, investigated and also modified towards a desired electrical response. As far as the modification of a given component is concerned, a key point is the development of microwave CAD environments [1] where the component is modified automatically by the microwave tool itself with an improvement of the component performances in an unsupervised fashion.

Such useful automatic design tools have been the object of research for some years [2–4]. As a common feature, these environments integrate an optimizer and a numerical simulator [2–4].

Even though in the last decade the performances of computers and of electromagnetic simulators have been improved in an impressive way, the computation costs required for the analysis of many microwave devices can last for several minutes on typical desktop computers [4].

For this reason, as pointed out in [4], there is the need of the "development of a variety of optimization techniques aimed at reducing to a bare minimum the number of times a full electromagnetic-field analysis has to be performed". Another way to obtain significant time savings is to calculate relatively inaccurate numerical solutions in the early stages of the optimization and progressively improve the accuracy as the optimization process goes on [4]. However, quite recently another possibility has emerged [5–7]. It is based, on the one hand, on the fact that nowadays several interconnected computers are available in most laboratories, thus making the actual computational power already installed much bigger than that of a single standard computer usually exploited, and, on the other hand, on the fact that there are free software packages [6] (MPI, PVM, etc. [8]) making the exploitation of this big distributed computational power simple and affordable [5].

In this relatively new context, the reduction to a bare minimum of the number of full electromagnetic field analysis required by the optimization process is not anymore a need and new strategies can be developed. As a matter of fact, it proves conveniently to use optimization algorithms that are well suited to parallelism or intrinsically parallel such as Genetic Algorithms (GAs) instead of using serial optimization algorithms like Conjugate Gradient (CG). Such an approach was used, for instance, in [9] and [10]. It is well known that the choice of a GA is not the best one, in terms of the number of cost-function evaluations, when it is possible to start from an initial guess which is close to the final design configuration [4]. However, the switch from serial to parallel computing could allow quicker device designs even if more cost-function evaluations are required. Moreover, the condition needed on the good initial guess not necessarily holds true, especially when complex microwave devices are to be designed and this is likely to be the usual application of CAD tools in the future. Thus, on the one hand, GAs can easily be implemented to fully exploit distributed computing resources quite efficiently and, on the other hand, their performances as global optimizer could be relevant in sampling the solution space.

This paper is aimed at showing, on the one hand, that it is a trivial task to devise a simple but performing parallel implementation of the whole automatic design code, when this is done by using PVM [8], the optimization algorithm is based on a GA and the tool of numerical

analysis can be considered as a black box (Section 2). On the other hand, the manuscript will show that the proposed approach guarantees, at the same time, interesting overall performances in terms of speed-up [7] of the automatic design phase (Section 3). In particular, by using a sufficiently high number of computers working in parallel, the major drawback of GAs is overcome whereas its features as global optimizers are retained. The speed-up, which can be achieved by using different numbers of parallel computers, will be reported and the corresponding parallel efficiency is shown to be relatively close to unity as far as the number of computers available is less than or equal to the number of individuals of each population of the GA. Finally, it will be shown that the proposed approach does not prevent the exploitation of other ideas which could reduce the global design time.

2. PARALLEL IMPLEMENTATION OF THE AUTOMATIC CAD TOOL

Let us assume that the numerical analyses required for any particular device configuration considered during the optimization process can be dealt with as a unique serial process. In this sense, let us consider design tools particularly suited to deal with devices of small to medium complexity. The automatic design of highly complex devices could require the parallelization of the code of numerical analysis and this is out of the aims of this work.

As far as the numerical analysis is concerned, there are several numerical methods that can be exploited for this serial task, which are reliable and which are available in most laboratories (by the way, we decided to use the finite element (FE) method [11, 12] based on lowest order edge elements [12, 13]). Thus, without loss of generality, we can suppose that a script file able to read from a file some parameters identifying the configuration of the device of interest to be analyzed and to manage all necessary computations to calculate the corresponding cost function value is available.

With the indicated simplifying assumption, parallelism can be exploited just at the optimization level by using the so-called master-slave paradigm [8]. The optimizer is the master, which spawns several FE based slave serial processes, one per cost function evaluation required.

