Integrated Link Level and Circuit Level Simulation of Mobile Communication Systems

Jeyhan Karaoguz

Center for Communication and Signal Processing
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, North Carolina
November 1989

CCSP TR-89/25
Abstract

KARAOĞUZ, JEYHAN. Integrated Link level and Circuit Level Simulation of Mobile Communication Systems. (Under the direction of Michael B. Steer and Sasan H. Ardalan)

This thesis discusses the development of a simulation environment for the analysis and design of RF communication systems from the link level down to the nonlinear RF circuit level. The integrated simulation environment supports block diagram representations, multirate sampling, and an interactive graphical user interface. It enables the incorporation of sophisticated user models of individual blocks, in a mixed time-domain and frequency-domain environment.

The simulation of a multi-channel mobile communication system is presented. The integrated link level and circuit level simulation environment is used to determine the large-signal nonlinear distortion effects such as harmonic distortion, intermodulation distortion and spurious noise which are due to the RF receiver front-end. Also presented are the results of large-signal distortion effects on the adjacent channel interference and signal to noise ratio.

Finally, the simulation results are compared to measured receiver front-end performance for single-tone harmonic distortion.
Table of Contents

1 Introduction

1.1 Motivation For This Study ................................................. 1
1.2 Thesis Overview .............................................................. 2

2 Review of Computer-aided Techniques for Communication System Analysis and Design

2.1 Discussion of Terms: Modeling, Simulation, and Simulation Model 5
  2.1.1 The Aid of Simulation Modeling in Communication Systems Analysis and Design ............................................. 6
2.2 Review of Previous Work ...................................................... 7
  2.2.1 Computer-aided Analysis and Design Techniques ............... 8
  2.2.2 Simulation Based Methods ............................................. 10
  2.2.3 Communication System Modeling ..................................... 14
2.3 Discussion and Conclusion .................................................. 19

3 Integrated Simulation Environment Design .................................. 21

3.1 Introduction ........................................................................... 21
3.2 General Structure and Functions of the Integrated Simulation Environment .................................................... 22
  3.2.1 Link level Simulation : CAPSIM ..................................... 23
  3.2.2 Circuit Level Simulation : FREDA ................................... 28
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2 Harmonic Distortion</td>
<td>54</td>
</tr>
<tr>
<td>6.3 Intermodulation Distortion</td>
<td>58</td>
</tr>
<tr>
<td>6.4 Noise Distortion</td>
<td>65</td>
</tr>
<tr>
<td>6.5 Adjacent Channel Interference</td>
<td>72</td>
</tr>
<tr>
<td>6.6 IF Spectrum for 29 Incommensurable Input Frequencies</td>
<td>80</td>
</tr>
<tr>
<td>7 90 kHz Experiment</td>
<td>82</td>
</tr>
<tr>
<td>7.1 Introduction</td>
<td>82</td>
</tr>
<tr>
<td>7.2 Experimental Setup</td>
<td>82</td>
</tr>
<tr>
<td>7.3 RF Power vs. IF Power Measurements</td>
<td>87</td>
</tr>
<tr>
<td>7.4 Noise Measurements</td>
<td>89</td>
</tr>
<tr>
<td>7.5 Conclusion and Discussion</td>
<td>93</td>
</tr>
<tr>
<td>8 Conclusion</td>
<td>96</td>
</tr>
<tr>
<td>8.1 Summary</td>
<td>96</td>
</tr>
<tr>
<td>8.2 Discussion</td>
<td>97</td>
</tr>
<tr>
<td>8.3 Suggestions for Further Study</td>
<td>99</td>
</tr>
<tr>
<td>References</td>
<td>101</td>
</tr>
<tr>
<td>Appendices</td>
<td>106</td>
</tr>
<tr>
<td>A Detailed Software Structure of Integrated Simulation Environment</td>
<td>106</td>
</tr>
<tr>
<td>A.1 Introduction</td>
<td>106</td>
</tr>
<tr>
<td>A.2 CAPSIM Software</td>
<td>106</td>
</tr>
</tbody>
</table>
A.3  FREDA Software ............................................. 107
A.4  Interface ...................................................... 109

