ABSTRACT

UPPATHIL, SATISH VIJAYAN. Layout Oriented Design Practice for Capturing Distributed Effects in High-Speed Circuits. (Under the direction of Michael B. Steer.)

An integrated environment for layout-oriented design of circuits for spatial power combining system is aimed in this study. The simulation environment would include a full-wave electromagnetic simulator and a circuit simulator, TRANSIM, with a front engine, which is the major emphasis in this document. Also, the controlled and independent sources based on the Hspice models are implemented in the state-variable based object-oriented circuit simulator, TRANSIM. Transim implements local reference node instead of a global ground and this is crucial for distributed circuits. The models are implemented in object-oriented fashion and uses automatic differentiation. The same model can be used for DC, transient and harmonic balance analysis.

Layout Oriented Design Practice for Capturing Distributed Effects in High-Speed Circuits

by

SATISH VIJAYAN UPPATHIL

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of

Master of Science

Electrical Engineering

Raleigh

2002

Approved by

Prof Griff Bilbro	**Dr Gianluca Lazzi**

Prof Michael Steer, Chair of Advisory Committee

PERSONAL BIOGRAPHY

Satish Vijayan Uppathil was born September 10, 1975 in Thrissur, Kerala, India. He received his elementary and secondary education in Orissa, India, graduating from Kendriya Vidhyalaya No.1, Bhubaneswar in 1994. He received the Bachelor of Technology degree with a major in Computer Science and Engineering from the University of Kerala-Government College of Engineering, Thiruvanathapuram in 1999. From then until July 2000 he worked for the Tata Consultancy Services in Thiruvanathapuram and Chennai as Assistant Systems Engineer. In August 2000 he was admitted to North Carolina State University to begin studying for the Master of Science in Electrical Engineering. While pursuing the MS degree he held a Research Assistantship with the Electronics Research Laboratory in the department of Electrical and Computer Engineering. His interests include Analog Circuit design for Radio Frequency and Wireless Applications. He is a member of the Institute of Electrical and Electronics Engineers.

ACKNOWLEDGEMENTS

Graduate studies at NC State have been a great experience and I would like to thank all people who made my two years of studies enriching and wonderful.

First, I would like to thank Prof Michael Steer for his help, encouragement and guidance in my graduate studies, research work and thesis preparation. He has been instrumental in making me an Electrical Engineer. Thank you very much.

I would like to thank Prof Griff Bilbro and Dr Gianluca Lazzi for being on my committee, reviewing my thesis and suggestions.

Thanks to Prof Paul Franzon for giving an appointment, which eventually turned out to be a rewarding experience and career.

I thank Dr Carlos Christoffersen for showing the patience and willingness to introduce me to Transim and sit through long hours of doubts and clarifications.

Thanks to all group members, present and past, in EGRC 410 and 412 for making the research work a pleasant learning experience. Discussions with Jayesh Nath is one of the few reasons I wished graduate studies could go on forever.

Finally, I would like to thank my parents and family for making me what I am, today. I do not have enough words to thank them.

My work has been supported by the US Army Research Office. I am grateful for their support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation for and Objective of this study

The need for Computer Aided Design (CAD) tools is ever increasing. With high-end processors and memory becoming cheaper, the simulation time has also decreased. And, with use of powerful algorithms it is possible to model several electrical circuit systems and simulate their behavior in a reasonable amount of time by developing software code.

Microwave, millimeter wave and radio frequency circuits are usually distributed in nature, which necessitates electromagnetic modeling and simulation in addition to circuit simulation. Thus the full application of computer-aided engineering (CAE) to these circuits requires the integration of electromagnetic models of distributed structures with conventional circuit analysis. Quasi-optical (QO) system, shown in Fig. 1.1 is typical example of such a system.

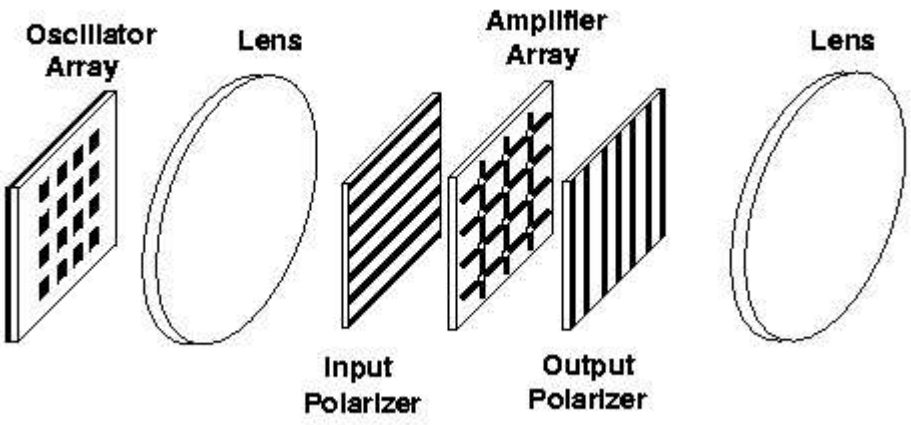

Figure 1.1: Quasi-optic power combining system.

In a QO system, a large number of active devices are distributed on the grid surface. The grid is excited by a horn and lens system that concentrates the incident field on the grid and polarizers are used to isolate the input and output. The power from numerous solid-state devices is combined in free space over a distance of many wavelengths. By using lenses fields are focused in a single paraxial mode. An oscillator array is the millimeter-wave source. The oscillator array is in an open cavity resonator that allows power combining to take place. The amplifier array between the lenses provides the required gain. A major advantage of this type of a system is that energy is guided through free space and hence losses in metallic interconnects which are large at high frequencies are avoided.

Applications of these systems include: near vehicle detection radar (collision

avoidance radar), millimeter-wave LANs (60 GHz), cellular radio base stations, active missile seekers (94 GHz), and millimeter-wave imaging (100+ GHz).

Electromagnetic models are required to design these spatially distributed systems. In the nascent stages of the CAE of microwave circuits, ports were used in specifying the interconnectivity of networks. The utilization of ports avoided the issue of specifying reference nodes. In analysis, using matrix manipulations or signal flow graphs, one of the terminals of a port was used implicitly as a reference node and generally ignored in formulating the mathematical model. However, a spatially distributed circuit does not have a global reference node as required in a conventional nodal analysis. This is because, a global reference node cannot reasonably be defined for two spatially separated nodes, since the electromagnetic field is transient or alternating making the electric field non-conservative. This also includes the case of two separated points on an ideal conductor. If we look at this in a time-domain reference, it takes a finite amount of time for the state at one of the points on the ideal conductor to affect the state at the other point. Consequently, no two separated points in space are instantaneously coupled in time. This necessitates an approach different from the conventional circuit analysis: requirement of local reference node for each group of devices that are electrically not distributed. Nodal specification provides tremendous flexibility in circuit specification but the multiple reference nodes leads to N-fold indefiniteness of the (modified) nodal admittance matrix. Also, with spatially distributed circuits it is possible to make

illegal connections such as connecting a lumped element, e.g., a resistor, across a spatially distributed element.

The motivation for this thesis is to work towards an integrated modeling environment for such systems. The main objective was to develop an interface which can handle the layout and schematic of a QO system, pass the netlist for non-linear circuit analysis to Transim circuit simulator and the geometry information to the EM simulator, in the appropriate format. Also, the controlled and independent sources were written into Transim for completeness.

More in terms of a long term goal the basic idea is to create a tool package which can be used in the public domain and would allow the user to draw and analyze EM structures and also perform circuit analysis with TRANSIM: a step towards the study of Circuit-Field Interaction.

1.2 Thesis Overview

Chapter 2 gives adequate background of materials used in the rest of the thesis. This includes Transim, Electric Editor, and Local Reference Node concepts.

Chapter 3 describes the method of implementation of the ideas of local reference nodes and matrix formulation, problem formulation and implementation.

Chapter 4 mainly presents some simulation results and comparisons with previous work.

Chapter 5 gives a summary of the thesis with conclusions and suggestion for future work on this topic.

Appendix A gives some information about the Caltech Intermediate Format used widely to port geometric information.

Appendix B describes the controlled and independent sources implemented in Transim.

Chapter 2

Background

2.1 Introduction

Computer-aided design and modeling provides efficient ways to evaluate the performance of a system before actually building the system. Various CAD tools are available to a user for modeling several different systems and circuits. For a quasi-optical system, which is a combination of microwave and optical units, the requirement from the CAD tool is different from that put by other analog or microwave circuits. For example this system consists of a resonator, pair of optical lenses and a grid amplifier array. Each of these units needs to be modeled by both an EM simulator as well as a circuit simulator. The main objective of this chapter is to discuss the tools required for analysis of the system, the underlying principle in their implementation, and the options.

2.2 Nodal Analysis

A circuit is a graphical construct bringing together algebraic and first order differential equations. These equations come from the constitutive relations of the

individual elements, which specify the actual form of the individual equations, and form Kirchoff's current law specifying how the equations are coupled. This is called the nodal admittance analysis. Circuits do not have a sense of space – a circuit is defined as though it existed at a point. In order to incorporate an electromagnetic model of a structure that is inherently distributed in space, two different approaches can be taken. One, to insert the device equations into an appropriate time-stepping electromagnetic simulator. This reduces the level of abstraction of the circuit. An alternative is to retain the high-level circuit abstraction and incorporate the results of a field analysis in to a circuit abstraction.

A review of modified nodal admittance (MNA) analysis to lay the foundation for the work presented is done here. MNA analysis was developed to handle elements that do not have nodal admittance descriptions. For each element one or more additional equations are added to the nodal admittance equations and these new equations become additional rows and columns in the evolving matrix system of equations. For the electromagnetic elements, the concept of local reference node was developed and can be found in [1]. This provides another way to incorporate alternate equations in the evolving MNA matrix. The local reference node concept also changes the way the port-based parameters are used. In conventional circuit where there is only one reference node (global or ground), and application of KCL to this node introduces just one additional

redundant row and column in the indefinite form of the MNA matrix. For a spatially distributed circuit, KCL is applied to each reference node one at a time and this results in addition of one row and column for each local reference node.

2.3 Circuit Representation

In a multi-port element, the terminals of ports with different local reference nodes can be considered isolated. The formulation of the system equations begins with the nonlinear elements replaced by variable voltage or current sources. For each non-linear element, one terminal is taken as the reference and the element is replaced by a set of sources connected to the reference terminal. The state of the element can be determined considering only the current of the sources, or the voltage across the sources, or the combination of both. In the nodal admittance matrix method, a matrix is developed that relates the unknown node voltages to the external currents.