Since parallelism is just considered at the optimization level it was natural to choose a minimization algorithm intrinsically parallel. Thus a GA was the natural choice. GAs are multi-agent stochastic search methods that have been successfully applied in many syntheses and optimization problems (see [14] and the references

cited therein). Unlike most other optimization techniques, a GA maintains a population of M encoded trial solutions that evolve in the solution space by means of some genetic operators in order to reach a satisfactory solution. Let us outline the skeleton of such algorithms (Figure 1). A GA proceeds in an iterative way by generating new populations of trial solutions $\left\{\underline{x}_m^{(k+1)};\, m = 1, ..., M\right\}$ starting from the old ones $\left\{\underline{x}_m^{(k)};\, m = 1, ..., M\right\}$. Each individual is ranked according to the associated fitness function $\left(f\left\{\underline{x}_m^{(k)}\right\};\, m = 1, ..., M\right)$ and suitably encoded $\underline{\chi}_m^{(k)} = \Im\left\{\underline{x}_m^{(k)}\right\}$. The standard algorithm applies stochastic operators such as selection, crossover, and mutation to generate a new population. The iterative process is terminated when a fixed number of iterations has been reached ($k = K$) or when the fitness function of the optimal solution satisfies assigned synthesis constraints.

For a practical parallel implementation of the algorithm it is sensible to choose a software package that simplifies the exploitation of the power of a cluster of computers and the intrinsic parallelism of the optimizer. Our choice was PVM [8] since it provides simple subroutines for the management of the network, the spawning of processes and the transmission and reception of data between processes. Moreover, in order to exploit parallelism it is natural to focus on the step of the genetic algorithm whose target is the "evaluation of the cost function for all individuals of a given population", since this task can be carried out in parallel by assigning the calculation of the cost function of different individuals to different computers. In order to further reduce the computational time required for ranking the individuals of each population fully exploiting the intrinsic parallelism of the algorithm, the parallel implementation considers also a memory-based control for function evaluation. To this end, at each generation, the higher order schemata [15] of the GA are stored and the new trial solutions are compared with reference models. If a condition of high similarity is satisfied, the cost function is not evaluated. We will actually consider this step concerning the ranking of the individuals of a population as a subroutine of the GA code. In our parallel implementation this is the only subroutine which required a modification since all other parts of the GA code are not changed (apart from some trivial variable declarations and parameter definitions at the main level) with respect to their serial implementations.

A simple high level pseudocode of this subroutine is the following, where it has been assumed that a population of M different individuals is given as an input parameter:

