ADAPTING THE NUMERICAL ELECTROMAGNETICS CODE TO RUN IN PARALLEL ON A NETWORK OF TRANSPUTERS

D.C. Nitch and A.P.C Fourie
Department of Electrical Engineering
University of the Witwatersrand
Private Bag 3
2050
South Africa

1 ABSTRACT

The Numerical Electromagnetics Code (NEC) has been adapted to run in parallel on a 16 transputer PARSYTEC machine. The modification of the code involved the manipulation of the Fortran source code and the development of parallel algorithms to fill and factor the matrix. The performance of the parallel NEC improved as the number of segments used in the simulation increased. When simulating a model with 300 segments the time taken was 45 seconds which is 13.3 times faster than using one processor.

2 INTRODUCTION

The ACES PC version of the Numerical Electromagnetics Code (NEC2)[1] has been run on a PC at the University of the Witwatersrand for much of its electromagnetic work. The main advantage of the PC is its convenience, but it is unfortunately exceedingly slow and insufficient memory limits the electrical size of structures that can be analyzed. This is especially so when one simulates structures at specific frequency increments, or when the structure is electrically large. As an example of the limitations encountered with available PC's (80286, 20 MHz clock frequency) the time taken to simulate a 100 segment structure is 50 minutes.

Stellenbosch University introduced a method of maintaining the convenience of a PC whilst considerably improving its performance. This was achieved in April 1988 when Stellenbosch University announced the successful running of NEC on a single INMOS T800 transputer[2].

A transputer is a single chip microprocessor which was developed by the British company INMOS. It consists of an on-chip processor, floating point unit, memory and four communication links for direct communication to other transputers. This microprocessor distinguishes itself from others in that it is designed as a building block for parallel processing. Concurrent systems

may be constructed from a collection of transputers operating concurrently and communicating through links (see Figure 1 for a block representation of the T800). This paper presents the modifications made to the NEC algorithms to allow parallel execution on 16 transputers. The machine used is a portion of the Parsytec Super Cluster with 16 transputers, each with 1MByte of RAM.

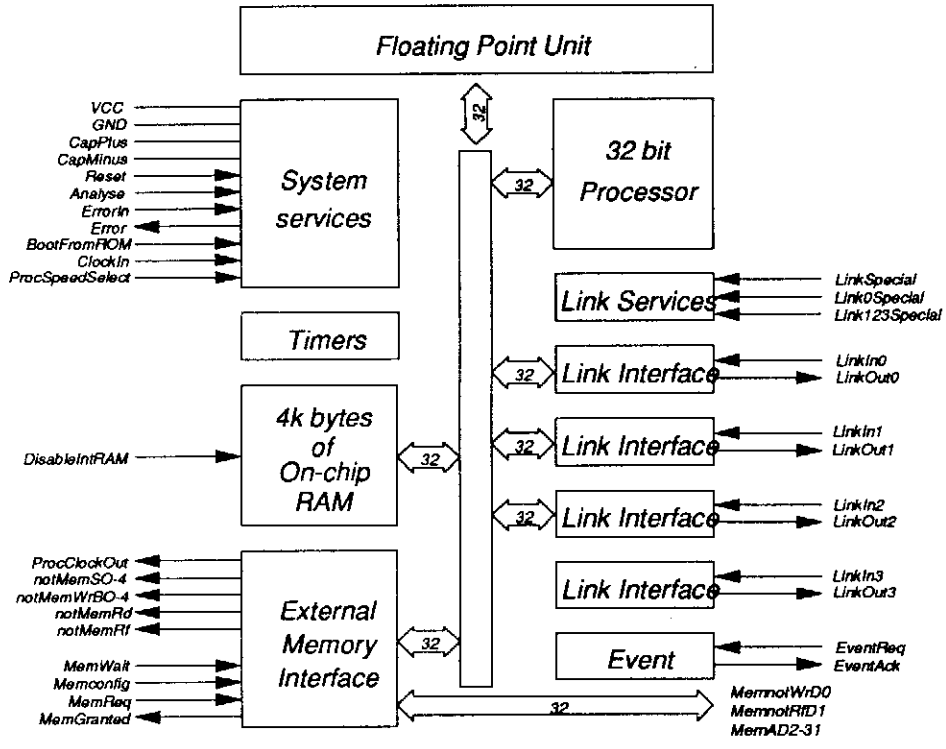

Figure 1: The internal architecture of the T800 transputer.

