A Dedicated TLM Array Processor

D. Stothard, S.C. Pomeroy
Dept. of Electronic and Electrical Engineering, Loughborough University, Loughborough
Leicestershire, U.K, LE11 3TU

ABSTRACT. The transmission line matrix (TLM) method is introduced and the specific issue of computational efficiency is discussed. The implementation of TLM on parallel computers is studied leading to the creation of a highly efficient processor designed specifically for TLM. Limitations introduced by the connection strategies employed by most parallel architectures are overcome through the use of a novel data routing architecture. The basic idea is extended to include stub loaded and three-dimensional TLM. The development of a prototype processor is discussed and potential applications are given.

1 Introduction

Analytical solutions to Maxwell's equations are possible only in restricted cases. The advent of the digital computer has given rise to a number of numerical techniques for solving Maxwell's equations and modelling electromagnetic wave phenomena. Of these the most common are the finite difference time domain (FD-TD), finite element (FE), and transmission line matrix (TLM) methods[1]. Finite difference and finite element methods respectively perform differentiation or integration of the electric or magnetic field over a defined region. In contrast, TLM is built around an array of connected secondary radiators, giving discretisation in both space and time. Thus TLM bears a uniquely close physical resemblance to the processes of propagation. TLM in fact offers a solution to the Telegraphers equation and has been applied to both propagation and diffusion modelling.

This paper is divided in to eight sections. After this introduction, Section 2 gives a brief introduction to the TLM method and demonstrates the key factors leading to lengthy run times. Section 3 looks at the implementation of TLM through parallel and distributed computing methods and analyses the reasons for the worse than expected performance increases shown by such methods. Section 4 introduces a new scatter processor designed specifically for the two dimensional TLM method and discusses the implications of working with the new design. A new strategy for mapping a TLM array in to hardware is introduced in section 5. The processes developed in sections 4 and 5 are extended to cover stub loaded and three dimensional TLM arrays in section 6. Section 7 details a general processor utilising reconfigurable logic to optimise performance for each of the above TLM schemes within a single architecture. The design and testing of a prototype processor is discussed in section 8. The conclusions in section 9 look towards the future of the processor and suggest potential applications.

2 Review of TLM

A basic introduction to the workings of TLM is given here. Fuller, more rigorous derivations, and discussion of the relationship between TLM and other techniques are contained in references [1,2,3,4,5,6,7]

The transmission line matrix (TLM) method, first reported by Johns and Beurle[4] in the early 1970s, offers a simple and unconditionally stable method for realising time domain solutions to propagation and diffusion problems. The basic building block for the two dimensional (2D) TLM method for waves is the shunt node [5], formed by an orthogonal junction between two ideal transmission lines of length Δl, (Figure 1)

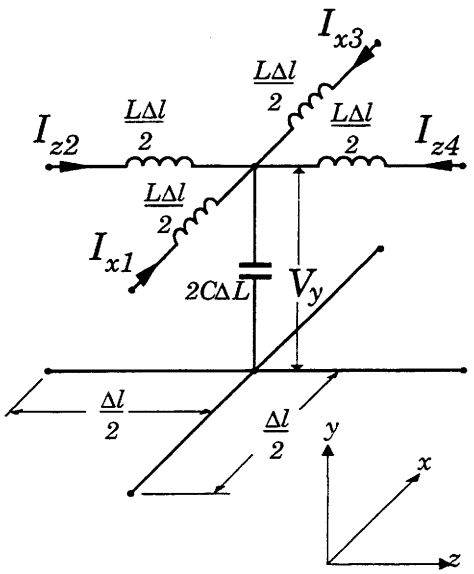

Figure 1 - Transmission Line Junction

This unit cell is repeated to fill the region under consideration with a Cartesian mesh of transmission lines.

An impulse travelling towards a node in the mesh will see an impedance mismatch at the junction and will be scattered according to (1).

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix}^r = \frac{1}{2}\begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}\cdot\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix}^i \quad (1)$$

where $V_{1\text{-}4}$ are the voltage impulses in the branches 1-4 and the suffixes i and r denote incident and reflected impulses respectively. (1) is more commonly expressed as

$$V_n^r = \frac{1}{2}\sum_{m=1}^{4} V_m^i - V_n^i \quad (2)$$

It can be shown[5] that there exists a direct relationship between the voltages and currents on the transmission lines and the electric and magnetic fields in the region modelled by the mesh. Impulses traverse the transmission lines with a fixed velocity, u, thus all impulses scattered from a node will become incident upon the neighbouring nodes after a time

$$\Delta t = \frac{\Delta l}{u}$$

The implementation of TLM in software follows an iterative process,

i) impulses are injected in to the mesh by exciting the appropriate voltages or currents.

ii) Equation (1) is applied to the incident data at each node using an instruction loop.

iii) A second loop passes the data scattered from each node to the neighbouring nodes for which it forms the incident impulses in the next iteration.