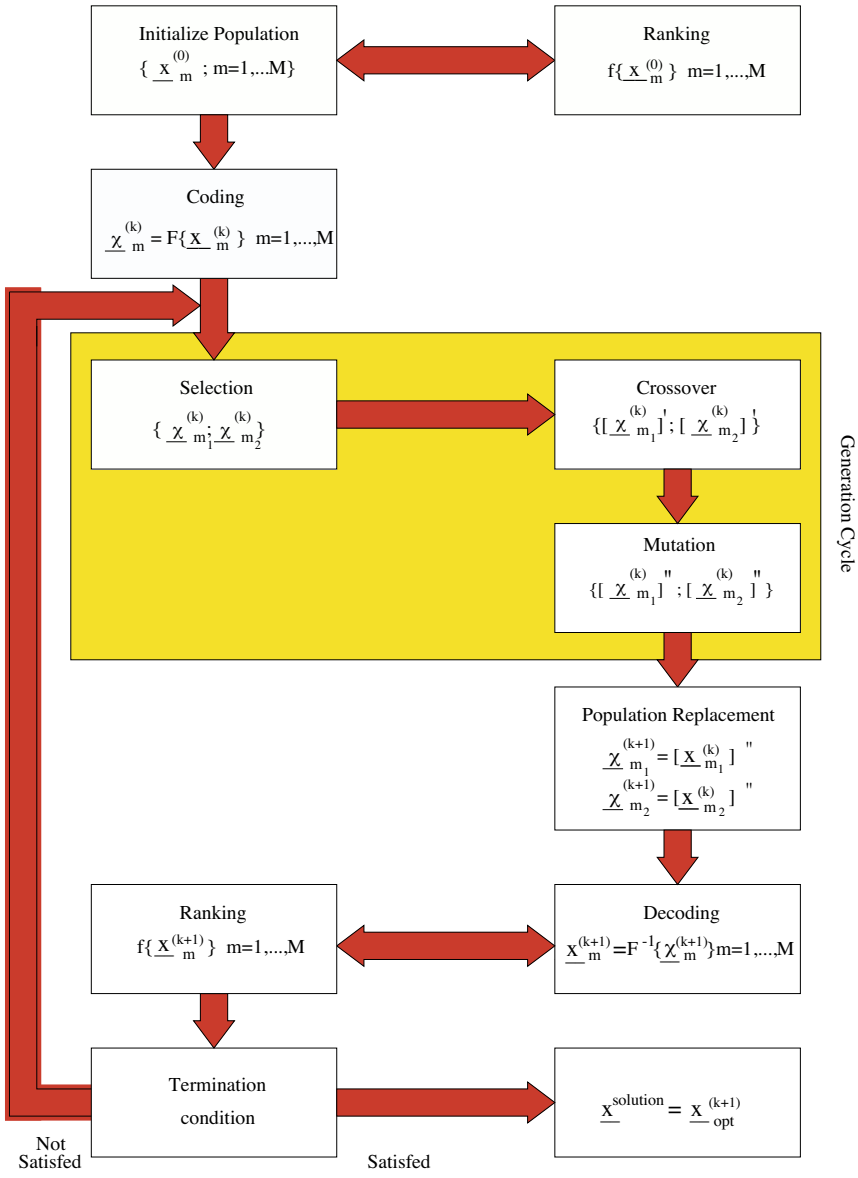

Figure 1. Flow chart of a standard GA.

```
Find the number M of new individuals
Find the number N of hosts available
if P ≤ N then
{
    Spawn P slave processes
    Send individual data to the spawned slave processes
    Wait for a reply from all spawned slave processes
}
else
{
    Spawn N slave processes
    Send individual data to the spawned slave processes
    Repeat until the no. of spawned processes is lower than P
{
        Wait for a reply from a slave process
        Spawn a new slave process
        Send the data of a single individual to the new process
}
    Wait for the slave replies not received yet
}
Return control to the calling program
```

Each time this subroutine is executed, it has to determine the number of new individuals requiring a cost function evaluation and the number of hosts available. Once this is done, it has to distinguish two cases: when there are more hosts than design configurations (new GA individuals) to be analyzed and when the opposite holds true. The first possibility can be managed in a simpler way (see the "if" part of the above pseudocode) since it is possible to assign different computations to different hosts. But note that also the latter is not complicated (see the "else" part of the above pseudocode). As we have already pointed out, in order to keep to a minimum the complexity of the switch from serial to parallel computing we considered the script file which manages the tool of numerical analysis as a black box. For this reason and also to reduce the complexity of the top-level slave code we use each host for the calculation of one cost function value at a time. Note that, however, the calculation of two or more cost function values on a single host at a time does not give significant improvement in terms of performances and could require an increasing in the complexity of the top-level slave code since that, for instance, two different numerical analyses could disturb one another if executed at the same host directory.

From a practical point of view, it is quite interesting to note that, by using PVM and C or Fortran languages, it is possible to implement such a subroutine in about one hundred lines of code.

So far we have just considered a high level description of a subroutine of the master. On the slave side, PVM must be exploited too in order to make the master-slave communication possible. Under indicated assumptions, the task a single slave process has to carry out is trivial and the high level pseudocode of this part of the algorithm is

Receive the data of a single individual sent by the master
Write the received data into a dialog file
Execute the finite element based simulator script file
Wait until its end
Read the cost function value from the dialog file
Send back this value to the master
Notify the end of the task to the master

A PVM based implementation of this code can be obtained in less than fifty lines of code. Thus, in summary, it is possible to obtain a parallel implementation of the whole automatic CAD tool by developing less than two hundred new lines of code overall (taking account even of variable declarations, parameter definitions, etc., at the main level) in a PVM environment whenever a GA and a FE based simulator are already available.