Table 1 was drawn up to compare a single transputer, in terms of its performance and cost, to various well known micro-processors. The performance of each is gauged by the time required to simulate a 100 segment single wire structure using NEC.

Processor	Run time (secs)	Cost (dollars)
286/20 (w/80287)	750	600
386/20 (w/80387)	60	2100
386/20 (w/1167)	24	2600
T800/20 transputer	45	1200

Table 1: A comparison of the performance and cost of various well known processors (mother board with 1 MByte of memory included for all cases).

The advantage of the transputer over other processors is that it is easy to increase the performance of a processing system by adding transputers. This could be achieved at a cost roughly proportional to the increase in processing power. Improving the performance of other computers normally requires discarding the existing processor and purchasing a more advanced processor.

3 ADAPTING THE CODE TO RUN ON PARALLEL PROCESSORS

There are a number of routines in the code that may be adapted to run in parallel. For many simulations, however, experience has shown that the most time consuming section is the solution of the integral equations using Method of Moments. Thus the Method of Moment (MOM) solution was taken as a starting point in the parallelization process. Mention is made, however, of how one might adapt the radiation pattern routines for use on the transputer network. The MOM solution was considered in two parts, first the filling of the interaction matrix and secondly the solution of this matrix.

3.1 FILLING THE MATRIX

The approximation of the current in a segment used for the method of moment solution in NEC is of the following form :-

$$fj = Aj + Bj\text{Sin } k(z-z') + Cj\text{Cos } k(z-z')$$

This function peaks on segment j and extends onto all segments connected to segment j as shown in figure 2 :-

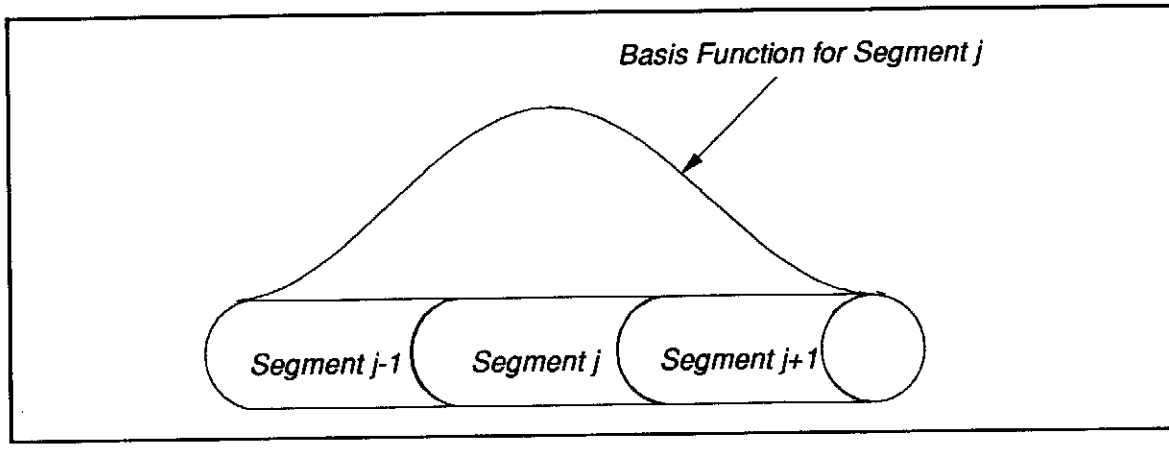

Figure 2: The basis function employed in NEC.

In the process of filling the matrix each segment is considered individually and its current determined in terms of their Sine and Cosine components. The value of the constant term A_j is unknown at this stage and is set equal to -1. Using this current, the values of the tangential E field induced at the centre of all segments in the structure are calculated. These values make up the elements of the interaction matrix.

The parallelization of any code requires knowledge of the data dependence on the various steps made in the computation. In filling the matrix one must consider calculation of the current and the fields induced by this current. The calculation of the currents on a particular segment requires consideration of local boundary conditions of the segment in question. These boundary conditions are functions of the antenna geometry in the vicinity of this segment. The fields induced in the structure at the segment centres are functions of the geometric location of the source and observation segments. Thus the individual calculations of the induced fields due to the current on a particular segment are independent of each other and are purely a function of the antenna geometry.

Parallelization of the filling of the matrix was thus accomplished by distributing the entire geometry of the structure to all the processors and giving each processor the task of calculating the fields induced by the currents on specific segments.

On completion of their tasks the processors return their results to a host processor which places them in their corresponding matrix locations. Thus the matrix is filled in parallel.