The addition of another node to the mesh requires one further application of the scatter and connect loops, thus processing time increases linearly with the model size.

It has been demonstrated [5] that the propagation velocity of a wave through the mesh is dependent upon the direction of travel and the mesh discretisation, Δl. Propagation at 45° to the axes is unperturbed, however axial propagation is frequency dependent, giving rise to dispersion. The plot of the dispersion characteristic shows that at least 10 nodes are required per wavelength at the highest frequency under consideration to reduce dispersion in the mesh to an acceptable level. This restriction upon Δl, along with the need for fine meshes to accurately model detailed geometries, can lead to very large meshes with many nodes. This produces a correspondingly large run time for TLM code on serial computers.

The basic scatter process of (1) may be adapted to model propagation in inhomogenous and lossy media through the addition respectively of capacitive or absorptive stubs of length $\Delta l/2$ to the node. Some of the energy at the node is scattered in to the stub and then, in the case of capacitive stubs, returned to the node in the next iteration. This changes the form of the scattering equation to

$$V_n^r = \left\{\frac{2}{y}\left[\sum_{m=1}^{4} V_m^i + y_0 V_s^i\right]\right\} - V_n^i \quad (3)$$

Where $y = 4 + y_0 + g_0$, y_0 and g_0 are the normalised capacitive and lossy stub impedances respectively. Because the stub energy is not passed to neighbouring nodes the connection process remains unchanged.

The most common scheme for three dimensional (3D) modelling is the symmetrical condensed node (SCN) [6] shown in fig. 2. The scattering matrix of the SCN is a sparse 12 x 12 matrix which has discrete solutions of a form similar to (2). As with 2D modelling, scattered data are passed to the neighbouring nodes.

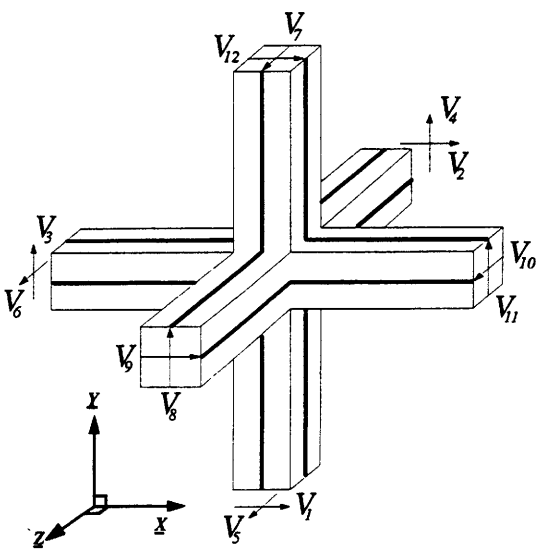

Figure 2 - The Symmetrical Condensed Node

3D modelling of inhomogenous media is possible through the addition of stubs to the SCN as with the 2D shunt node. However, recent developments have produced more computationally efficient 3D node schemes[7] such as the symmetrical super condensed node (SSCN). These schemes remove the need for some or all of the stubs, reducing the data storage requirements for each node and producing more efficient scattering algorithms with simple, discrete solutions.

The introduction of more complex arrays for inhomogenous media and 3D meshes add further to the run times for TLM. 3D modelling generally requires considerably larger arrays with more nodes than 2D modelling. The addition of stubs to the array complicates the scattering process and increases the number of operations required to perform each scattering operation, thus increasing the time taken for each loop of the scatter process.

3 Review of Parallel Implementations

The scatter and connect processes in TLM are explicit operations. To perform the operation at each node requires only data from that node. This means that the processes may be applied simultaneously to each node without conflict. This inherent parallelism has been exploited in the past by implementing TLM through parallel computing[8,9,10,11,12,13]. A variety of processing elements and connection strategies have been used. However a number of features are common to most or all of the implementations.

i) There is a direct physical mapping of the mesh to the parallel processor, i.e. each processing element maps to one node in the mesh. Interconnections are generally simple near neighbour links.
ii) The simple nature of the TLM algorithm leaves much of the processing power of the parallel processors unused.
iii) Limited bandwidths and TLM's low ratio of computation to I/O cause communication bottlenecks, reducing performance.
iv) The hardware requirements of each of these methods place them beyond the reach of most researchers.