Moreover, it should be pointed out that the performances of such an algorithm can still be improved in a significant way by relaxing our initial assumption at the cost of an increasing of the complexity of the code to be developed, provided that the cluster of computers exploited is bigger than the number of new chromosomes of any single population. As a matter of fact, the evaluation of the cost function for any new particular device configuration considered by the optimization algorithm requires the numerical analysis of the device for a number [16] of discrete frequencies in the frequency band of interest [4] and all such numerical analyses can be carried out in parallel. In order to manage such a generalization by retaining the master-slave paradigm exploited above, it is necessary to simplify the script file which manages the numerical analyses of a given device configuration at all, let us say, F frequencies of interest by reducing it to a tool of simulation at a single frequency and, at the same time, make the master able to manage the spawning of $P \times F$ processes by calculating just P cost function values. With this change more complicated devices can be analyzed with an increased value of speed-up provided that the number N of available computers working in parallel is bigger than P.

3. PERFORMANCES OF THE PARALLEL AUTOMATIC CAD TOOL

In order to complete the analysis of the chosen approach, it is mandatory to show that the developed CAD tool is able to fully exploit the power of a cluster of computers providing impressive speed-up values compared to the serial implementation of the same algorithm.

For this purpose, the described code has been applied to the design of a three port junction. The three ports are $WR28$ rectangular waveguides (7.112 mm by 3.556 mm). It is required that at a frequency of 38.5 GHz the following conditions are satisfied:

$$\eta_{12}^{min} \leq |s_{12}| \leq \eta_{12}^{max}, \quad \eta_{13}^{min} \leq |s_{13}| \leq \eta_{13}^{max}, \quad \eta_{22}^{min} \leq |s_{22}| \leq \eta_{22}^{max} \tag{1}$$

being

$$\frac{\eta_{12}^{max} + \eta_{12}^{min}}{2} = 0.606, \qquad \frac{\eta_{12}^{max} - \eta_{12}^{min}}{2} = 0.032,$$
$$\frac{\eta_{13}^{max} + \eta_{13}^{min}}{2} = 0.650, \qquad \frac{\eta_{13}^{max} - \eta_{13}^{min}}{2} = 0.030$$

and

$$\frac{\eta_{22}^{max} + \eta_{22}^{min}}{2} = 0.445, \qquad \frac{\eta_{22}^{max} - \eta_{22}^{min}}{2} = 0.045$$

In order to do so the position d, the width w, and the length d of a diaphragm (whose height is equal to that of the waveguide) placed in the central region of an H-plane T junction are to be determined (see Figure 2).

These design requirements are considered just to point out the important features and the achievable performances of the proposed parallel and automatic CAD tool and no practical direct application is presented.

Since the requirements are specified in terms of s_{13} and s_{22} it is necessary to perform two numerical calculations for any design configuration considered by the GA. Thus having specified a fixed working frequency is not to be regarded as a too heavy simplification since each serial FE-based task could in principle be splitted into two parallel FE numerical analyses to improve the parallel efficiency, exactly as indicated at the end of the previous section for the efficient management of multiple frequency analyses.

In order to synthesize a microwave circuit satisfying (1), the following cost function was defined

$$f\{\underline{x}\} = f_1\{\underline{x}\} + f_2\{\underline{x}\} + f_3\{\underline{x}\} \tag{2}$$

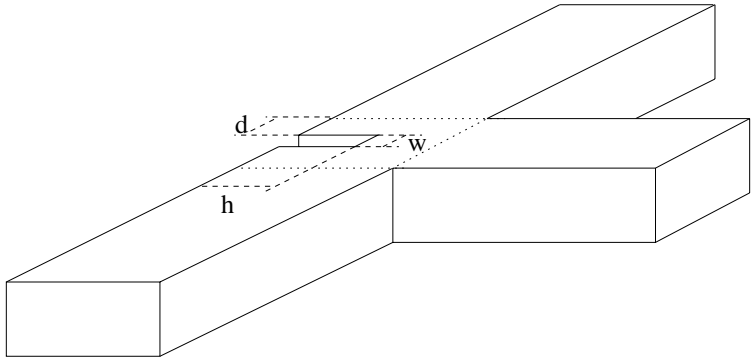

Figure 2. H-plane T junction with a diaphragm. The diaphragm position d, width w, and length h are to be determined.

where

$$f_1\{\underline{x}\} = max\left\{\left||s_{12}| - \frac{\eta_{12}^{max}+\eta_{12}^{min}}{2}\right| - \frac{\eta_{12}^{max}-\eta_{12}^{min}}{2}; 0\right\}$$

$$f_2\{\underline{x}\} = max\left\{\left||s_{13}| - \frac{\eta_{13}^{max}+\eta_{13}^{min}}{2}\right| - \frac{\eta_{13}^{max}-\eta_{13}^{min}}{2}; 0\right\}$$

$$f_3\{\underline{x}\} = max\left\{\left||s_{22}| - \frac{\eta_{22}^{max}+\eta_{22}^{min}}{2}\right| - \frac{\eta_{22}^{max}-\eta_{22}^{min}}{2}; 0\right\}$$

$\underline{x} = \{d, w, h\}$ being the array of the design parameters.