3.2 SOLVING THE MATRIX

The matrix in NEC is solved using Gaussian Elimination with back substitution. Of these two processes the triangularization of the matrix is the most time-consuming and it was this part of the code that was parallelized. The sequential algorithm for triangularizing an n*n matrix using pivoting is given in figure 3. The parallel implementation of this algorithm was based on the algorithm presented in a paper by Giest and Romine[3]. The algorithm is as shown in figure 4.

```
FOR k = 0 TO n-1
   determine pivot row
   swap elements
   FOR i = k+1 TO n-1
      a_ik = a_ik/a_kk
   END (FOR i)
   FOR j = k+1 TO n-1
      swap elements of column j
      FOR i = k+1 TO n-1
         a_ij = a_ij - a_ik a_kj
      END (FOR i)
   END (FOR j)
END (FOR k)
```

Figure 3: The serial algorithm for triangularizing an N*N matrix.

```
FOR k = 0 TO n-1
  IF (I own column k) THEN
    determine pivot row
    interchange
    FOR i = k+1 TO n-1
      l_ik := a_ik/a_kk
    END (FOR i)
    broadcast l and pivot index
  ELSE
    receive l and pivot index
  END IF
  FOR (all columns j>k that I own)
    interchange
    FOR i = k+1 TO n-1
      a_ij := a_ij - l_ik a_kj
    END (FOR i)
  END FOR (all columns j>k that I own)
END (FOR k)
```

Figure 4: The parallel algorithm presented by Giest and Romine.

Where a_{ij} is an element in the i^{th} row and j^{th} column of the coefficient matrix a,

l is a temporary vector housing the pivot column

n is the dimension of the matrix

Implementing the algorithm in figure 4 on a network of transputers requires that, for good load balancing, the columns of the matrix be wrapped onto the processors (i.e for a network of 16 processors the first processor receives columns 1, 17, 33 etc the second processor gets columns 2, 18, 34 and so on). The reason for the wrapped column mapping is that once a processor has operated on the pivot column (column k), this column is not used in any calculation for the completion of the algorithm. The work load of the processor is therefore reduced. Thus to ensure that the processors each have equal loads throughout the execution of the algorithm, a wrapped column mapping is employed.

The communication between processors during the computation involves only the broadcasting and reception of the pivot column. On completion of their individual tasks, the processors return their respective columns to the host processor. In this manner the matrix is triangularized on the transputer network.

Other methods of solving matrix equations such as the conjugate gradient or banded matrix approximations have not been considered during this study. Such methods may well improve code execution, especially on parallel processors and deserve careful consideration in future attempts.

3.3 NEAR AND FAR FIELDS

The calculation of the near and far fields are ideally suited to evaluation in parallel. This has not been done in the version of the parallel NEC2 discussed in this paper but a quick explanation of how this may be done follows :

Consider, for example the calculation of the radiation patterns, where the far fields at a number of points are evaluated from the currents on each segment. Parallelizing this code may be accomplished by specifying that each transputer calculate the field at specified points in the radiation pattern. Alternatively, if there are many radiation patterns to compute, designating a complete radiation pattern to each processor may be a more efficient solution.

The only information required by the network processors to do the field calculations would be the current in each segment of the structure. The geometry of the structure would already exist on the processors since it would have been required by the matrix filling algorithm.

3.4 MEMORY REQUIREMENTS FOR THE NETWORK

The minimum memory required for p processors on the network is 1/p of the memory required for an in-core solution on one processor. The reason for this minimum requirement is that the network processors do not have access to a storage device, and thus have to be able to store 1/p of the matrix in memory.

The host processor memory requirements are slightly different. If an in-core solution is required then the host processor must have enough memory to store the entire matrix. It is possible however, to use an out-of-core solution. Thus if the processor does not have enough memory to store the matrix, then the matrix may be stored on disk.