B  User’s Guide ..................................................... 113
   B.1  Introduction ................................................. 113
   B.2  CAPSIM Environment ....................................... 113
   B.3  FREDA Topology ........................................... 120
   B.4  Terminal Session ........................................... 124

C  Stars ............................................................ 127
Chapter 1

Introduction

1.1 Motivation For This Study

Digital communications and computers are having a tremendous impact on the world today. The number of technologies available for providing a given service is growing daily, covering transmission media (microwave and coaxial cable transmission, optical communication via fibers and space satellite transmission), devices (GaAS and silicon devices and VLSI circuits), and software (protocols, system administration, diagnostics, and maintenance). The resulting design, analysis, and optimization of performance can be very demanding and difficult.

Many computer-aided techniques have been introduced in recent years to assist in the modeling, analysis, design and testing of communication and signal processing systems. Although there have been significant developments in computer-aided modeling, analysis, and design of communication systems, there are still some essential issues which have not been addressed strongly or not at all. One of the important issues is the amalgamation [1] of several levels of a system into a single simulation package which can handle the hierarchical top-down analysis and design of communication circuits, links, and networks. Integrating network, link, and
circuit level simulation is essential because different layers of the system strongly affect the performance of others, and sometimes it is extremely difficult to isolate a part of the system from the others except in some oversimplified cases.

The problem of estimating the performance of the circuit level in the presence of all link level sources of impairments such as inter- and co-channel interferences, up- and down-link noise [2] necessitated the design of an integrated simulation environment. The objective of this study is the development of such an integrated simulation environment concept. In particular, the study presents a multi-channel mobile communication system simulation to verify the operation of the proposed integrated simulation environment.

The following section provides an overview of the organization of this thesis.

1.2 Thesis Overview

Chapter 2 presents a thorough literature review of the computer-aided techniques for communication system analysis and design. Different level simulation tools are described and their performances are evaluated. Another purpose of this review section is to familiarize the reader with the large amount of terminology used in this area, and place the proposed simulation environment into the proper category.

Chapter 3 is devoted to the development of the integrated simulation environment. The different level simulators, CAPSIM and FREDA, which were used to build up the integrated simulation environment, are described and detailed. The
concepts and procedure in interfacing FREDA into CAPSIM are given. The features of the integrated simulation tool are summarized briefly at the end of that chapter.

Chapter 4 explains the reasons for choosing mobile communication systems as the simulation example. The technical problems which make the mobile communication systems challenging for design and analysis purposes are presented. Finally, the chapter ends with the detailed description of the general mobile communication system at the link and circuit level.

Chapter 5 focuses on the implementation of the simulation example. After the introduction of the simulation procedure, the functional descriptions of the individual blocks which build up the system are given. The incorporation of the link level and circuit level impairments into the operation of functional blocks is presented.

Chapter 6 presents the results of the simulation example implemented in Chapter 5. The results include the harmonic and spurious response of the nonlinear RF mixer circuit, intermodulation distortion, adjacent channel interference and noise distortion. Along with the results, the large signal nonlinear distortion effects of the link level impairments on the RF circuit level are discussed.

Chapter 7 presents the 90 kHz receiver front-end experiments. First the experimental setup is introduced. Then the results for conversion loss (RF power vs. IF power) and noise measurements are presented. Finally, the experimental results
are compared with the simulation results, and their discrepancies are discussed.

Chapter 8 contains the summary, conclusion and suggestions for future study.

Appendix A presents the detailed software structure of the integrated simulation environment. The user's guide for the simulation tool is given in Appendix B. The C-programming codes for the stars (building blocks) are provided in Appendix C.
Chapter 2

Review of Computer-aided Techniques for Communication System Analysis and Design

2.1 Discussion of Terms: Modeling, Simulation, and Simulation Model

Before starting the review section, we would like to clarify and discuss the meanings of the terms *modeling*, *simulation*, and *simulation model* since these terms are frequently used throughout the thesis.