It is well known that GAs are global optimizers. For such a reason they are particularly useful in the early stages of the design when local minima can still have heavy effects on the convergence of algorithms like CG to the global minimum. No hypothesis on the quality of the initial guess is made, because this is representative of more complicated situations, even though for the simple design we are considering in this example a good initial guess could probably be found. Moreover, as pointed out in [4], relatively inaccurate numerical solutions can be carried out during the early stages of the design provided that this accuracy is increased as the optimization process goes on. On the basis of these considerations, we carried out an automatic design starting from a (pseudo) randomly selected initial population of the GA. Moreover, a relatively coarse discretization was used in order to perform the FE based simulations of the junction. Note that, however, the discretization is not made finer and finer as the work of the optimizer proceeds, since this improvement could complicate the

analysis of the relative performances of the parallel automatic CAD tool in comparison with those of its serial implementation.

The above indicated relatively coarse discretization is obtained by discretizing the longer side of the waveguide into $i = 10$ linear elements, the shorter one into $j = 5$ segments, the length (= 25 mm) of the lateral arms (ports 1 and 3) into $l = 25$ segments and the length (= 20 mm) of the central port (port 2) into $q = 20$ linear elements. The obtained hexahedra are then divided into six tetrahedra [11]. The diaphragm always extends over a region given by the union of an integer number of hexahedra and, for this reason, an integer coded GA was adopted [9].

If i_1 indicates the first hexahedra belonging to the diaphragm starting from the left of the central region (see Figure 2), i_2 provides its width in terms of number of hexahedra, and i_3 indicates its length, again in terms of number of hexahedra, we have that, if the diaphragm is completely contained in the central region of the junction (see the dotted lines in Figure 2), the following conditions must be satisfied:

$$1 \leq i_1 \leq i,$$

$$1 \leq i_3 \leq i,$$

$$1 \leq i_2,$$

$$i_1 + i_2 \leq i + 1,$$

$$i_3 \leq i \text{ when } (i_1 = 1) \text{ or } (i_1 + i_2 = i + 1)$$

Thus, overall, there are

$$i\,(i-1) + \sum_{i_1=2}^{i} \{i\,(i - i_1) + (i-1)\} = \frac{i^3 + i^2 - 4i + 2}{2},$$

possible different configurations of the diaphragm. To carry out fair comparisons between serial and parallel implementations of the same algorithm, the initial pseudo-random population of the GA was always the same and in all cases the populations are made up of $M = 10$ individuals. This guarantees that the evolution of the populations generated by the GA is always the same independently of the number of computers exploited.

During its evolution the GA generated five populations ($k = 5$ in Figure 1) in addition to the first one in order to synthesize a good (satisfying inequalities (1)) design configuration. A top view of the central part of the junction thus determined is shown in Figure 3 where some indications on the fixed 3D discretization adopted are reported as a 2D mesh of rectangles (for each rectangle in the 2D mesh shown there are $j = 5$ hexahedra; the 3D mesh is then given by discretizing