A general circuit with local reference nodes required with a Spatially Distributed Linear Circuit (SDLC) and non-linearities is shown in Fig. 2.1.

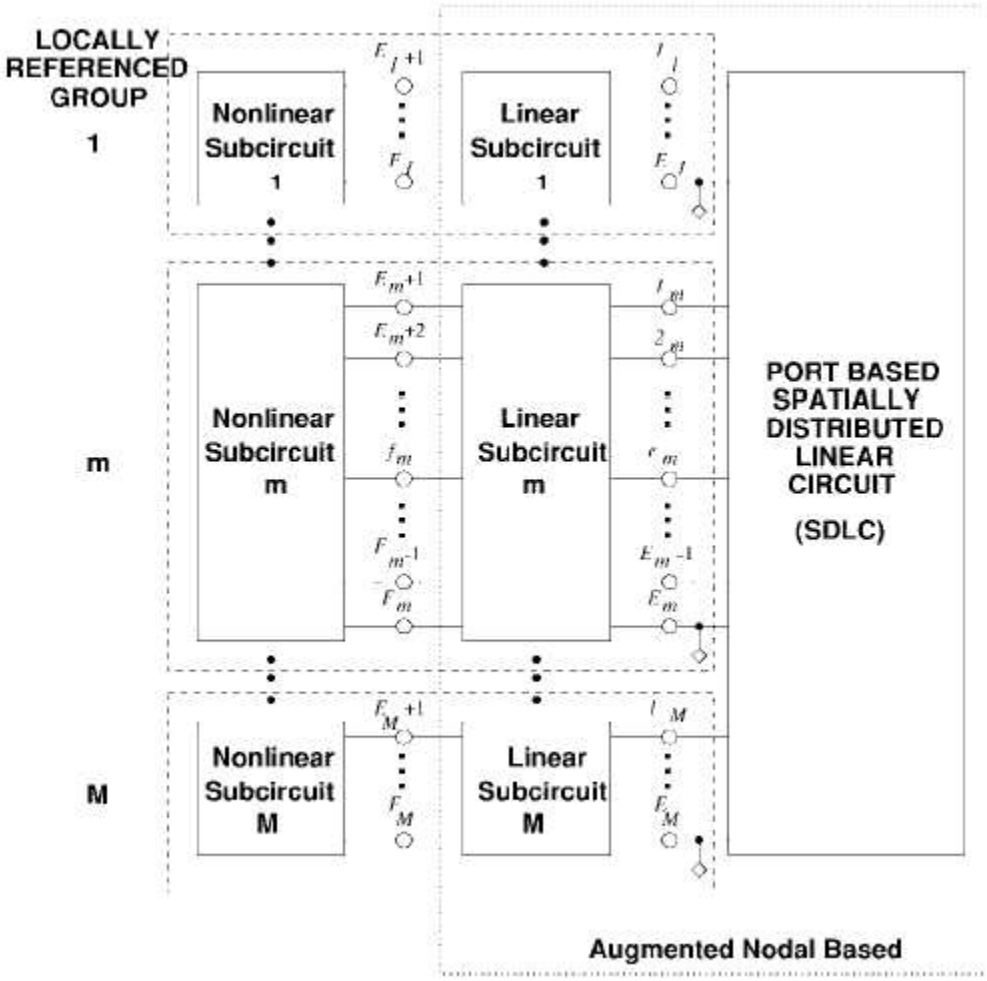

Figure 2.1: Port defined system connected to nodal defined circuit.

This figure depicts the essential circuit analysis issue: integrating the representation of an SDLC with a circuit defined in a conventional nodal manner to obtain an augmented nodal-based description. This creates a redundancy, however. For locally referenced group number m, there are E_m terminals, E_m-1

locally referenced ports, and one local reference node, which is designated E_m.

The port-based system may be expressed as

$$[\ {}_pY][pV] = [{}_pI].$$

Where the port-based admittance matrix is

$$. p\mathbf{Y} = \begin{pmatrix} {}_p\boldsymbol{Y}_{1,1} & \ldots\ldots & {}_p\boldsymbol{Y}_{1,m} & \ldots\ldots\ldots\ldots & {}_p\boldsymbol{Y}_{1,M} \\ \vdots & & \vdots & & \vdots \\ {}_p\boldsymbol{Y}_{m,1} & \ldots\ldots\ldots & {}_p\boldsymbol{Y}_{m,m} & \ldots\ldots\ldots & {}_p\boldsymbol{Y}_{m,M} \\ \vdots & & \vdots & & \vdots \\ {}_p\boldsymbol{Y}_{M,1} & \ldots\ldots\ldots & {}_p\boldsymbol{Y}_{M,m} & \ldots\ldots\ldots & {}_p\boldsymbol{Y}_{M,M} \end{pmatrix}$$

the port-based voltage vector

$$p\mathbf{V} = [p\mathbf{V}_1 \ldots\ldots\ p\mathbf{V}_m \ldots\ldots\ldots\ P\mathbf{V}_M]^T$$

and the port-based current vector

$$p\mathbf{I} = [p\mathbf{I}_1 \ \ldots\ldots\ p\mathbf{I}_m \ldots\ldots\ldots\ p\mathbf{I}_M]^T.$$

The sub matrix $p\mathbf{Y}_{i,j}$ is the mutual port admittance matrix (of dimension E_i-1 $\times$ E_j-1) between groups *i* and *j* of the SDLC.

In order to use nodal analysis, the port-based system must be formulated in a nodal admittance form. If there are N local reference nodes, another redundant N rows and N columns can be added to the port admittance matrix such that

$$[{}_{n}\mathbf{Y}][{}_{n}\mathbf{V}] = [{}_{n}\mathbf{I}].$$

Where the nodal admittance matrix is

$${}_{n}\mathbf{Y} = \begin{pmatrix} {}_{p}\mathbf{Y} & \mathbf{Y}_1 \\ \mathbf{Y}_2 & \mathbf{Y}_3 \end{pmatrix}$$

The linear circuits of each group are assumed to be disconnected from the rest of the circuit: thus, each circuit group can be visualized as a disjointed graph. E.g., a linear circuit at group *j* (*i* != *j*). The only coupling that can exist between different locally referenced groups is accounted for in the description of an SDLC. As such, for the linear subcircuits (as in Fig. 3.4), all the entries in the admittance matrix are zero, except those relating the node parameters at the same group (*i* = *j*). We can have a connection between any two distributed circuits with only a distributed element. Not a lumped element. This is to ensure that the two groups of circuit are electrically separated. We can define the interfacing nodes as the nodes between lumped linear circuits and non-linear circuits, a lumped linear circuit embedded at group *m* can be represented as

$$[Y][V] = [I].$$

Transim is a circuit simulator that incorporates the ideas presented in this section.

2.4 TRANSIM

2.4.1 Introduction

Sub-micron design and rapid growth of microwave, millimeter wave and RF systems puts a higher demand for the development of an easily extensible and modifiable computer aided engineering (CAE) environment. Transim is a circuit simulator tool based written with object oriented (OO) design practice in mind. OO design need not always be written with OO-specific programming languages. It's the design, which needs to be OO oriented and can be implemented in a conventional coding language, as C. Electronic systems are inherently more inclined towards the OO fashion. E.g., circuit elements are already viewed as discrete objects and at the same time as an integral part of a circuit. The OO view helps in unifying the electronic concept with the way humans perceive things.

Languages for OO implementation like C++ have been considered slow and at a higher abstraction than conventional languages and thus thought to be unsuitable for scientific applications. However, advances in compilers and

programming techniques have made C++ attractive, and in some benchmarks C++ outperforms Fortran. Several OO numerical libraries have been developed with standard template library (STL) being a classical example. The STL is a C++ library of container classes, algorithms, and iterators; it provides many of the basic algorithms and data structures of computer science. Almost every component in the STL is a template, thus making it very parameterized. The current ISO/ANSI C++ standard has not been fully implemented and C++ compilers support a variable subset of the standard. The goal in design was to faster development, to use off the shelf advanced numerical techniques, and to allow easy expansion and testing of new models and numerical methods. The circuit simulator implementing these ideas is TRANSIM. Transim uses recent OO techniques. The design intent was to combine the advantages of previous OO circuit simulators with these new developments as well as expanding capability. Transim uses C++ libraries and some written in C or Fortran.

2.4.2 Support Libraries

Solution of Sparse Linear Systems

Sparse 1.3 2 is a flexible package of subroutines written in C used to numerically solve large sparse systems of linear equations. The package is able to handle arbitrary real and complex square matrix equations. Besides being able to solve

linear systems, it is also able to quickly solve transposed systems, find determinants, and estimate errors due to ill conditioning in the system of equations and instability in the computations. Sparse also provides a test program that is able to read matrix equation from a file, solve it, and print useful information (such as condition number of the matrix) about the equation and its solution. Sparse was originally written for use in circuit simulators and is well adapted to handling nodal- and modified-nodal admittance matrices. SuperLU 3 is used in the wavelet and time marching transient analyses. It contains a set of subroutines to numerically solve a sparse linear system Ax = b. It uses Gaussian elimination with partial pivoting (GEPP). The columns of A may be pre-ordered before factorization; the pre-ordering for sparsity is completely separate from the factorization. SuperLU is implemented in ANSI C. It provides support for both real and complex matrices, in both single and double precision.

Vectors and Matrices

Most of the vector and matrix handling in Transim uses MV++4. This is a small set of vector and simple matrix classes for numerical computing written in C++. The various MV++ classes form the building blocks of larger user-level libraries. The MV++ package includes interfaces to the computational kernels of the Basic Linear Algebra Subprograms package (BLAS), which includes scalar updates, vector sums, and dot products. The idea is to utilize vendor-supplied, or optimized BLAS routines that are fine-tuned for particular platforms. The Matrix template Library (MTL) 5 is a high-performance generic component library that

provides comprehensive linear algebra functionality for a wide variety of matrix formats. It is used in the wavelet and time marching transient analyses. As with the STL, MTL uses a five-fold approach, consisting of generic functions, containers, iterators, adaptors, and function objects, all developed specifically for high performance numerical linear algebra. Within this framework, MTL provides generic algorithms corresponding to the mathematical operations that define linear algebra. Similarly, the containers, adaptors, and iterators are used to represent and to manipulate matrices and vectors.