Whilst it is clear from the literature that implementing TLM on a parallel computer can produce significant performance increases it is also clear that modification of the nature of the implementation could produce further performance improvements. There are 3 key points that must be addressed.

i) The direct mapping of one node in the mesh to one processing element limits the size of the model to the number of processing elements available.
ii) The granularity of the processing elements must match that of the problem for efficient performance.
iii) The bandwidth provided must be sufficient to prevent bottlenecks during the connection phase.

The best way to ensure a granularity match is to use a processing element designed specifically for TLM. The use of application specific processors to provide efficient computing performance has grown rapidly in recent years. There have been several application specific TLM processors designed in the past, these can be placed in to two categories.

i) Single node coprocessors.
ii) Complete arrays.

The former approach, developed by Saleh[14], uses a single node which is utilised as a coprocessor by the software. Each time the software encounters a scattering operation the incident data is passed to the coprocessor and the scattered data is returned to the host. Scattering remains a serial process, each node is treated individually, and the performance increase comes from the efficiency of the reduced instruction set (RISC) architecture of the node processor. This approach has the advantage that the array size is limited only by memory availability.

The latter approach, as used by Gregory[15], provides an array of RISC processors on to which the TLM mesh is mapped. The host system provides initial data and reads out results from the array. Scatter and connect are performed in parallel providing a significant performance increase, however the main advantage of the system is in the efficiency with which each scatter computation is performed. Each processing element has been designed to perform only the TLM algorithm and is therefore fully utilised the whole time. The application specific approach, while more efficient, has the disadvantage that the processor may be limited to one type of TLM calculation, e.g. stub loaded 2D.

4 Design of a Scattering Processor

When developing a new application specific processor for TLM it is possible to consider the scatter and connect processes separately. The choice of architecture, i.e. coprocessor or complete array is important. It appears that the

array architecture provides higher throughput as it performs some operations in parallel, however this is offset by the higher bandwidth and hardware requirements. The design of the processing elements which perform the scattering operations will go a long way towards deciding which is the most efficient architecture.

The design process begins with the development of a suitable algorithm for hardware implementation. Numerical devices such as multipliers are difficult to implement in logic and can lead to large, slow circuits where as adders and subtractors are easily constructed. Nodal schemes such as the shunt node and the SCN require only additions, subtractions and a divide by two, which may be implemented by shifting in binary, therefore for these schemes the most suitable algorithm is the one which minimises the number of addition and subtraction operations. The modelling of variable media using either stub loading or the SSCN requires multiplication. Because the multiplier is the dominant component, minimisation of the number of multiply operations becomes the overriding concern.

The optimised algorithm must be developed in to a suitable hardware configuration. This process is simplified through the use of behavioural modelling and the VHDL hardware description language. This allows a circuit to be described in terms of its behaviour (in this case the chosen TLM algorithm) and its performance simulated. Tools are then available to synthesise a gate level circuit description from the behavioural VHDL. This circuit may again be tested through simulation before production takes place. A logical starting point is the design of a simple two dimensional TLM system.

The design of the scattering processor is a trade off between a number of conflicting requirements. Should the processor have a RISC architecture or should the scatter process be mapped directly to the hardware? The former provides more flexibility but the latter will be faster. Flexibility is a key issue; although most of the mesh is homogenous, boundaries, sources and targets all require handling differently. Specialised nodes for boundaries etc. are one solution, however these would either be placed at fixed locations within the mesh or would require a complex routing procedure to allow arbitrary placement. A more viable solution is a generalised processor which can act as a simple scattering point, a source, target or boundary node as required.

An early attempt by the authors to design a TLM scatter processor is documented in [16]. This simple design was a direct mapping of the 2D scatter equation on to a field programmable gate array (FPGA). Despite producing very high throughput the design failed in a number of key areas.

i) The processor would only perform a simple scattering operation. Boundaries etc. were untreated.
ii) The data parallel design requires a very high bandwidth.
iii) There is no access to data within the array of processors, only data reaching the edge of the array can be read. Similarly there is no access to the total incident energy data commonly used to visualise propagation within the mesh.
iv) The word length used is fixed by the width of the logic.
v)

In order to overcome these problems a new design has evolved which utilises a pipelined, bit serial architecture. A block diagram of the processor is shown in fig. 3.