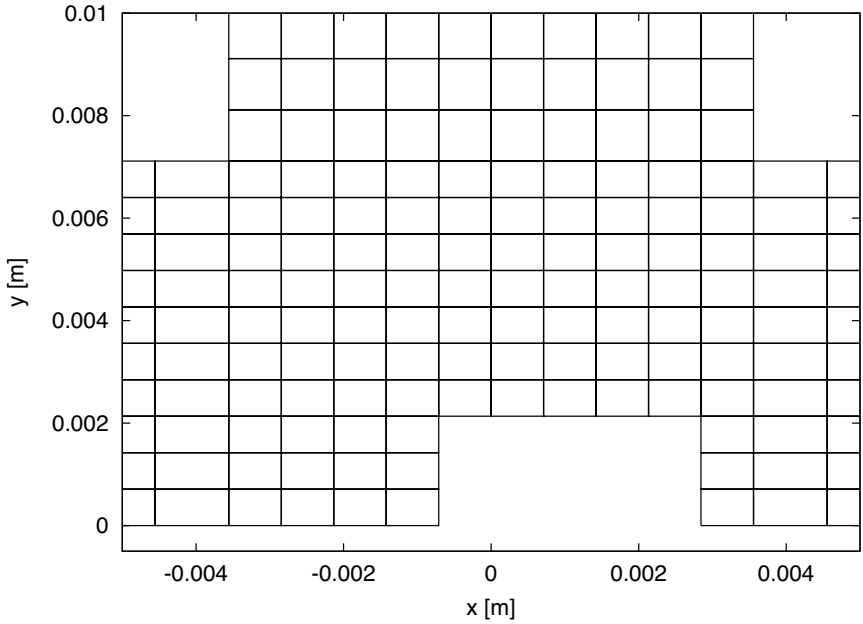

Figure 3. Final design configuration of the diaphragm. Top view of the central part of the junction.

each hexahedra into six tetrahedra). The final design configuration has a diaphragm identified by $i_1 = 5$, $i_2 = 5$, and $i_3 = 3$ corresponding to a cost function value of $f\left\{\underline{x}_{opt}^{(5)}\right\} = 0.0$.

The cost function values corresponding to the best individuals of all populations generated by the GA are reported in Figure 4. For completeness all addends of the right-hand side member of (2) are reported.

In order to limit the number of computationally intensive simulations and taking into account of the memory-based control strategy the computation of the cost function is avoided for all individuals of the current population that have already been considered in previously generated populations. For this integer coded GA exploiting coarse discretizations in FE analyses the high similarity criteria is actually an equality criteria. Thus the algorithm had to carry out M FE-based simulations for all M individuals of the initial population but just $(M-2)$ for the second $(k=1)$, $(M-3)$ for the third $(k=2)$ and fourth $(k=3)$ and $(M-4)$ for the fifth $(k=4)$ and sixth $(k=5)$, resulting in 44 different diaphragm configuration

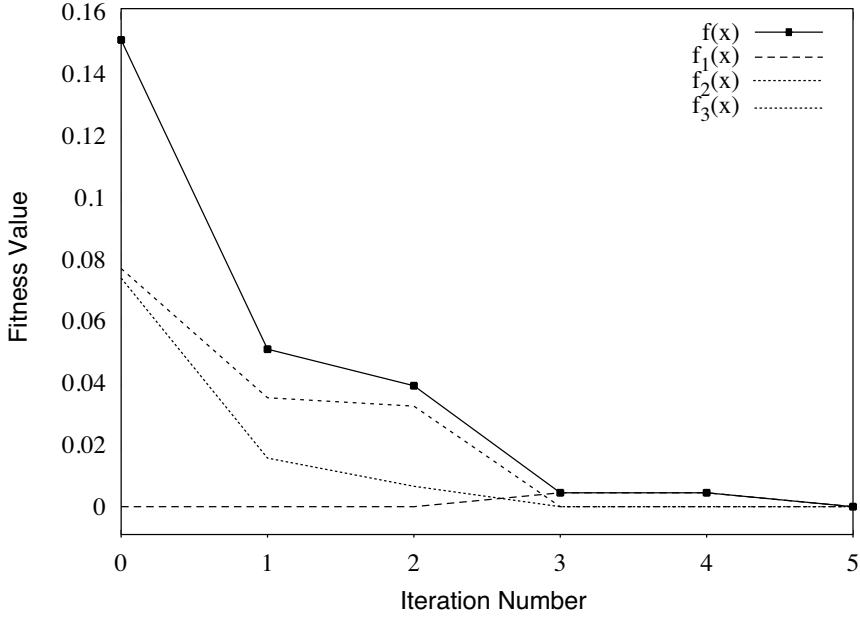

Figure 4. Behavior of the optimal value of the cost function and related terms versus k.

analyses overall (out of 531 different possibilities).

By considering the numerical evaluation of the cost function for a single diaphragm configuration as a single serial task the results reported in Figure 5 are achieved.