Solution of Non-Linear systems

Nonlinear systems of equations in Transim are solved using the NNES6 library. This package is written in Fortran and provides Newton and quasi- Newton methods with many options including the use of analytic Jacobian or forward, backwards or central differences to approximate it, different quasi-Newton Jacobian updates, or two globally convergent methods, etc. This library is used through an interface class (NLSInterface), so it is possible to install a different routine to solve nonlinear systems if desired by just replacing the interface (four different nonlinear solvers have already been used). The Fortran routine NLEQ1 (Numerical solution of nonlinear (NL) equations (EQ) 7) can also be used as a compile option.

Fourier Transform

Fourier transformation is implemented in Transim using the FFTW8 library. FFTW is a C subroutine library for computing the Discrete Fourier Transform

(DFT) in one or more dimensions, of both real and complex data, and of arbitrary input size. Benchmarks, performed on a variety of platforms show that FFTW's performance is typically superior to that of other publicly available FFT software. Moreover, FFTW's performance is portable: the program performs well on most computer architectures without modification.

Automatic Differentiation

Most nonlinear computations require the evaluation of first and higher derivatives of vector functions with m components in n real or complex variables. Often these functions are defined by sequential evaluation procedures involving many intermediate variables. By eliminating the intermediate variables symbolically, it is theoretically always possible to express the m dependent variables directly in terms of the n independent variables. Typically, however, the attempt results in unwieldy algebraic formulae, if it can be completed at all. Symbolic differentiation of the resulting formulae will usually exacerbate this problem of expression swell and often entails the repeated evaluation of common expressions. An obvious way to avoid such redundant calculations is to apply an optimizing compiler to the source code that can be generated from the symbolic representation of the derivatives in question. AD has two basic modes, forward mode and reverse mode. The difference between these two is the way the chain rule is used to propagate the derivatives. A versatile implementation of the AD technique is ADOL-C 9, a software package written in C and C++. The numerical values of

derivative vectors (required to fill a Jacobian for solving non-linear elements using Newton's method) are obtained free of truncation errors at a small multiple of the run time required to evaluate the original function with little additional memory required. It is important to note that AD is not numerical differentiation and the same accuracy achieved by evaluating analytically developed derivatives is obtained. The eval () method of the nonlinear element class is executed at initialization time and so the operations to calculate the currents and voltages of each element are recorded by ADOL-C in a tape, which is actually an internal buffer. After that, each time that the values or the derivatives of the nonlinear elements are required, an ADOL-C function is called and the values are calculated using the tapes. This implementation is efficient because the taping process is done only once (this almost doubles the speed of the calculation compared to the case where the functions are taped each time they are needed). When the Jacobian is needed, the corresponding Adol-C function is called using the same tape. In the case of Harmonic Balance simulations, the program has been tested with large circuits with many tones, and the function or Jacobian evaluation times are always very small compared with the time required to solve the matrix equation (typically some form of Newton's method) that uses the Jacobian. The conclusion is that there is little detriment to the performance of the program introduced by using automatic differentiation. However the advantage in terms of rapid model development is significant. The majority of the development time in implementing models in simulator is in the manual development of the

derivative equations. Unfortunately the determination of derivatives using numerical differences is not sufficiently accurate for any but the simplest circuits and in any event, is computationally intensive. With ADOL-C full 'analytic' accuracy is obtained and the implementation of nonlinear device models is dramatically simplified. From experience the average time to develop and implement a transistor model is an order of magnitude less than deriving and coding the derivatives manually. Note that time differentiation, time delay and transformations are left outside the automatic differentiation block. The calculation speed achieved is approximately ten times faster than the speed achieved by including time differentiation, time delay and transformations inside the block.

2.5 Electric Editor VLSI System Design Tool

2.5.1 Introduction

The Electric VLSI Design System tool can handle many different types of circuit design (MOS, Bipolar, schematics, printed circuitry, and hardware description languages, etc.) It also handles non-Manhattan geometry and curves. Layout is done by placing and wiring electrical components. Electric provides design-rule checking, simulation, and network comparison. The user interface is quite sophisticated and runs on all popular workstations (Windows, Macintosh, and UNIX).

The global enforcement of connectivity, which provides top-down design capability and ease of post-design modifications.

The components of Electric are summarized in Table 2.1

Tools	**Technologies**	**Interfaces**
Design Rule Checking (3 options)	nMOS	CIF I/O
Electrical Rules Checking	CMOS (6 variations)	GDS I/O
Simulation	Bipolar	EDIF I/O
Simulation Interface (12 options)	BiCMOS	DXF I/O
Layout Generation (3 options)	Schematics	SDF Input
Compaction	Printed Circuits	SUE Input
Compensation	Digital Filters	VHDL I/O
Routing (4 options)	FPGA	Verilog Output
VHDL Compilation		CDL Output
Silicon Compilation		EAGLE Output
Network Consistency Checking(LVS)		PADS Output
Logical Effort Analysis		ECAD Output
Project Management		HPGL Output
		QuickDraw Output
		PostScript Output

Table 2.1: Electric Capabilities.

Most CAD systems use two methods to do circuit design: *connectivity* and *geometry*. The connectivity approach is used by every Schematic design system: place components and draw connecting wires. The components remain connected, even when they move. The geometry approach is used by most Integrated Circuit layout systems: rectangles of "paint" are laid down on different layers to form the masks for chip fabrication.

Electric uses connectivity for all design, even Integrated Circuit layout. When the user places components (MOS transistors, contacts, etc.) and draw wires (metal-2, polysilicon, etc.) to connect them, the screen shows the true geometry, but it knows the connectivity too.

The advantages of connectivity-based IC layout are:

Common user interface. One CAD system, with a single user interface, can be used to do both IC layout and schematics. Electric tightly integrates the process of drawing separate schematics and its LVS tool compares them.

No node extraction. Node extraction is not a separate, error-prone step. Instead, the connectivity is part of the layout description and is instantly available. This speeds up all network-oriented operations, including simulation, LVS, and electrical rules checkers.

No geometry errors. Complex components are no longer composed of unrelated pieces of geometry that can be moved independently. In paint systems, you can accidentally move the gate geometry away from a transistor, thus

deleting the transistor. In Electric, the transistor is a single component, and cannot be accidentally destroyed.

More powerful editing. Browsing the circuit is more powerful because the editor can show the entire network whenever part of it is selected. Also, Electric combines the connectivity with a layout constraint system to give the editor powerful manipulation tools. These tools keep the design well connected, even as the circuit is modified on different levels of hierarchy.

2.5.2 Work on Electric

One of the shortcomings of Electric is the lack of spatial parametric description. The two ends of a metal layer could not be identified with along the geometry and this was essential for the field simulator tool. Also, the metal layers did not store the material properties like $\varepsilon_{r(\text{dielectric constant})}$, δ (loss tangent), □ (sheet resistivity), σ (conductivity), which is essential in characterizing microwave circuits.

Chapter 3

Tool Flow

3.1 Introduction

The design and simulation of spatial power combining systems requires a general formulation of the problem and a solution strategy. This chapter presents the plan for design and simulation of spatial power combining systems with a discussion on the specific programs, their interfaces, and implementation. The specific goal is to simulate a grid amplifier, however, the problem will be approached for a generic case. The integrability and interoperability of Electric Editor with Transim and an EM simulator is also discussed.

3.2 General Formulation of the Tool Flow

A typical spatial power combining system discussed in previous chapters presents some very challenging problems: components are spatially distributed, power is radiated through the system, and nonlinear devices are present in the system. The first two problems dictate that electromagnetic models be used, with the nonlinear circuits simulated in a circuit simulator. Then, the solution is to divide the two problems up as shown in Fig 3.1.

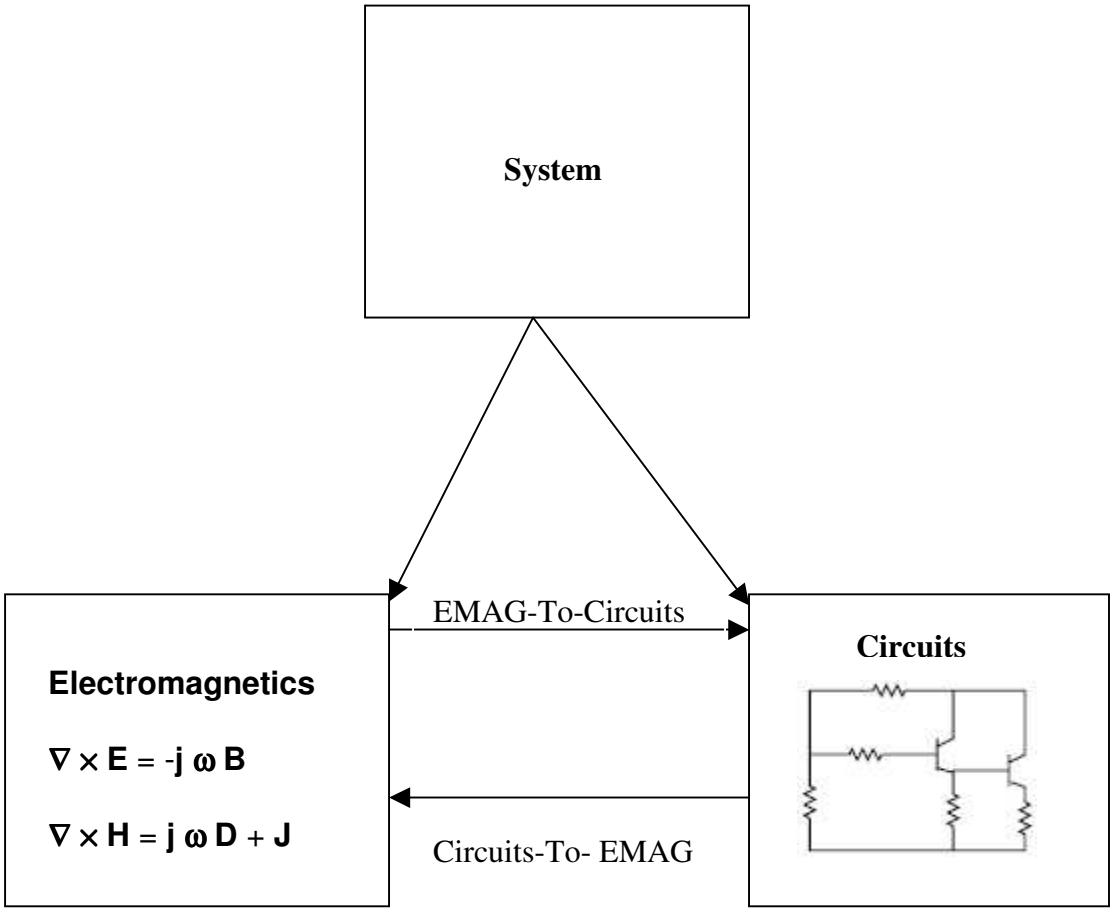

Figure 3.1: Splitting of system into fields and circuits.

