Chapter 6

Summary and Conclusions

In this work, a new frequency domain TLM approach was introduced which combines the superior features of both the time domain and the frequency domain TLM. The approach is based on a steady state analysis in the frequency domain using transient analysis techniques and hence was referred to as TFDTLM. In this approach, the link line impedances are derived in the frequency domain and are chosen to model the frequency dispersive material parameters. The impedances and propagation constants are allowed to be complex and consequently provide more accurate modeling for wave propagation in a frequency dispersive medium. At the same time, the TFDTLM has the advantage of being able to extract all the frequency domain information in the frequency range of interest from only one simulation. This special feature of the TFDTLM makes it computationally more efficient as compared to any other frequency domain TLM.

The new approach in addition to being computationally efficient as compared to other frequency domain TLM methods, was found to have superior dispersion behavior in modeling lossy inhomogeneous media as compared to time domain TLM . A first order approximation filter was able to model inhomogeneous media with a relatively high dielectric constant and /or relative permeability. A second order filter was able to provide enough accuracy in modeling a lossy inhomogeneous medium with a relatively high loss tangent. It is also worth mentioning that one important advantage of the approximation filter is that it can approximate a general frequency dispersive constitutive parameter, a feature that can not provided by a traditional time domain TLM scheme with open circuited, short circuited and lossy stubs.

Another important advantage of the TFDTLM, is that it can easily be interfaced with any time domain TLM method. Therefore in lossless regions with relatively low relative dielectric constants and/or permeabilities, a traditional time domain TLM technique can be used. The only drawback is that even in regions using the time domain TLM, all the computations must be complex. However, no approximation filter will be used in these regions which would consequently save a lot of computations and help improve the overall computational efficiency of the TFDTLM.

The TFDTLM can also be easily interfaced with any of the absorbing boundary conditions originally developed for time domain TLM with the slightest modifications. The absorbing boundary was found to perform even better with the TFDTLM than with a time domain TLM for reasons discussed in chapter 5.

For the purpose of verification, the TFDTLM was implemented in a three dimensional mesh. Some structure were simulated and the ability of the TFDTLM to accurately model wave propagation in lossy inhomogeneous media was demonstrated.

On The Computational Efficiency of the TFDTLM

It has been mentioned earlier that as compared to a frequency domain TLM scheme, where the intensity of computation per frequency is approximately of the same order, the TFDTLM would be computationally more efficient. The reason is that in the TFDTLM , all the frequency domain information in the entire frequency range of interest can be extracted from only one simulation. In a traditional frequency domain TLM on the other hand, the simulation has to repeated at every frequency point.

As compared to a time domain TLM scheme, the TFDTLM may be less efficient. The reason is that in a TFDTLM, all the computations must be complex. Also, more complex computations are used for the implementation of the approximation filter. In what follows, the number of multiplications and additions in the TFDTLM will be compared to that required in a time domain TLM. The storage requirement will be compared as well. The following calculations for a time domain TLM will be based on a general node having six different link line impedances for different coordinate directions and polarizations, three lossy stubs and three inductive or capacitive stubs. For the TFDTLM, six different link line impedances are considered with no lossy, inductive or capacitive stubs. For the TFDTLM, all the filter coefficients are normalized to b_0 i.e. b_0 is taken to be unity. The following table compares the number of multiplications and additions required by a TFDTLM node and the time domain TLM node.

Table 6.1 Comparison of the number of multiplication and additions in a first and second order TFDTLM and a time domain TLM

	TD TLM	TFDTLM 1st order	TFDTLM 2nd order
complex multiplications/divisions	-	66	78
real multiplications/divisions	57	-	-
complex additions/subtractions	-	48	60
real additions/subtractions	57	-	-
Equivalent total number of real multiplications / divisions	57	264	312
Equivalent total number of real additions/subtractions	57	294	354

From the above table, it appears that the number of multiplications in the TFDTLM is approximately 4.5 times that required by a time domain TLM for a first order filter approximation and is about 5.5 times for a second order filter approximation. It is important to note that in the above calculations every complex multiplication is converted to 4 equivalent real multiplications and three equivalent real additions. In order to improve that efficiency of the TFDTLM, it was important to notice that part of the multiplications involved in the TFDTLM are not full complex i.e. are not multiplication of two complex numbers but rather multiplication of a real and a complex number. Multiplication of a real and a complex number is equivalent to two real multiplications. The complex class developed in C++ could actually differentiate between these two types of multiplications and consequently save a lot of computations. The following table summarizes the total number of multiplications and additions for a first and second order efficiently coded TFDTLM as compared to a time domain TLM.

Table 6.2 Comparison of the number of multiplication and additions in a first and second order efficiently coded TFDTLM

	TD TLM	TFDTLM 1st order	TFDTLM 2nd order
Equivalent total number of real multiplications / divisions	57	168	216
Equivalent total number of real additions/subtractions	57	150	210

From the above table, it appears that by only differentiating between multiplication of two complex numbers and multiplication of complex number by a real number in a TFDTLM code, the efficiency of the TFDTLM can be significantly improved. In this case, the number of multiplication in a first order TFDTLM is 3 times that required by a time domain TLM . For a second order TFDTLM the number of multiplications becomes approximately 3.6 times that required by a time domain TLM.

It is also important to note that, in the region which is treated as a reference medium in a TFDTLM, the connection procedure does not involve any approximation filter. This would drop the equivalent number of real multiplication in this region to 98 and the number of equivalent real additions to 72. Consequently, the overall number of multiplications and additions will be reduced.

Concerning the memory storage requirement of a TFDTLM, a first order TFDTLM will require 24 complex memory locations per node as opposed to 15 real memory locations in time domain TLM with three stubs. A second order TFDTLM on the other hand will require 36 complex memory locations per node.

Actual CPU times were calculated for some of the results obtained using the HSCN and first and second order TFDTLM. All simulations were performed on an IBM Scalable Parallel Processor (SP2) system at Virginia Tech. In the simulation of the rectangular waveguide structure with 10 cells along the x and y directions and 20 cells along the z direction, the simulation was run for 4000 iterations. The average execution time per node per iteration is calculated by dividing the overall execution time by the total number of nodes and by the total number of iterations. The average execution time per node per iteration for the HSCN was found to be 22.1 μs, 120 μs for a first order TFDTLM and 155 μs for a second order TFDTLM. These results indicate that the first order TFDTLM required about 5.4 times the time required by the HSCN whereas the second order TFDTLM required about 7 times the time required by the HSCN. In the simulation of the rectangular waveguide bandpass filter with 20 cells in the x and y directions and 40 cells along the z direction for 4000 iterations, the average execution time per node per iteration for the HSCN was 40 μs, 131 μs for a first order TFDTLM and 169 μs for a second order TFDTLM. It is important to note that for this simulation, the structure became more complicated, the average execution time per node per iteration for the HSCN was almost doubled. This can be attributed to the fact that in the presence of a lot of discontinuities represented in this case by the inductive irises, the time spent in verifying the existence of a discontinuity and applying the appropriate boundary condition became much larger than that required for scattering in a region with no discontinuities. On the other hand, for the TFDTLM, the average execution time per node per iteration was increased by the presence of more discontinuities but only slightly. The reason is that the execution time even in a region free from discontinuities in a TFDTLM was much larger than that in the HSCN. Consequently, the presence of discontinuities did not significantly affect the average execution time. In this case, the first order TFDTLM required about 3.3 times the time required by the HSCN whereas the second order TFDTLM required about 4.2 times the time required by the HSCN