4 THE TRANSPUTER NETWORK

The aforementioned two parallel sections of code for the Method of Moment solution both have some communication characteristics in common. They both require initialisation by the host transputer and both return portions of the matrix to the host. The triangularization algorithm requires the ability to broadcast information from one processor to every other processor in the network. Thus in choosing a suitable network on which to run the foregoing algorithms two important points have to be taken into consideration. First, it is important to ensure that the communication path from one processor to any other is minimised and secondly, that the algorithm controlling the communication is kept as simple as possible. With these points in mind the network mapping shown in figure 5 was employed[4] :

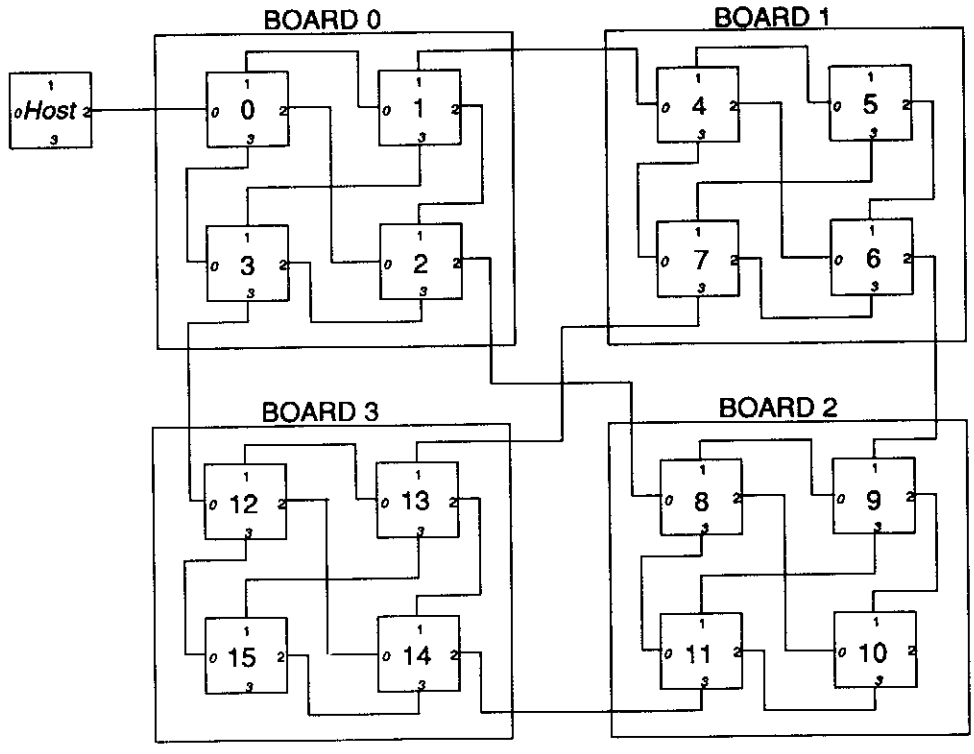

Figure 5: The transputer network.

The maximum number of processors through which one has to pass in order to communicate with any other processor is two. This figure is one less than the maximum required on other structures considered in choosing a suitable network, for example, the hypercube. The algorithm for sending information from one processor to any other is as shown in figure 6:-

```
IF (recipient on my.board)
    send message on link (recipient.id REM[1] 4)
ELSE
    send message on link (recipient.id DIV[2] 4)
```

[1] The expression ***a REM b*** yields the remainder of $\frac{a}{b}$

[2] The expression ***a DIV b*** yields the truncated value of $\frac{a}{b}$

Figure 6: The algorithm used to control the communication in the transputer network.

where ***my.board*** is found using (***my.id DIV 4***) and the recipients' board number is found by evaluating (***recipient.id DIV 4***).

In the case where there is no direct path to the recipient, the above algorithm is repeated on the intermediate processors and the message is redirected.

5 RESULTS

In assessing the performance of an algorithm on a network of transputers it is useful to compare the time taken to complete the task on the network to the time taken on one processor. Thus in gauging the performance of the parallel versions of the matrix filling and the triangularization, the efficiency and speed up of these processes were calculated.

Efficiency and speed up are defined as follows :-

$$\textbf{\textit{Efficiency}} = \frac{time\ taken\ to\ complete\ task\ on\ one\ processor}{(time\ taken\ to\ complete\ task\ on\ p\ processors) \times p}$$

$$\textbf{\textit{Speed Up}} = \frac{time\ taken\ to\ complete\ task\ on\ one\ processor}{time\ taken\ to\ complete\ task\ on\ p\ processors}$$

where the time taken to complete the task on p processors is made up of the time spent communicating between processors and the time spent doing the computation.

Tables 2 and 3 show the times taken to fill and factor matrices of various dimension. It should be noted that the speed up attained for smaller matrices is considerably less than the speed up attained for the larger matrices. The reason for this is that the ratio of communication to computation is larger for the smaller matrices.

Matrix Dimension	Filling Time One Processor (secs)	Filling Time 16 Processors (secs)	Speed Up	Efficiency in %
50*50	8.1	0.9	9	56
100*100	31	2.5	12.4	77
200*200	119	8.9	13.4	83
300*300	271	19	14.3	89

Table 2: Times taken to fill the matrix.