The spatially distributed elements are handled through electromagnetic simulation, and the nonlinear devices simulated in Transim circuit simulator. A front-end for the entire system is implemented here. Specifically, the associated tasks of this front end is to let the user:

- design the schematic,
- draw the layout and perform design rule checks,
- perform layout-versus-schematic check,

- extract netlist for circuit simulation,
- extract geometric information as CIF, and
- pass this to the EM and the circuit simulator.

Before passing the netlist to the circuit simulator, group of elements spatially separated from another group needs to be identified. The associated local reference node for each group also has to be identified.

The block represented here by the 'System' encapsulates the entire function described above. Most libraries like the CMOS, BiCMOS, Bipolar etc for schematic and layout are supported. The electro-magnetic simulator is encapsulated in the 'Electromagnetics' block and the circuit simulator in the 'Circuits' block. Models for the circuit simulator Transim has also been part of this work.

3.3 Distributed Nature of Grid Amplifier

The array or grid amplifier in a Quasi-Optical system is an example of a spatially distributed circuit and thus needs the modeling environment suitable for distributed system. A simple 2×2 spatial gird amplifier where a signal transmitted by a horn antenna illuminates a 2×2 grid of amplifier unit cells is shown in Fig. 3.2

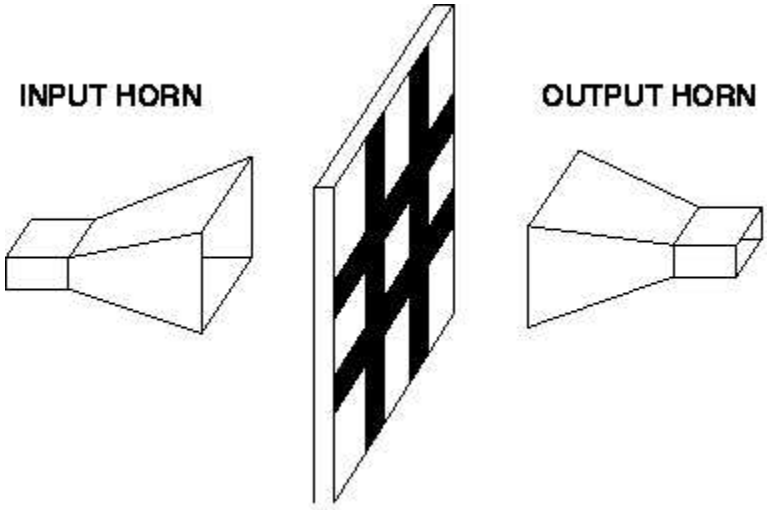

Figure 3.2: A 2×2 spatial gird amplifier.

The planar array of amplifiers is shown in greater detail in Fig. 3.3 where each unit cell is localized at a grid intersection and comprises two MMICs in a push-pull configuration. Each unit cell is separated from its neighbors and each radiates amplified power at the orthogonal polarization to the input signal. The contributions from each cell are combined spatially, and the output horn antenna collects this power.

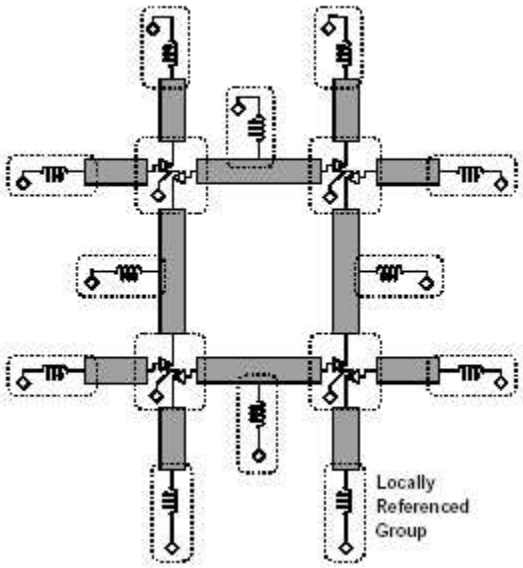

Figure 3.3: Local groups in a grid amplifier.

In Fig. 3.3, each group of devices is separated by a distributed element (transmission line) and, thus a global reference node cannot be defined for this electrically large structure. Instead local reference nodes are defined, one per unit cell and one for each of the bias circuit connections. The metal layer in Fig. 3.3 can be a transmission line is distributed in nature. The devices in each local referenced group would be lumped in nature, e.g., transistors, resistors etc.

The schematic drawn using the Electric Editor produces a netlist with the required information of local reference nodes and the associated groups. Transim then identifies each locally referenced group and its reference node.

3.4 Tool flow

The integrated design environment consists of three different simulators, each for a specific function and is summarized in the Fig. 3.4. The Electric Editor is used to design the schematic and the layout. A netlist for Transim and the geometric information for the EM simulator is generated after this. The netlist from this goes to the Transim circuit simulator and the geometry information in CIF file is given into the Field Solver.

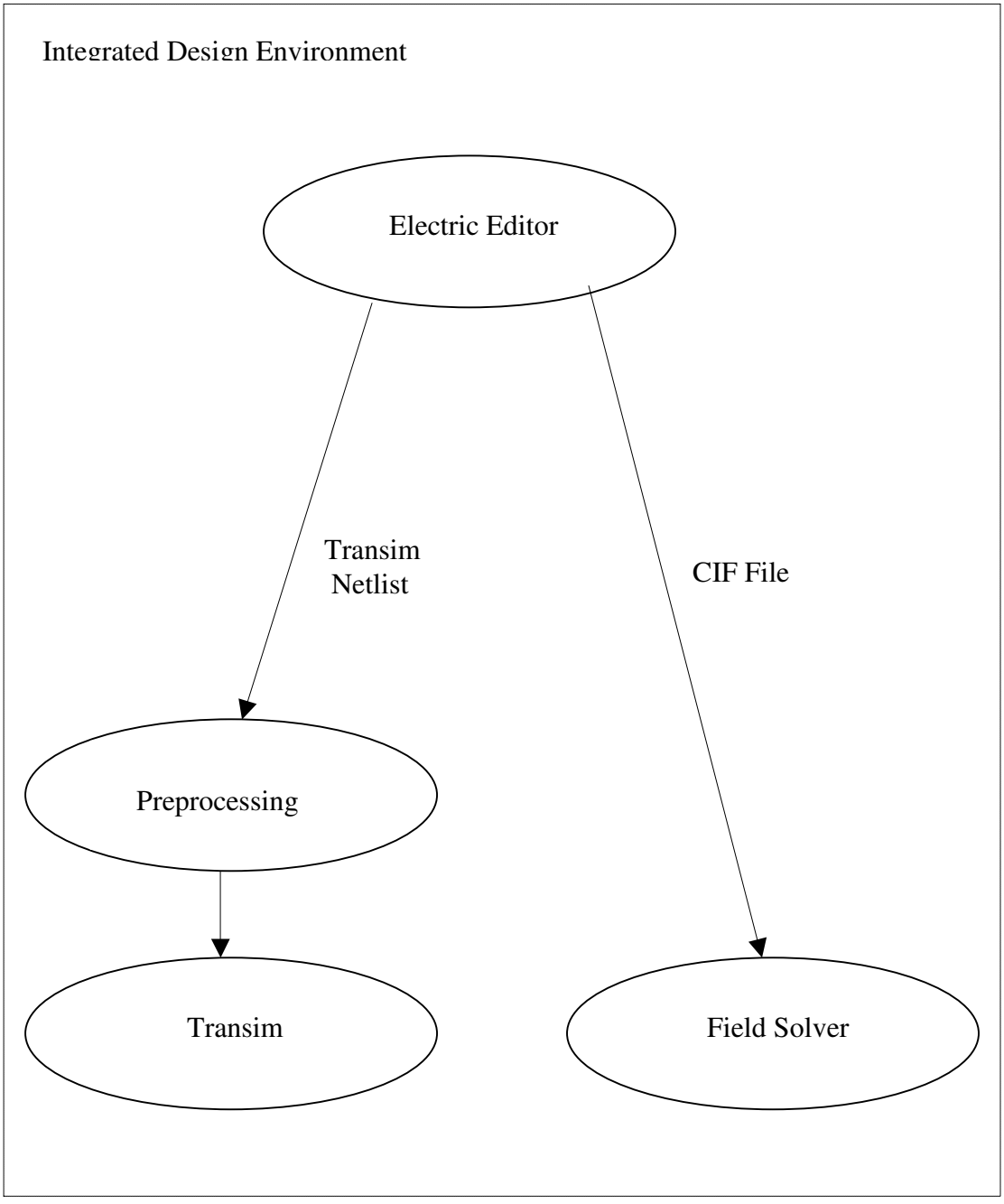

Figure 3.4: Integrated Environment for the QO System Design.

3.5 Netlist and CIF Generation

The Electric Editor has a common user interface, which allows the designer to have both schematic and the layout in a single window. This is important in systems that involve a lot of passives in addition to the active devices. With a common interface, a designer can model the active devices in its schematic form and at the same time draw and analyze the transmission lines and other passives like spiral inductor laying it out [Patwardhan1]. The tool capability is enhanced to serve as a generic design tool in addition being a design tool for a power combining system. This section elaborates on it further.

3.5.1 Spatial Parametric Description in Electric Editor

For a useful interface between the circuit simulator and the field solver, the ports of a metal layer or structure has to be stored and mapped later. The layout-schematic interface of Electric Editor is meant as an Integrated Circuit Design tool, more than a microwave system design tool [Rubin1]. The two ends of, say a metal layer is not defined and stored internally. The port naming and storing capability needs to be added to Electric to make it suitable for the system in hand.

An example is illustrated in Fig. 3.5. Here, the tool allows the connection between the schematic of the transistor with the metal layer. But, for metal layers 1, 2, 3 or 4, their endpoints cannot be named or stored internally. This makes it difficult to get the ports of a metal layer. The netlist for the circuit simulator Transim and

the geometric information for the EM simulator should have a node mapping with each other, so that the results of their simulation can be correlated from [Patwardhan1].