Modeling and simulation have been practiced ever since there were difficult problems to solve and complex objects to describe. Modeling can be simply defined as creating any abstract representation of something or some activity short of the real thing or the thing-to-be. For example, a model can be a mental picture or concept, a sketch, a description, a set of plans, a set of mathematical equations, a computer program, a pattern, etc. It is a means by which something which is modeled can be explained, expressed, presented, or studied. Simulation, as defined by Webster's, is "the act or process of giving the appearance or effect of". Thus, in general, the two terms *modeling* and *simulation* are quite synonymous. The major distinction between the two terms, however, is that modeling deals with
"form" and "function", whereas simulation deals with "effect". In a rather limited sense one may say that simulation is a process by which model(s) of something or an activity is exercised or stimulated to produce the effects (on certain objects) similar to or relatable to that of the real object or activity. It is also in this rather limited sense that the term simulation model is used — a model specifically built for simulation.

2.1.1 The Aid of Simulation Modeling in Communication Systems Analysis and Design

The complexity of communication and signal processing systems has grown considerably during the past years. As the growth in the subtleness of the system has increased, the analysis and design have become demanding and difficult. Also the rapid emergence of the new technologies has made many existing products obsolete and therefore has created a competitive market for new products. This aspect of technology requires engineers to design new products in a timely, cost effective, and error free manner. These demands can be met only through the use of powerful computer-aided analysis and design (CAAD) tools [3].

A large body of computer-aided techniques have been developed during recent years to assist in the process of modeling, analyzing, designing and testing communication and signal processing systems. Each of them has its own approach to system simulation. These techniques rely on models of devices and systems,
both analytical and simulation, to guide the analysis and design throughout the
life cycle of a system [4]. Performance evaluations and tradeoff analysis are the
central issues in the design of communication networks, links and circuits. Dig­
ital simulation provides a useful and effective means to direct analytical evalu­
ation of communication system performance. Indeed there are many situations
where explicit performance evaluation defies analysis and meaningful results can
be obtained only through either actual prototype hardware evaluation or digital
computer simulations [4].

The interactive nature of different levels — network, link, circuit and signal
processing level — make the design and analysis of communication systems a chal­
lenging task for engineers. Therefore in the communications industry, designers
are looking into means to analyze the whole communication system interactively.
The top-down analysis of communication systems is essential because the perfor­
mançe and response of the circuit level strongly depend on the network or link
level, and vice versa. Except for some idealized and often oversimplified cases, it is
extremely difficult and time consuming to find analytical or closed form solutions
to this aspect of the system.

2.2 Review of Previous Work

In this section, computer-aided modeling, analysis, design and simulation tools
for communication networks, links and circuits, signal processing are surveyed.
The classification of CAAD tools and their evolutions are presented. Also the performances of these CAAD tools are evaluated.

2.2.1 Computer-aided Analysis and Design Techniques

Computer-aided techniques for system analysis and design can be divided into two categories [5] :

- *Formula Based Approach* — In this approach, computer is used to solve sets of equations in finding closed-form solutions.

- *Simulation Based Approach* — In this case, computer simulates voltage and current waveforms or signals that flow through the system.

The basic distinction between the two is that formula-based models are implemented as mathematical expressions that are usually oriented towards some measures of merit (e.g., performance), whereas simulation models are designed to emulate functions and are oriented towards stimulus and response (e.g., input-output signals). Both of these approaches have unique sets of advantages and disadvantages. These are further discussed as follows.

The formula-based model [6] requires only a suitable mathematical model to represent, for instance, the performance of a receiver subsystem such as the demodulator/detector. The advantages [5] of the formula-based or so called analytical modeling approach are that:
i) numerous previous results can easily be utilized.

ii) computer implementation is usually quite straightforward and efficient without requiring a great amount of computer storage.

iii) computation time is generally short and, hence easily accommodated.