The times taken to simulate structures of various electrical sizes (matrix dimension) with NEC are tabulated in table 4. These times include all the serial components of NEC, for example, the manipulation of the input data and the solution of the matrix equation.

Matrix Dimension	Factoring Time One Processor (secs)	Factoring Time 16 Processors (secs)	Speed Up	Efficiency in %
50*50	0.98	0.16	6	38
100*100	7.5	0.85	8.8	55
200*200	59	5.1	11.6	72
300*300	197	15.5	12.7	80

Table 3: Times taken to factor the matrix.

Number of Segments	Time to Simulate on one Processor (secs)	Time to Simulate on 16 Processors (secs)	Speed Up	Efficiency in %
50	10.14	1.98	5.1	32
100	45.3	5.17	8.7	54
200	182	18.0	10	63
300	476	45	10.6	66

Table 4: The times required to simulate structures of various electrical sizes.

In the Stellenbosch paper[2] announcing the execution of NEC on a single T800 transputer, a comparison was made between the execution times of the NEC code on various computers. The simulation used as a benchmark was the 12 element Log Periodic Dipole Array given at the back of the NEC user manual.

This simulation is not a good benchmark for the transputer network for a number of reasons. First, the simulation requires the use of the transmission line card. This facility has not yet been implemented in parallel and it thus considerably increases the serial component of the simulation. Also the simulation uses only 78 segments and the transputer network efficiency with this number of segments is about 35% as opposed to an efficiency of 66% when simulating structures of 300 segments. A comparison of the times taken in the various sections of the program for the serial (1 transputer) and parallel versions are given in table 5.

Bench mark results for the NEC2 code on the CONVEX C210S computer equipped with 128 MBytes of core memory has recently become available. This computer contains a vector processor and the code was optimized by the vendors of the machine to take the best advantage

Program Tasks	Serial Version (secs)	Parallel Version (secs)	Comment
Processing data cards	1.17	1.17	Serial
Processing control cards	0.33	0.33	Serial
Filling matrix	17.1	1.9	Network Efficiency 42%
Factoring matrix	5.67	0.8	Network Efficiency 33%
Calculating effect of transmission line	**5.88**	**5.88**	Serial- takes 50% of the parallel NEC time.
Back substitution	0.26	0.26	Serial
Radiation pattern	1.3	1.3	Serial
Total time	31.7	11.6	Efficiency of 17%

Table 5: The times taken in the various stages of the simulation of the Log Periodic Dipole Antenna using the parallel and serial versions of NEC.

of this processor. The CONVEX is a multi-user Mainframe computer but results quoted were obtained with only one user present. Results for this machine are compared to those obtained using a 16 transputer network for a 1000 segment NEC2 problem.

	C210S	16 transputer network
CPU Time (sec)	1812	770
Approx. cost	$ 0.5 to 3 Million	$ 40 000.00

Table 6: A comparison of the performance of the CONVEX C210S and a 16 transputer network.

6 CONCLUSION

The speedup attained by the 16 processor network indicates that it is possible to significantly reduce the execution time of NEC by distributing the program onto a network of processors and executing the code in parallel.

The transputer connection network was specifically designed to satisfy the communication requirements of the parallel fill and factor algorithms. The most attractive feature of the network is the ease and efficiency of communication between processors.

A parallel implementation may be achieved by substituting another processor for the transputer in the network. The transputer was chosen because its design allows easy extension of a network with a corresponding increase in performance and price. A transputer based machine is therefore a cost effective way of speeding up NEC.

7 REFERENCES

[1]Burke G.J., Poggio A.J., "Numerical Electromagnetics Code (NEC2) - Method of Moments," Naval Oceans Systems Center, San Diego, CA Tech Doc 116, 1981.

[2]Le Roux J.J., "Numerical Electromagnetics Computation using the INMOS T800 Transputer on an Olivetti M24 Personal Computer" Applied Computational Electromagnetics Society Journal and Newsletter, Vol. 3, No. 2, 1988, pp 88-94.

[3]Giest G.A., Romine C.H., "Parallel LU Factorization," SIAM J. Sci. Statist. Comput., Vol. 9, No.4, July 1988, pp 639-649.

[4]Faasen C.R., "An Efficient, Generalisable Transputer Network Configuraton and Routing Algorithm" University of the Witwatersrand, Internal report, 1990.