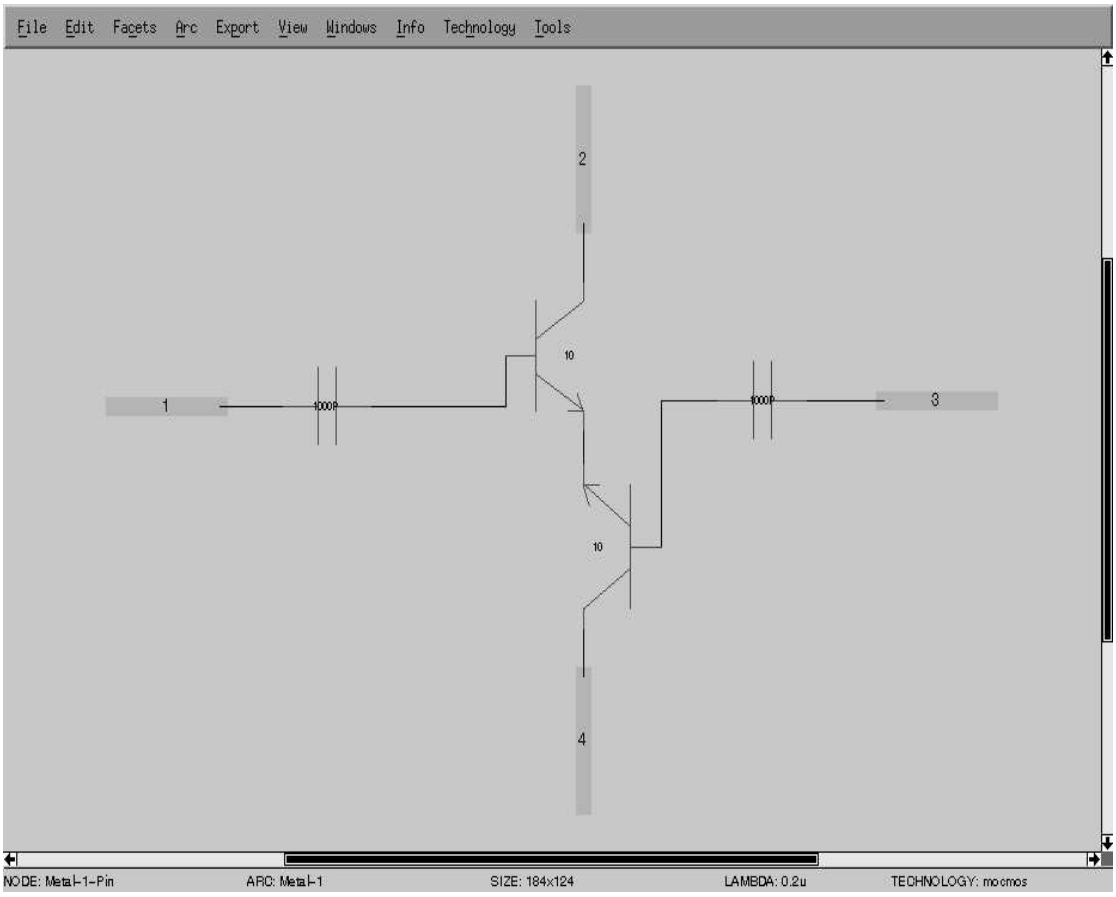

Figure 3.5: Electric screen shot with mixed schematic and layout.

3.5.2 Implementation of Nodal Description

This section describes the implementation of port naming at the end points of a metal layer. The Electric Editor tool is written in C language and has over 312

thousand lines of code. The code is organized into 15 directories, each directory having many files. The challenge of writing new code into Electric was to preserve consistency with the future releases. This is important because of Electric Editor is a GNU public tool and public domain tools grow over a period of time. Any customization should be such that it stays effective over a period of time, with minimal effort. An algorithm was implemented in Electric with the following flow chart shown in Fig. 3.6. For this a new data structure was defined in Electric Editor.

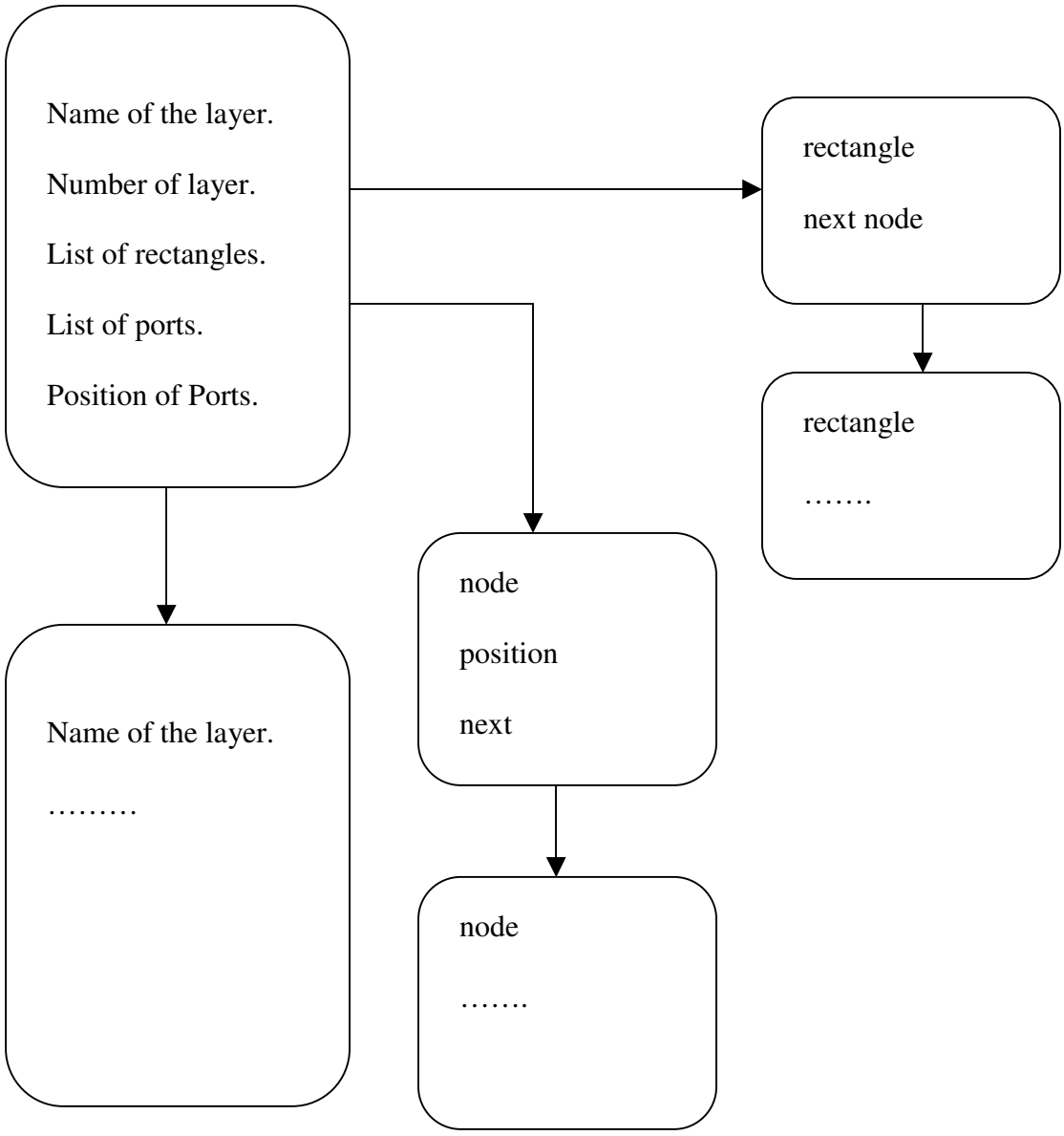

Figure 3.6: Implementation of port description in Electric Editor.

A function which returns a pointer to a list of layers (with the layer and port data) was written. This was mainly to implement the port naming capability to Electric Editor.

3.5.3 Technology Layer

Since Electric Editor was designed primarily to as an IC design tool, it does not have enough support for the micro-strip and stripline design. Also, the multiple layer capability needs to be added to support the design process for a spatial power combining system. This involves a metal layer enclosed between dielectric materials. In that case, the loss tangent, dielectric constant, conductivity and thickness of the dielectric material needs to be stored in addition to the metal properties [Summers1].

The first step to add this capability has been accomplished in this work as shown below in Fig 3.7.