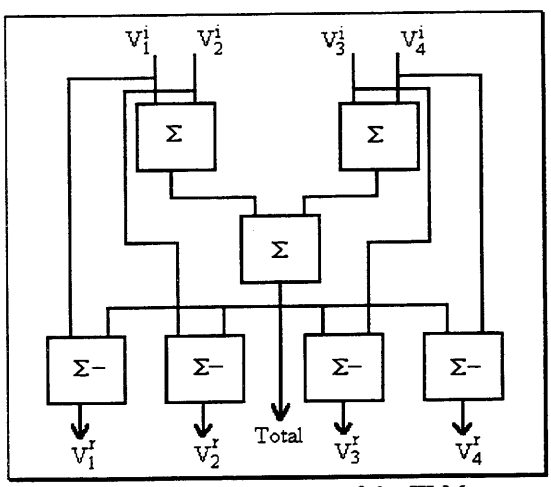

Figure 3 - Block Diagram of the TLM Processing Element

Σ indicates a single bit wide full adder, Σ- indicates a single bit wide subtractor. The operation of the processor is simple. The first two levels add together the input data and the result is divided by two by discarding the first bit output from level 2. The sum is then routed to four subtractors which produce the output values. A single pin output provides access to the total incident energy data. By varying the timing of the control signals this processor is capable of operating on data of any word length. The bit serial design also reduces the

required bandwidth considerably. The design has one further advantage in that it is able to perform simple boundaries (those with reflection coefficients of ρ = 0, 1, -1). This is done by incorporating the boundary in to the scatter equation thus

$$V_m^r = \rho\left\{\frac{1}{2}\sum_{n=1}^{4} V_n^i - V_m^i\right\} \qquad (4)$$

The data is preceded by a two bit code which is used to define what type of boundary is present. The code is added to all data words allowing the arbitrary placement of boundaries within the TLM mesh.

5 Mapping the Connect Process

The connect process is required to provide near neighbour connection between the nodes in the TLM mesh. Previous systems have either performed this mapping in software or have produced a large physical array with hardwired interconnects between the processors on to which the TLM mesh is mapped. The former approach is slow whereas the latter requires a large number of processors and limits the size of the mesh which may be implemented. A more flexible approach would be to have an architecture which allows a mesh of arbitrary size to be mapped on to a fixed number of processors.

The localised nature of the connect process means that any given node can communicate only with the adjacent nodes in its own row of the mesh or with the adjacent nodes in the rows immediately above and below it. Thus only three rows of data are active at any given time. Scattered data will either be passed to a neighbouring node or, if a boundary is present, it will be returned to the node from which it was scattered. This information has been utilised to develop a hardware mapped connect process. Three small blocks of memory, called the active lines, are used, holding copies of data from three adjacent rows in the array. Data is scattered from the centre row of the three and the output from the processors is sent to a logic cell which reads the boundary flag attached to each data word and routes the data to either the correct adjacent node or back to the scattering node as appropriate. If the number of processors available are less than the number of nodes in each row of the model then another memory block is required to hold data scattered left and right from the edges of the row of processors until it is required.

From the scatter and connect processes described above it is simple to produce a complete system [17] for the solution of a two dimensional TLM mesh. A large main memory is required to store all the current data in the array, that is the four incident values at each node in the current iteration. Starting at one corner of the mesh data from the first N nodes (where N is the number of processors available) are passed to the scatter processors. These output the total energy incident upon the nodes, which may be read out to a host system or stored in a further memory, and the scattered data values which are passed to the connection logic and associated memory. At the end of each row processing moves on to the start of the next row and continues. Data held in the third row of the connection memory is no longer affected by the current scattering events and can be written back to the main store where it forms the incident data for the next iteration. A block diagram of the system is shown in fig. 4.