Let us consider the serial case ($N = 1$) for comparison purposed. If t_s denotes the overall CPU-time required to complete the automatic design in the serial case and $t_p(N)$ denotes the time needed to complete the automatic design when N computers are exploited in parallel, the parameter most commonly used to measure the usefulness of the parallel implementation is the speed-up, which is defined as follows:

$$SU\,(N) = \frac{t_s}{t_p\,(N)} \tag{3}$$

the ideal speed-up being

$$ISU\,(N) = N \tag{4}$$

when the N computers exploited in parallel are all identical. To evaluate the computational effectiveness, other parameters are used as well. The normalized time is by definition

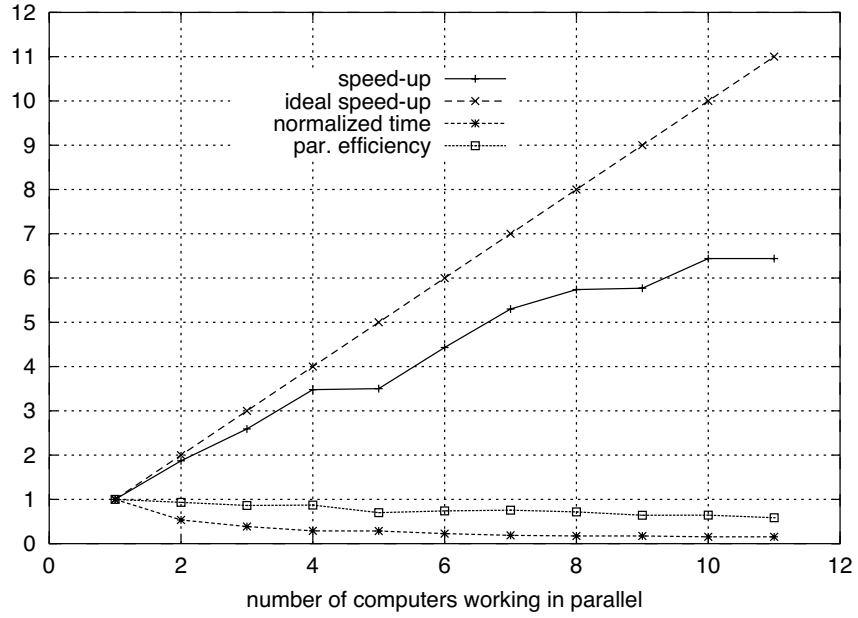

Figure 5. Performances of the parallel architecture.

$$NT\,(N) = \frac{1}{SU\,(N)} \tag{5}$$

and the parallel efficiency is

$$PE\,(N) = \frac{SU\,(N)}{ISU\,(N)}. \tag{6}$$

The results shown in Figure 5 are obtained when no other tasks (except for the master and those tasks it spawns) are carried out by the computers exploited during the parallel automatic design. However, due to the fact that an iterative solver is used in the FE based simulator it is difficult to predict the execution time required to complete the design even though all simulations have to deal with the same number of unknowns.

Figure 5 shows that even though the GA requires 44 different evaluations of the cost function, the design time is reduced by a factor equal to 6.5 when $N = 10$ parallel computers are exploited, a time required to carry out less than 7 cost function evaluations on average on a serial machine (one of the cluster made up of 12 equal PCs with the following characteristics: Pentium IV, $f = 1.7\,\mathrm{GHz}$, 256 MB of RAM). The parallel efficiency, representing the algorithm ability to

fully exploit the available computational power, is close to unity in all cases ($PE(N) \geq 0.65$, $N = 1, ..., 10$) of interest. The case of $N = 11$ computers is reported for the sake of completeness but the result in this case was exactly predictable since the GA with $N = 10$ chromosomes per population is not able to exploit more than 10 computers in parallel when the simplest algorithm is used.

Moreover, it should be pointed out that if the generalization described in Section 2 is implemented, by splitting the FE based serial task that requires the numerical solution of two different problems, it is possible to divide approximately by an additional factor 2 the overall computational time when the number of computers is twice the number of individuals of a single population. Thus, to be conservative, it is expected that by using a cluster of $N = 20$ computers the final design could be achieved in a time required for just 4 (> 3.5) cost function evaluations on a serial machine.

On summary, the parallel implementation of the algorithm overcomes the major drawback of GAs and it retains the GA's features as a global optimizer provided that several computers can be concurrently exploited. When such a number is sufficiently high even a trivial implementation of the whole code can guarantee important absolute performances.