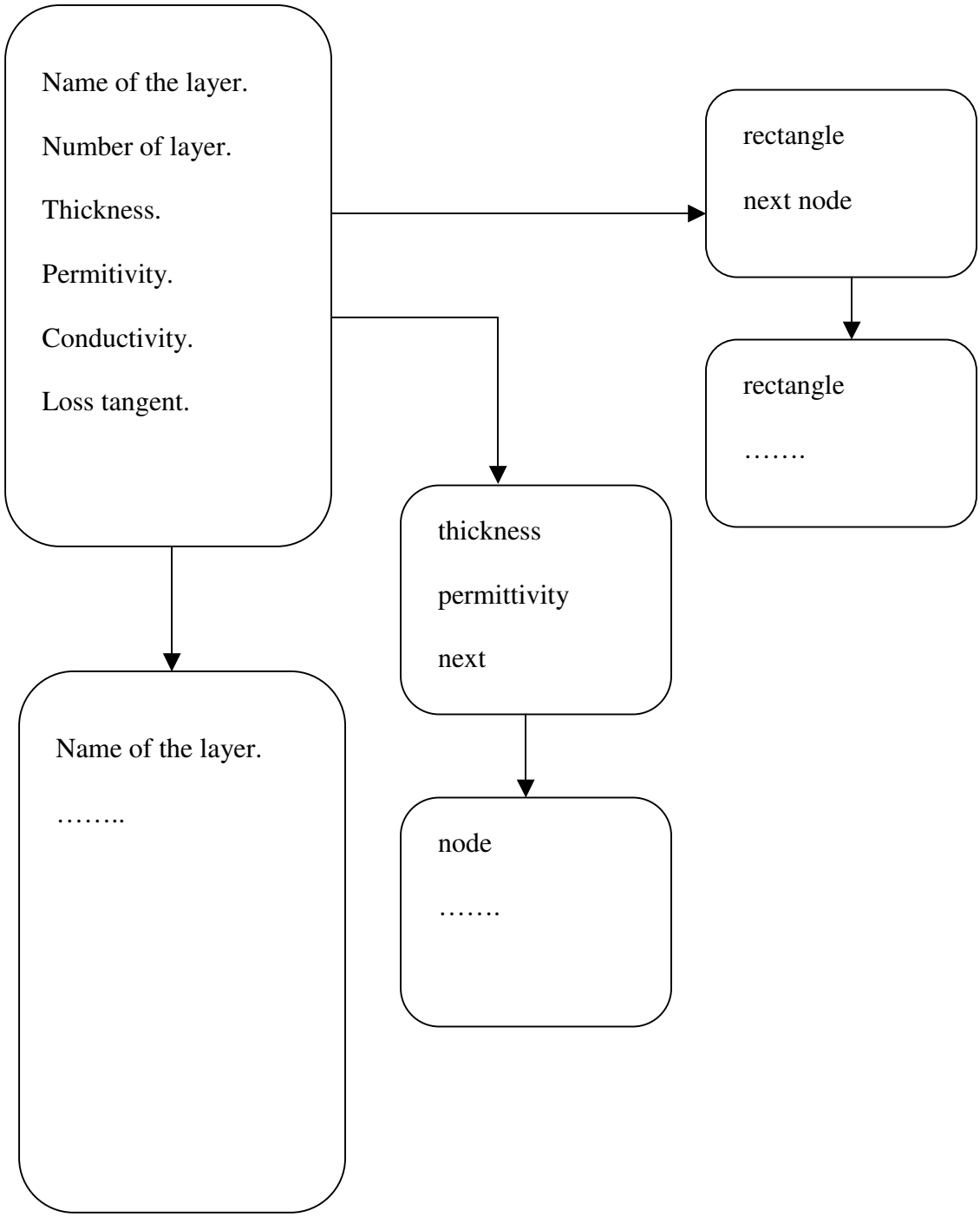

Figure 3.7: Implementation for material properties in Electric Editor.

3.5.4 CIF Extensions

For convenient description of electromagnetic grids, few extensions have been defined to the CIF. One is the metal layer and nodal information.

Layer

B 60 140 -20 390;

0 10 20 CMF 1 n2;

This command specifies that a BOX of dimensions 60 and 140 units is at the coordinate –20 and 390. Next line specifies one of the end points of the metal layer and is nodal information. It says that node n2 is at coordinate 10 and 20 and is a CMOS metal layer 1.

With these additions, Electric Editor can generate the geometry information of any layout design.

Thickness

For complete description of a metal layer, the CIF file parsed into the EM simulator should have extensions to encapsulate the thickness of the metal layers [Patwardhan1], as described below.

94 Thickness=0.33 35 29;

This command would specify the thickness of a layer, in millimeters. The parameters are the thickness in millimeters, and the coordinates of the point where the label is to be displayed.

Relative Permittvity

94 Relative_Permittivity=2.1 26,34;

This command has a format similar to the thickness format, and is used to specify the relative permittivity of a layer.

Conductivity

94 Conductivity=0.5 21,18;

The format of this command is similar to the relative permittivity format, and it is used to specify the conductivity of a layer.

User-defined Commands

94 <user-text><:|=><value> x, y;

The user-defined commands can be any alphanumeric identifier followed by a colon or an equal sign with the value being a floating-point number or a string.

3.5.4 Interface with the Simulator

After the CIF for the design has been generated automatically, the necessary geometric information in the CIF is passed into the EM simulator. When the CIF file is generated, every port is assigned a unique number. A separate text file is written which keeps a mapping of the port number generated in the CIF file and that meant for Transim.

3.5.5 Transim Netlist

The circuit for a QO system needs circuit simulation in addition to the electromagnetic simulation. Electric Editor produces the netlist for Transim, with the local groups. The local reference node corresponding to each group is identified using a pre-processing stage before being simulated in Transim. The local reference nodes are identified in the Transim using a specialized algorithm. Implementation of the technique can be described using a conventional procedural approach or an object oriented approach. The object oriented view maps onto the circuit analysis problem more effectively. A circuit is a collection of objects (like inductors, transistors, etc.) that are related to each other (by nodal connection). Also, spatially distributed circuits can be connected together in any manner but the interconnections of non-spatially distributed elements cannot span spatially distributed elements. In Transim, each element and terminal is a node in a graph structure. The algorithm to detect connectivity violation is a variation of the depth-search algorithm and can be best illustrated using the collaboration diagram. The algorithm has also been implemented using conventional procedural language [Rivest1]. More details on the implementation are in [Christoffersen1]. The pre-processing stage is implemented in Transim internally and in the netlist to the Transim, the local reference nodes are identified with the command “.ref <terminal>”. And, by default the nodes “0” and “gnd” are assumed to be reference nodes.

3.6 Transim Sources

As part of this thesis, the Controlled and Independent sources in Transim were also implemented. The Independent Sources implemented were the DC, Sinusoidal with Damping, Pulse, Exponential, Single-Frequency FM and Amplitude Modulated. The controlled sources are Voltage-Controlled-Voltage-Source, Voltage-Controlled-Current-Source, Current-Controlled-Voltage-Source and the Current-Controlled-Current-Source. Implementation of these models is further elaborated in the Appendix B.

Chapter 4

Results

4.1 Introduction

The earlier chapters described the approach towards an integrated design environment for design of spatial power combining systems. In this chapter various results are presented. The achievement is that the same results are obtained with the current design environment compared to previous results in [Patwardhan1]. Transim specific results for sources are also presented.

4.2 Results for Grids

In this section various size grids are analyzed and the intermediate CIF data is compared with that in [Patwardhan1]. The principle achievement is that the CIF output generated automatically by this work is exactly same to that hand-generated by user in [Patwardhan1]. This indicates that the new environment with the CIF output is working.

4.2.1 Unit Cell

The CIF output generated is for a unit cell shown in Fig. 4.1. Its magnitude and phase are plotted versus frequency in Fig. 4.2 and Fig. 4.3 respectively. The solid line shows results obtained by simulation and the dashed line is for the measurements taken. In Fig. 4.1, the geometry for a unit cell of dimension 93.8 mm $\times$ 93.8 mm with a gap spacing of 9.8 mm and a line width of 6.35 mm is shown.

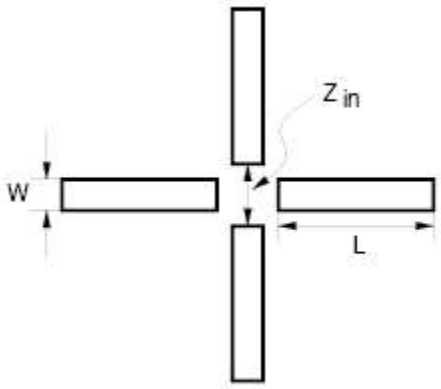

Figure 4.1: Unit cell configuration [Patwardhan1].

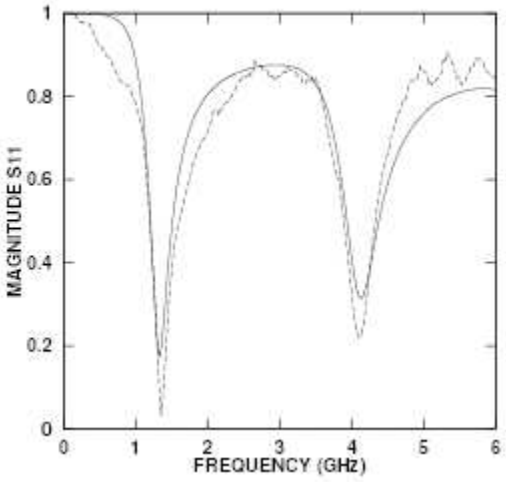

Figure 4.2: Driving point reflection magnitude of the unit cell [Patwardhan2].

4.2.2 5×5 Grid

The data (CIF) generated for this grid was compared and found to match with the data in [Patwardhan1]. In [Patwardhan1], the design and simulation were performed for the 5×5 grid array placed in the lens system at $z = 0$ on a dielectric substrate with $\varepsilon_r = 2.56$ and thickness 9.5 mm.

It may be noted that a significant improvement over [Patwardhan1] is the resolution of automatic port naming and mapping.

4.3 Transim Results

The results for the sinusoidal source with damping implemented in Transim is also provided. The Transim results has been compared with the Hspice simulation results and found to be agreeable.

The test netlist uses a sinusoidal source with a vdc = 1V, vac = 5V, frequency = 3 MHz and damping factor value of 130000. The waveforms for the Transim simulation up to 5 microseconds are shown in Fig. 4.3.

The Hspice simulation waveform is given in Fig. 4.4 for comparison.

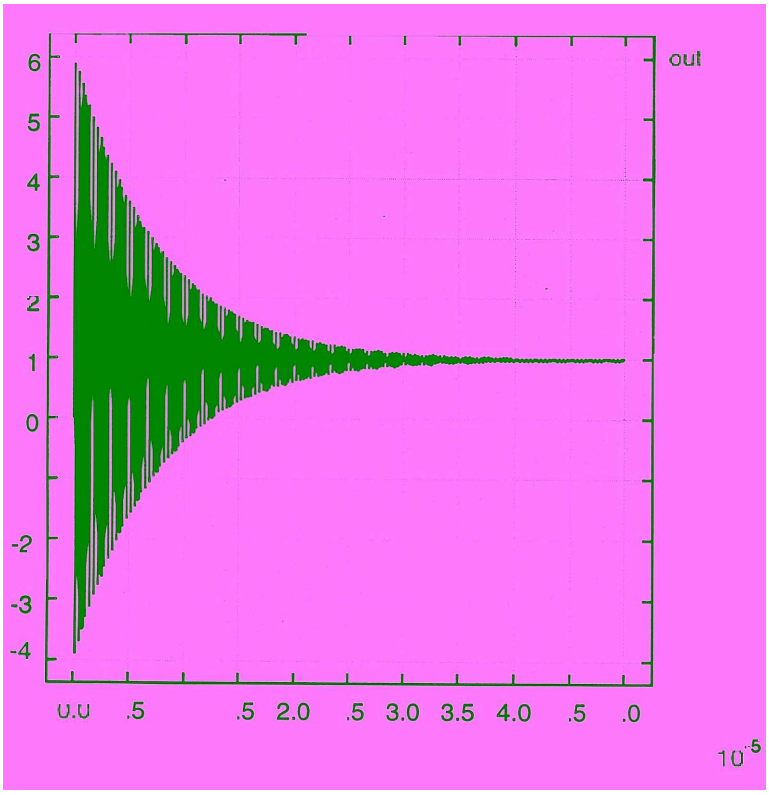

Fig. 4.3: Transim Waveform.