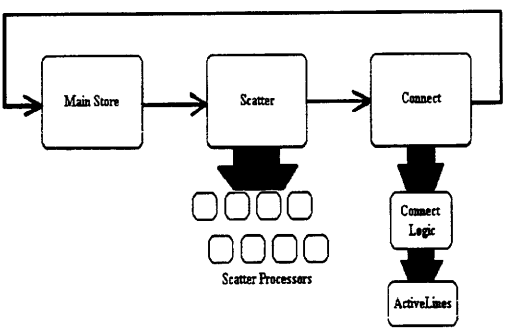

Figure 4 - Hierarchic Block Diagram of the TLM System

This unique mapping of the connect process allows a small number of processors to be mapped to a mesh of arbitrary size. This approach has the advantage of performing the scatter and connect processes partially in parallel, thereby increasing performance, but the mapping of a large mesh on to a small array of processors eliminates the restrictions on model size imposed on most parallel TLM applications.

6 Extending System Capabilities

The processor described above is capable of performing only basic 2D TLM on a homogeneous mesh, however the techniques it introduces may be developed to produce similar systems capable of more complex operations. An extension of the basic method to allow for stub loaded 2D modelling is straightforward and only requires modification

of the scatter processors. The stub loaded scatter processor is similar to that of the basic processor except for the addition of a multiplier which is preloaded with the value of y_0 for the next node while processing takes place on the current data. In addition to this modification a small amount of memory is required to hold both the energy in the stub at each node and the node parameters y and y_0. This data is not connected to neighbouring nodes, therefore the connect process remains unchanged.

The techniques used in two dimensional modelling may be extended to three dimensional modelling. The connect process in three dimensions can be seen as identical to the two dimensional connect process with additional data passed to nodes in the planes above and below the scattering node. This can be accomplished using the memory architecture shown in fig. 5.

The active lines and main store are identical to the active lines and main store in the 2D system. Data from the active lines is used to update the main store along with the data scattered in to the last plane from the scattering processors.

The scattering processor requires some modification. Three dimensional nodes are 12 port devices thus the processors require 12 input data words as opposed to 4 for the 2D node. The architecture of an SCN scatter processor is identical in its operation to the basic 2D scatter processor using pipelined layers of adders and subtractors to produce the output data. A further output

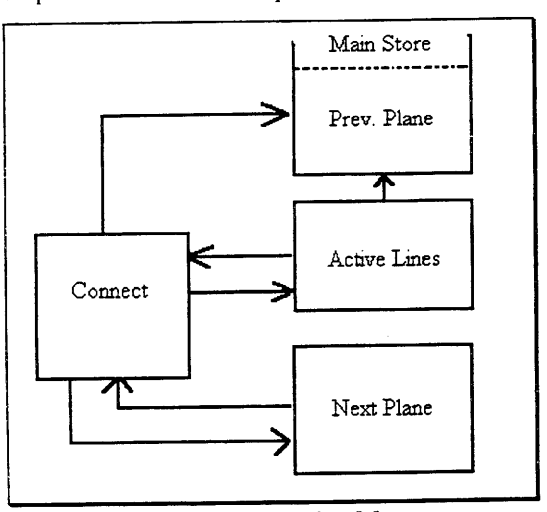

Figure 5 - 3D Connection Memory Architecture

provides access to one of the field components, selected before processing begins, for visualisation purposes. The SSCN, used for isotropic, inhomogeneous three dimensional media, requires the storage of two impedance values for each node. These can be treated in a similar way to the stub data for the 2D stub loaded processor, being held in a separate memory. As with the 2D stub loaded processor, preloading of the multipliers with the impedance values improves performance.

7 A General, Reconfigurable Processor

Consider the architecture of the processor required to perform 3D TLM using the SSCN, fig. 6(a). By utilising only the required components from this architecture we arrive at the processor architectures for the three other classes of TLM processor, basic (SCN) 3D, stub loaded 2D and basic 2D, fig. 6(b-d). This illustrates not only the close physical similarity between all TLM schemes, but also the fact that one architecture, with a little modification, should be capable of performing any of the four key TLM schemes. There are two ways of allowing the necessary modifications to the scatter and connect hardware, reprogrammability or reconfigurability. A reprogrammable processor decodes instructions and performs the operations dictated by its program. This approach is slower than the hardware mapped systems developed above as it requires instruction fetching and decoding cycles. This approach would also, naturally, introduce some redundancy as not all features are implemented in each scheme. A more suitable option is to use reconfigurable logic such as a field programmable gate array (FPGA)[18], which is configured at start up but may be given a different configuration each time it is used. The authors use Xilinx FPGAs as the granularity of the configurable logic blocks (CLBs) is well suited to the problem, reducing redundancy in the design. The system requires three reconfigurable components, scatter, connect and control. Using the architecture of figure 6(a) the most suitable processor configuration can be chosen for each particular problem, optimising performance. The operation of each processor is identical as far as the user is concerned, each processing scheme requires the same set of control signals. The configuration for each FPGA can be stored on EPROM or as a file on a host system, therefore it is possible to build up a library of common configurations which may be combined as necessary depending upon the given problem.