Finally, let us point out that the proposed implementation does not prevent the exploitation of other useful ideas, such as those presented in [4]. Moreover, from an implementation point of view, it would be extremely easy to use different discretizations to calculate the cost function for individuals of different populations.

4. CONCLUSIONS

An evolutionary-based automatic CAD tool able to exploit distributed computing resources is presented and a detailed description of its possible implementation is provided. A simple design is carried out to show what impact parallelism can have on the performances of such kinds of CAD tools. Finally, some comments are provided on the possible exploitation in this parallel framework of further improvements currently under development.

REFERENCES

1. Steer, M. B., J. W. Bandler, and C. M. Snowden, "Computer-aided design of RF and microwave circuits and systems," *IEEE Trans. Microwave Theory Techn.*, Vol. 50, No. 3, 996–1005, Mar. 2002.

2. Bandler, J. W., R. M. Biernacki, S. H. Chen, and D. Omeragic, "Space mapping optimization of waveguide filters using finite element and mode-matching electromagnetic simulators," *Int. J. Radiofreq. Microwave Computer-aided Eng.*, Vol. 9, No. 1, 54–70, Jan. 1999.
3. Bila, S., D. Baillargeat, M. Aubourg, S. Verdeyme, and P. Guillon, "A full electromagnetic CAD tool for microwave devices using a finite element method and neural networks," *Int. J. Num. Modeling*, Vol. 13, 167–180, Mar. 2000.
4. Gavrilovic, M. M. and J. P. Webb, "Accuracy control in the optimization of microwave devices by finite-element methods," *IEEE Trans. Microwave Theory Techn.*, Vol. 50, 1901–1911, Aug. 2002.
5. Tarricone, L., M. Mongiardo, and C. Tomassoni, "A parallel framework for the analysis of metal-flanged rectangular-aperture arrays," *IEEE Trans. Microwave Theory Techn.*, Vol. 49, 1479–1484, Oct. 2001.
6. Esposito, A., F. Malucelli, and L. Tarricone, "An optimized parallel admittance matrix approach using the adjacence-graph recursive-thresholding technique," *IEEE Trans. Microwave Theory Techn.*, Vol. 50, 2102–2107, Sep. 2002.
7. Jiang, D., W. Meleis, M. El-Shenawee, E. Mizan, M. Ashouei, and C. Rappaport, "Parallel implementation of the steepest descent fast multipole method (SDFMM) on a Beowulf cluster for subsurface sensing applications," *IEEE Microwave Wireless Comp. Lett.*, Vol. 12, 24–26, Jan. 2002.
8. Geist, A., A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine. A users' guide and tutorial for networked parallel computing*, The MIT Press, Cambridge, MA, 1994.
9. Caorsi, S., M. Donelli, A. Massa, and M. Raffetto, "A parallel implementation of an evolutionary-based automatic tool for microwave circuit synthesis — preliminary results," *Microwave Opt. Technol. Lett.*, Vol. 35, 169–172, Nov. 2002.
10. Lucci, L., R. Nesti, G. Pelosi, and S. Selleri, "Optimization of profiled corrugated circular horns with parallel genetic algorithms," *Proc. Antennas Propagat. Soc. Int. Symp. 2002*, 338–341, Salt Lake City, Utah, USA, Jul. 2002.
11. Silvester, P. P. and R. L. Ferrari, *Finite Elements for Electrical Engineers,* Cambridge University Press, Cambridge, 1990.
12. Jin, J., *The Finite Element Method in Electromagnetics*, John Wiley & Sons, New York, 1993.

13. Caorsi, S., P. Fernandes, and M. Raffetto, "On the convergence of Galerkin finite element approximations of electromagnetic eigenproblems," *SIAM J. Num. Anal.*, Vol. 38, 580–607, 2000.
14. Rahmat-Samii, Y. and E. Michielssen, *Electromagnetic Optimization by Genetic Algorithms*, Wiley and Sons, New York, 1999.
15. Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
16. Sun, D. K., Z. J. Cendes, and J. F. Lee, "Alps — A new fast frequency-sweep procedure for microwave devices," *IEEE Trans. Microwave Theory Techn.*, Vol. 49, 398–402, 2001.