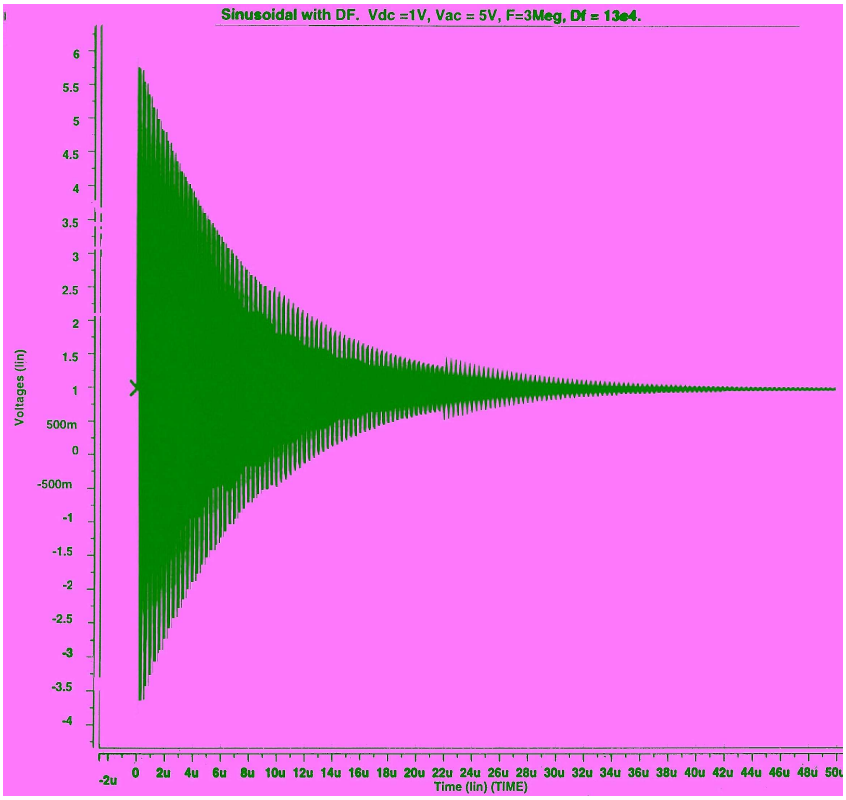

Fig. 4.4: Hspice Waveform.

5.4 Results for Port Naming.

The modified Electric Editor can now support node naming at the ends of a metal layer as shown in Fig. 4.5.

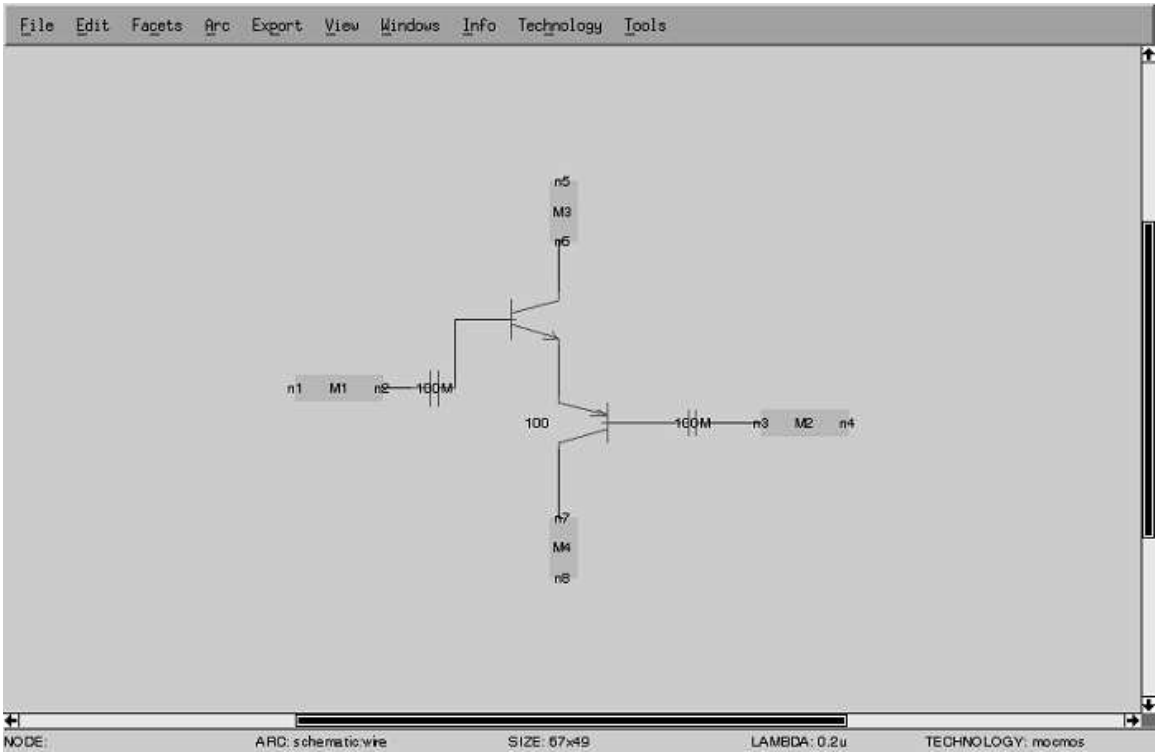

Fig. 4.5: Screen shot for Electric Editor after modification.

Chapter 5

Conclusions and Future Research

5.1 Conclusions

An integrated design environment consisting of a GUI, electromagnetic and circuit simulation program for design and simulation of spatial power combining structures is aimed at in this work. Surface modes and full nonlinear effects can be studied. Validation has been done by correlating data from previous results [Patwardhan1]. The programs for the nonlinear analysis and the interfaces have been defined and documented.

The circuit simulator Transim is a state variable based harmonic balance simulator. Interface program to Transim for design and validation of arbitrary structures has been implemented. The Controlled and Independent sources models were also added to Transim as part of this work.

The simulation environment allows grid amplifier structures to be examined from a wholistic point of view. Overlap of structures, design rule check (DRC), schematic-based entry, multi-layer design can now be done. The simulation environment can be extended to allow what-if studies to be conducted. Common interface for schematic and layout would improve design productivity and

efficiency. Design ideas can now be implemented on the computer first, increasing the overall chance of success.

5.2 Future Research

There are many areas that could be pursued further with regard to this integrated simulation environment. The programs in the environment could be pieced together in a more coherent fashion, once the EM simulator is built. Multi-layer functionality could be extended to make the design environment to support structures like embedded micro-strip, offset stripline. 2D structures other than rectangle shaped like arcs would be a good addition. Schematic based output validation would make the design environment better in productive.

The circuit simulator can be made complete by incorporating parasitic models for all active devices.

Further work is in progress to incorporate capabilities like merge function in the GUI.

References

[Christofferensen] C. E. Christoffersen, M. B. Steer, “Implementation of the Local Reference Node Concept for Spatially Distributed Circuits,” *Int. J. on RF and Microwave Computer Aided Engineering,* Vol. 9, No. 5, Sept. 1999, pp. 376–384. (MAFET)

[Cormen] T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, McGraw-Hill Book Company, 1900.

[Fjeldy] Fjeldy, Ytterdal and Shur, *Introduction to device modeling and circuit simulation*, A Wiley-Interscience Publication, 1998.

[Khalil] A. I. Khalil and M. B. Steer, ”Circuit theory for spatially distributed microwave circuits,” *IEEE Trans. On Microwave Theory and Techniques,* vol. 46, pp. 1500-1503, Oct. 1998.

[Kim] M. Kim, *Grid Amplifiers*, Ph.D. Dissertation, California Institute of Technology, 1994.

[Kriplani] N. M. Kriplani,

[Mink] J. W. Mink, “Quasi-optical power combining of solid-state millimeter-wave sources,” *IEEE Trans. Microwave Theory Tech*., vol. MTT-34, pp. 273-279, Feb. 1986.

[Moret] B. M. Moret and H. D. Shapiro, “*Algorithms from P to NP*.”

[Patwardhan1] J. Patwardhan, *Modular computer aided field modeling of spatial power combining systems*, M.S. Thesis, NC State University, 1997.

[Patwardhan2] F. Nusseibeh, T. W. Nuteson, J. Patwardhan, M. A. Summers, C. E. Christoffersen, J. Kreskovsky and M. B. Steer "Computer aided engineering

environment for spatial power combining systems," *Proc. 1997 IEEE MTT-S International Microwave Symp.*, Part 2, June 1997, pp. 1073-1076. (MAFET)

[Popovic] Z. B. Popovic, M. Kim, and D. B. Rutledge, "Grid Oscillators," *Int. Journal Infrared and Millimeter Waves,* vol. 9, pp. 1003-1010, Nov. 1998.

[Rubin] S. M. Rubin, "*Computer Aids for VLSI Design*."

[Steer1] M. B. Steer, C. E. Christoffersen, S. Velu and N. Kriplani, "Global modelling of RF and microwave circuits", *Mediterranean Microwave Conf.*, June 2002**.**

[Steer2] M. B. Steer, "Transient and Steady-State Analysis of Non-linear RF and Microwave Circuits," ECE 718 class notes, NCSU, Spring 2001.

[Steer3] M. B. Steer, M. Ozkar and C. E. Christoffersen, "Circuit level modeling of spatially distributed mm and sub mm-Wave Systems," *Proc. 1998 IEEE Sixth International Conference on Terahertz Electronics,* September 1998, pp. 21-24.

[Tannenbaum] A. M. Tannenbaum, Y. Langsam and M. J. Augenstein, "*Data Structures using C*."

Appendix A

Caltech Intermediate Format Information

A.1.1 Introduction

The Caltech Intermediate Format (CIF) is a means of describing graphic items (mask features) of interest for VLSI circuit and systems designers. We will analyze here only the features used for the description of electromagnetic grids.

A.1.2 Syntax

A CIF file is composed of a sequence of characters in a limited character set. The file contains a list of commands, followed by an end marker. The commands are separated by semicolons. The commands of interest are shown in Table A.1.

CIF Commands	
Form	Description
B *integer integer point point*	Box with length, width and coordinates
L *name*	Type of layer specified by name
DS *integer integer integer*	Start symbol definition
DF	Finish symbol definition
C *integer transformation*	Call symbol
E	End marker
digit text	User extension command
(*text*)	Comments
;	Command separator

Table A.1: CIF Commands

A.1.3 Semantics

Measurements

The intermediate form uses a right-handed coordinate system with x increasing to the right and y increasing upward. The units of distance measurement are few hundreds of a micron. There is no limit in the size of the number.