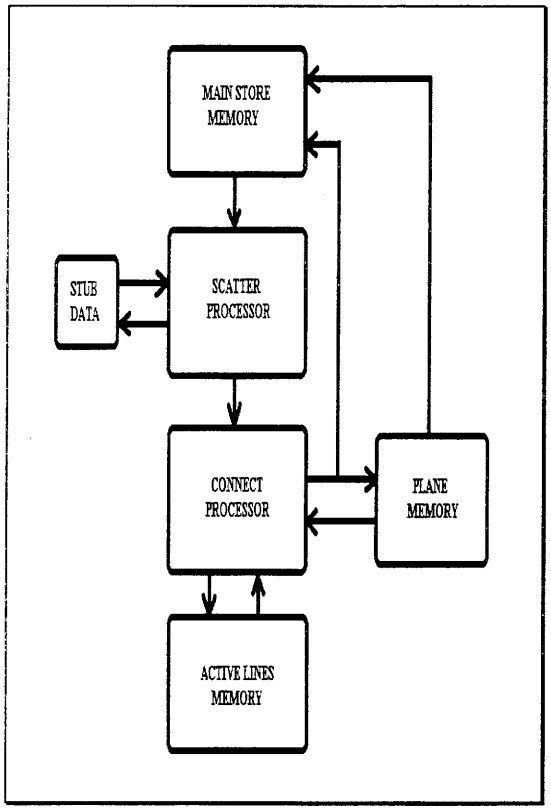

Figure 6a - SSCN System Top Level

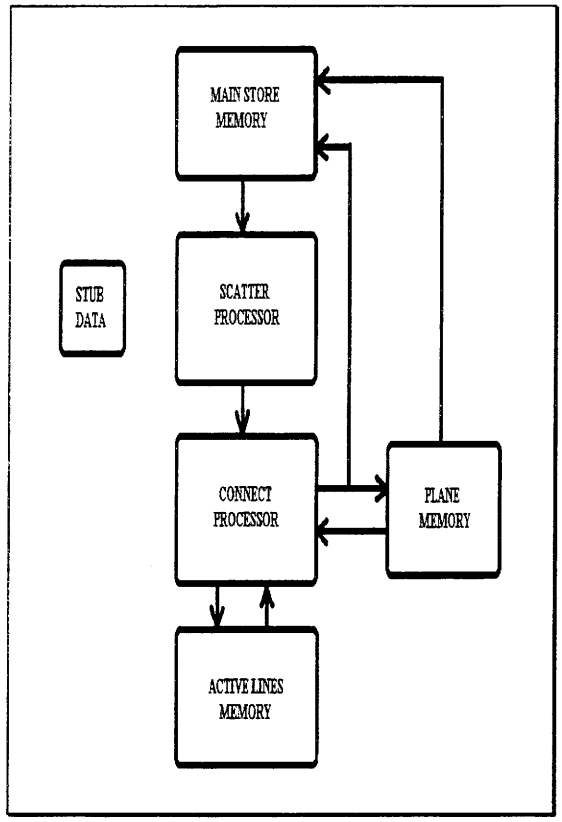

Figure 6b - SCN System

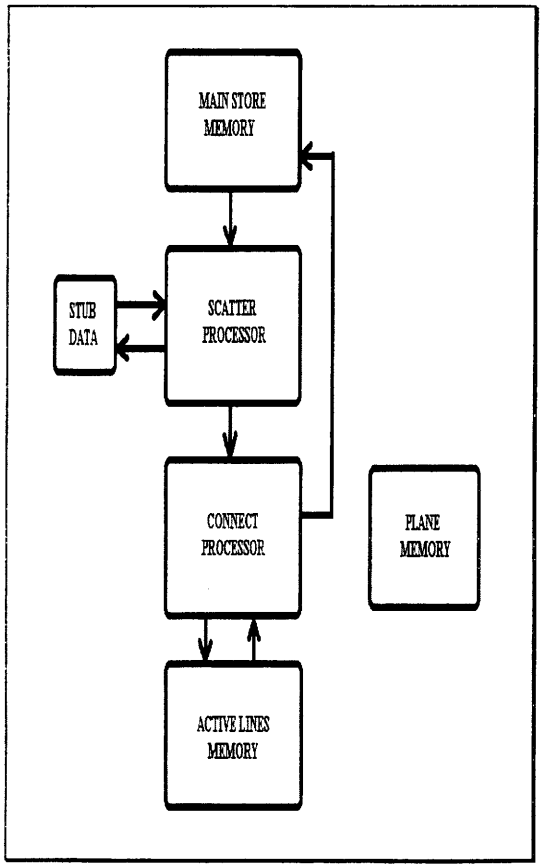

Figure 6c - Stub Loaded Shunt Node System

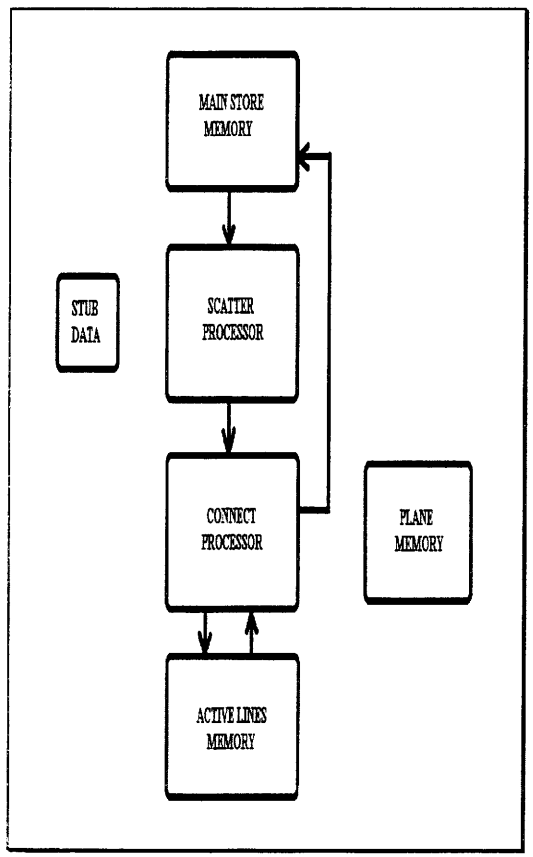

Figure 6d - Shunt Node System

8 Testing and Results

Modelling of the system using the VHDL hardware description language has allowed for extensive testing. Synthesis of several processor configurations realised using both behavioural and structural synthesis has allowed optimisation of each component in the system. The scattering processors have been tested on a Xilinx XC4013 and have demonstrated correct operation. A prototype system limited to performing basic 2D TLM is under construction. This will allow more rigorous testing of the system to be carried out and will provide useful performance measures. The prototype has taken the form of a PCI card which may be hosted on a personal computer. Initialisation data is fed in to the system which then runs in the background, allowing the host to perform other tasks. Output data may be read from the system either after every iteration or at the end of the processing run and access is provided to data from individual nodes in the model with little communication overhead. Predictions of current operating rates suggest that a scattering component with 8 processors will be capable of performing up to 8 million scattering events per second on 32 bit data. This compares favourably with $6x10^5$ scattering events per second achieved using serial code on a 200Mhz Pentium Pro equipped PC. This may be improved by partitioning the model between two or more boards and allowing them to operate concurrently. As with software implementations there is a linear relationship between run time and model size. This relationship is inversely proportional to the number of scattering processors available, hence increasing the number of scatter processors can have a pronounced effect on throughput. While the current prototype contains only 8 scatter processors a working system may contain several thousand.

9 Conclusions

The system has two potential areas of application, those where large arrays must be processed and those where smaller arrays must be processed quickly. Possible applications in the first group include EMC studies and ultrasonics/acoustics. The second type of application includes time critical operations such as medical imaging, real time remote sensing etc.

The unique nature of the connect process described above allows the system to operate on any algorithm which requires a simple near neighbour data transfer. The scatter processor is replaced with a processor capable of performing the new algorithm. These include diffusion modelling using a link line TLM scheme and non TLM applications such as FD-TD and cellular automata. Through the use of hardware description languages and logic synthesis the development of scatter processors for these other techniques is relatively simple. Indeed in some cases existing software routines may be transferred in to VHDL and thus directly in to hardware.

10 Summary

A complete application specific processor for the solution of the transmission line matrix (TLM) method has been presented which offers a considerable reduction in run times while overcoming the limitations imposed through conventional parallel architectures. A unique mapping of the TLM connect process to hardware allows a small number of scattering processors to process a mesh of any size. Through the use of reconfigurable logic the processor may be optimised to perform any of the four main TLM schemes, 2D basic, 2D stub loaded, 3D SCN, 3D SSCN without requiring any user reprogramming.

It is interesting to note that a processor developed entirely from the TLM equations may, through the use of reconfigurable computing, be applied to many other numerical modelling techniques.

A prototype system is under construction which is expected to provide throughput an order of magnitude greater than current software implementations while future, large scale systems may offer performance far beyond this point. Realisation of the system as a PCI card for a personal computer makes it an accessible, low cost alternative to traditional parallel computing.

References

[1] 'Numerical Techniques for Microwave and Millimeter-Wave Passive Structures' T.Itoh - Ed. J.Wiley, 1989

[2] Russer, "On the Field Theoretical Foundation of the Transmission Line Matrix Method", 1st International TLM Workshop, University of Victoria, Canada, 1-3 Aug. 1995

[3] Simons and Lovetri, "Derivations of Two-Dimensional Algorithms on Arbitrary Grids Using Finite Element Concepts", 1st

International TLM Workshop, University of Victoria, Canada, 1-3 Aug. 1995

[4]P.B. Johns & R.L. Beurle, 'Numerical Solution of 2-Dimensional Scattering Problems using a Transmission-Line matrix' Proc. IEE, Vol.118(9), pp.1203-08, 1971

[5] W.J.R. Hoefer 'The Transmission line Matrix (TLM) Method' in 'Numerical Techniques for Microwave and Millimeter-Wave Passive Structures' Chapter 8, pp.496-591, J.Wiley, 1989

[6] P.B. Johns 'New Symmetrical Condensed Node for Three Dimensional Solution of Electromagnetic wave Problems by TLM' Electronics Letters, Vol.22(3), pp.162-164, 1986

[7] V. Trenkic 'The Development and Characterisation of Advanced Nodes for the TLM Method' PhD Thesis, University of Nottingham, 1995

[8] P.P.M So; C. Eswarappa & W.J.R Hoefer 'Parallel and Distributed TLM Computation with Signal Processing for Electromagnetic Field Modelling' Int. Jnl. Num. Mod : Elect. Networks, Devs. And Fields, Vol.8, pp.169-185, 1995

[9] J.L Dubard; O. Benevello; D. Pompei; J. Le Roux; P.P.M So & W.J.R Hoefer 'Acceleration of TLM Through Signal Processing and Parallel Computing' Int. Conference on Computation in Electromagnetics, IEE, pp.71-74, 1991

[10]P.O Luthi; B. Chopard & J-F Wagen 'Wave Propagation in Urban Microcells : a Massively Parallel Approach Using the TLM Method' Applied Parallel Computing in Physics, 2nd International Workshop, pp.408-18, 1995

[11] C.C Tan & V.F Fusco 'TLM Modelling Using an SIMD Computer' Int. Jnl. Num. Mod : Elect. Networks, Devs. And Fields, Vol.6, pp.299-304, 1993

[12] P.J Parsons; S.R Jaques; S.H Pulko & F.A Rabhi 'TLM Modelling Using Distributed Computing' IEEE Microwave and Guided Wave Letters, Vol.6(3), pp.141-42, 1996

[13] A. Mallik 'Parallel Executing TLM Code and it's Application to EMC and the Motor Vehicle' IEE Colloquium on EMC and the Motor Vehicle, pp.3/1-4, 1990

[14] A.H. Saleh 'A Dedicated Processor for Solving TLM Field Problems' PhD Thesis, University of Nottingham, 1982

[15] Gregory, S 'Design of a Single Bit Processor for TLM Using Full Custom IC Design' Dissertation (BEng), University of Nottingham, 1989

[16] D. Stothard & S.C. Pomeroy 'An Application Specific Processor for TLM' 1st International TLM Workshop, pp.277-280, University of Victoria, Canada, 1-3 Aug. 1995

[17] D. Stothard & S.C. Pomeroy 'An Application Specific TLM Array Processor' TLM - the Wider Applications, pp.3.1-3.5, University of East Anglia, Norwich, 27 June, 1996

[18] 'The Programmable Logic Data Book' Xilinx Inc, 1994