Boxes

Box Length 25 Width 60 Center 80,40;

B25 60 80 40;

These are two equivalent Box commands. The tokens that define the box are the capital B, the numbers and the semicolon at the end. The characters between tokens are only cosmetic with certain restrictions on these characters. e.g., they cannot be a semicolon.

Layer Specification

Layer ND;

L ND;

The layer is specified as a “mode" that applies to all subsequent boxes, until the layer is set again.

Symbols

The symbols, together with the “Call" command reduce the bulk of the intermediate form, and provide a means for a hierarchical representation.

Definition Start #57 A/B= 100/1; ...; Definition Finish;

DS57 100 1; ...; DF;

A symbol is defined by preceding the symbol geometry with the DS command and following it with the DF command. The first argument of the DS command identifies symbol number (unrelated to the number of symbols definitions in the file).

The mechanism for symbol definition includes a convenient way to scale distance

measurements. The second and third arguments to the DS command are called a and b, respectively. Each distance measurement (position or size) in the commands in the symbol is scaled to (a∗distance=b).

Definitions cannot nest. That is, after a DS command is specified, the terminating DF must come before the next DS. The definition may, however, contain calls to other symbols, which may in turn call other symbols.

There is only one restriction on the placement of symbol definitions in a file: a symbol must be defined before its instantiation.

Call Symbol #57 Mirrored in X Rotated to -1,1, then Translated

to 10,20; C57 MX R-1 1 T 10 20;

The C command is used to call a symbol and to specify a transformation that should be applied to all geometry contained in the symbol definition. The call command identifies a symbol with its “symbol index", established when the symbol was defined. The transformation to be applied to the symbol is specified by a list of primitive transformations given in the call command. These are:

T point: Translate the current symbol origin to this point.

M X: Mirror in x, which is multiply x-coordinate by -1.

M Y: Mirror in y, which is multiply y-coordinate by -1.

R point: Rotate symbol's x-axis to this direction.

Each coordinate given in the symbol is transformed according to the first primitive transformation in the call command, then according to the second, etc.

Comments

(History of this design);

The comments facility is provided for improved readability.

End Command

End

E

The final E signals the end of the CIF file.

Cell Name

9 Inverter;

This command is one of the most common extensions to the CIF language. It is used to give a more meaningful name to symbols in addition to the symbol definition number.

Label Command

94 Input 10,20;

This command is another common extension and is used to mark points with a name. The parameters are the name of the label, and the x and y coordinates of the point.

A.1.4 Sample CIF File

This is a sample CIF file modeling a dipole antenna. The units are in millimeters. The name of the layout is `dipole'. It consists of two boxes on a metal1 type layer.

The centers of two cells are located at the coordinates (-100; 0) and (100; 0), and both cell sizes are 200 mm _ 200 mm (square boxes). A circuit port (See chapter 3 for port description) is specified at the coordinate (0; 0).

```
DS 1 1 1;
9 Dipole;
L metal1;
B 200 200 -100,0;B 200 200 100,0;
94 1:1 0,0;
DF;
C 1;
E
```

Appendix B

Sources in Transim

B.1 Introduction

Independent and Controlled sources are an integral part of a circuit simulator and these were implemented in Transim as part of this study. In addition to this, the Multiple-Input-Controlled-Source was also implemented. This chapter describes in length about these models.

B.2 Independent Sources

The Independent Sources implemented were the DC, Sinusoidal with Damping, Pulse, Exponential, Single-Frequency FM and Amplitude Modulated. This section elaborates on these models.

B.2.1 DC and Sinusoidal with damping voltage source

Transim has a damped sinusoidal source that is the product of a dying exponential with a sine wave. The DC source is incorporated in to this as well. Application of this waveform requires the specification of the sine wave frequency, the exponential decay constant, the beginning phase, and the beginning time of the waveform, as summarized below.

Usage:

vsource: <instance name> positive_node negative_node <parameter list>

The parameters for this source are:

vdc: DC voltage (V),

vac: AC voltage peak amplitude (V),

f: AC frequency (Hz),

phase: source phase (degrees),

delay: delay to AC start,

tr: rising time for DC component (s),

df : damping factor (1/s).

Waveform Shape:

0..TD : $V_{offset} + V_{amp}$ .Sin(2. □. □/360)

TD.. Tstop : $V_{offset} + V_{amp}$ .Exp{-(Time-TD). df}.

Sin[2. □.{Freq.(Time-TD) + □/360}]

B.2.2 Pulse voltage source

Transim has a trapezoidal pulse source function, which starts with an initial delay from the beginning of the transient simulation interval to an onset ramp. During the onset ramp, the voltage or current changes linearly from its initial value to the pulse plateau value. After the pulse plateau, the voltage or current moves linearly along a recovery ramp, back to its initial value. The entire pulse repeats with a period per from onset to onset.

Usage:

vpulse: <instance name> positive_node negative_node <parameter list>

The parameters for this source are:

v1: Initial value (V),

v2: Pulsed value (V),

td: Delay time (s),

tr: Rise time (s),

tf: Fall time (s),

pw: Pulse width (s),

per: Period (s).

B.2.3 Exponential voltage source

The usage for this source is:

Usage:

vexp:<instance name> positive_node negative_node <parameter list>

The parameters for this source are:

v1: inital voltage (V),

v2: final voltage (V),

tdr: rise time delay (s),

tdf: fall time delay (s),

tcr: rise time constant (s),

tcf: fall time constant (s).

The waveform shape of an exponential voltage source is given as:

0 to td1 v1

td1 to td2 v1 + (v2 – v1).[1 – exp(-(time – tdr)/tcr)]

td2 to tstop v1 + (v2 – v1).[1 – exp(-(tdf – tdr)/tcr)].exp[-(time – tdf)/tcf]

B.2.4 Single Frequency FM voltage source

The usage for this source is as follows:

Usage:

vsffm:<instance name> positive_node negative_node <parameter list>

The parameters for this source are:

vo: output voltage Offset (V),

va: output voltage amplitude (V),

Fcarrier: carrier frequency (Hz),

mdi: modulation index (no dimension),

fsignal: signal frequency (Hz).

The waveform shape for this source is

Sourcevalue = vo + va.[sin(2.π.fcarrier.time) + mdi.sin(2.π.fsignal.time)

B.2.5 Amplitude Modulated voltage source

The usage for this source is

Usage:

vam:<instance name> positive_node negative_node <parameter list>

The parameters for this source are:

oc: offset constant (dimensionless),

sa: signal amplitude (V),

fcarrier: carrier frequency (Hz),

fmod: modulation frequency(Hz),

td: time delay (s).

The waveform for this source is

$$Sourcevalue = sa.\{oc + \sin[2.\pi.fm.(time - td)]\}.\sin(2.\pi.fcarrier.(time - td)]$$

B.3 Controlled Sources

The controlled sources are Voltage-Controlled-Voltage-Source, Voltage Controlled-Current-Source, Current-Controlled-Voltage-Source and the Current-Controlled-Current-Source. This section elaborates on these models.

B.3.1 Voltage and Current Controlled Elements

Transim has four voltage and current controlled elements that can be used to model both MOS and bipolar transistors. The controlled elements are either linear or nonlinear functions of controlling node voltages or branch currents, depending on whether the polynomial is used or not.

Polynomial Functions

The controlled element statement allows the definition of the controlled output variable (current or voltage) as a polynomial function of one or more voltages or

branch currents. Three polynomial equations cab be used through the POLY(N) parameter.

POLY(1) one-dimensional equation

POLY(2) two-dimensional equation

POLY(3) three-dimensional equation

The POLY(1) polynomial equation specifies a polynomial equation as a function of one controlling variable, POLY(2) as a function of two controlling variables, and POLY(3) as a function of three controlling variables.

Along with each polynomial equation are polynomial coefficient parameters (P0, P1 … Pn) that can be set to explicitly define the equation.

One-Dimensional Function

If the function is one-dimensional (a function of one branch current or node voltage), the function value FV is determined by the following expression:

$FV = P0 + (P1.FA) + (P2.FA^2) + (P3.FA^3) + (PA.FA^4) + (P5.FA^5) + ……..$

FV controlled voltage or current from the controlled source

P0. . .PN coefficients of polynomial equation

FA controlling branch current or nodal voltage

If the polynomial is one-dimensional and exactly one coefficient is specified, Transim assumes it to be P1 (P0 = 0.0) to facilitate the input of linear controlled sources.

One-Dimensional Example

The following controlled source statement is an example of a one-dimensional function:

VCVS1 5 0 POLY(1) 3 2 1 2.5

The above voltage-controlled voltage source is connected to nodes 5 and 0. The single dimension polynomial function parameter, POLY(1), means that VCVS1 is a function of the difference of one nodal voltage pair, in this the voltage difference between nodes 3 and 2, hence FA=V(3,2). The dependent source statement then specifies that P0=1 and P1=2.5. From the one-dimensional polynomial equation above, the defining equation for V(5,0) is

$V(5,0) = 1 + 2.5 * V(3,2)$

Two-Dimensional Function

Where the function is two-dimensional (a function of two node voltages or two branch currents), FV is determined by the following expression:

$$FV = P0 + (P1.FA) + (P2.FB) + (P3.FA^2) + (P4.FA.FB) + (P5.FB^2) + (P6.FA^3) + (P7.FA^2.FB) + (PA.FA.FB^2) + (P9.FB^3) + \ldots..$$

For a two-dimensional polynomial, the controlled source is a function of two nodal voltages or currents. To specify a two-dimensional polynomial, set POLY(2) in the controlled source statement.

Three-Dimensional Function

For a three-dimensional polynomial function with arguments FA, FB, and FC, the function value FV is determined by the following expression:

$$FV = P0 + (P1.FA) + (P2.FB) + (P3.FC) + (P4.FA^2)$$
$$+ (P5.FA.FB) + (P6.FA.FC) + (P7.FB^2) + (P8.FB.FC)$$
$$+ (P9.FC^2) + (P10.FA^3) + (P11.FA^2.FB) + (P12.FA^2.FC)$$
$$+ (P13.FA.FB^2) + (P14.FA.FB.FC) + (P15.FA.FC^2)$$
$$+ (P16.FB^3) + (P17.FB^2.FC) + (P18.FB.FC^2)$$
$$+ (P19.FC^3) + (P20.FA^4) + \ldots\ldots\ldots$$