The major disadvantages of the analytical modeling approach are the lack of flexibility in representing different implementation methods, and the limited depth or accuracy of the system which depends on the adequacy of the analytical description. Also, the formula-based mathematical techniques require that the system models be simple, thus serving to broadly explore the design space.

In the simulation modeling approach, the same demodulator/detector as mentioned above would have to be numerically represented and implemented. In effect, the numerical implementation would be closely equivalent to actual hardware implementation. Given the required numerical model, a simulated numerical input signal(s) could be applied to the simulated demodulator/detector. The advantage of this approach is that the simulated demodulator/detector could be used again and again to generate and evaluate the responses to different input signal characteristics. Additionally, the simulation approach does not require a priori information about the received signal vector. With simulation-based techniques, systems can be modeled to any arbitrary level of detail, and hence the design space can be explored more finely. The major disadvantage, however, is that a great amount of computer storage and run time is required. However, the ongo-
ing advancements in computer technologies suggest a promising solution to the disadvantages of the simulation modeling approach.

Based on the above observations, the best approach would be to structure the simulation software primarily to support the simulation modeling approach with enough flexibility to support the incorporation of analytical models. As an example, CLASS [7,8] uses a combination of waveform level simulation and formula-based analysis for evaluating the performance of satellite communication systems. Although, these two different techniques are complementary in that many simulation based techniques now incorporate formula based results to reduce the computational burden, the simulation based approach has become the primary tool for modeling and analyzing complex communication systems, and therefore is the subject of this thesis.

2.2.2 Simulation Based Methods

The simulation based methods have gone through different phases throughout their existence. The aspects that have changed are their approaches to communication system simulation, the languages of simulation packages, and the host computers used.

First, an examination of the different approaches to communication system simulation shows that in the early days, users wrote individual programs to handle each application with very little attention paid to software reuse and modular
design [3]. These programs were typically executed on a main-frame computer in a batch mode, and the simulation output consisted primarily of numbers and tables. A considerable portion of the engineer's time was spent in writing and debugging the programs rather than on modeling and analysis [3]. Block-oriented approaches to modeling and simulation such as MIDAS, SCADS, and CSMP [9,10,11] were developed in the early 1960's for digital simulation of analog systems. This approach eliminates the need for the user to write simulation code. Block-oriented simulations draw their motivation from the analog block diagram as a simple and convenient way of describing continuous systems [12]. Advances in discrete-time systems and signal processing have led to new approaches for digital communication system simulation. Software packages for simulations based upon transform domain techniques (FFT for frequency domain and bilinear-Z transform for time domain techniques) began to emerge in the late 1960's and early 1970's. SYSTID [13,14], CMSP, CHAMP [15,16], LINK [17] and others [18,19] were developed in this time frame. The state of the art block-oriented simulators, such as BLOSIM/CAPSIM [20,21] and BOSS [22], provide an interactive framework for simulation-based analysis and design of communication systems with capabilities to 1) develop simulation models in a hierarchical fashion using graphical block diagram representations, 2) configure and execute waveform level simulations, 3) review the results of simulations, and 4) perform design iterations.

Second, computer languages of simulation packages have gone through some
important changes. In the early days, individual programs were written in higher level languages such as FORTRAN etc. Unfortunately, those higher level languages lack the opportunity [4] to allow input via block diagrams which is an essential part of the block-oriented approach. Therefore today's software simulation packages tend to have a pre-processor to a lower level language. This approach lends itself to portability and frees the user from having to know the details of the underlying operating system and the hardware configuration of the simulation facility. The pre-processor STARGAZE [23] of CAPSIM is an example of the state of the art.

Finally, the host computer environment has changed as the computer hardware advanced. In the early days, circuit and control analysis software tools (e.g., ECAP, SCEPTRE, MIMIC [14]) or analog computers were used to simulate certain systems in the time domain. However, both methods had serious drawbacks, including excessive computer run time and step-up complexity. This was particularly true when addressing complicated problems involving performance analysis measurements, such as noise power ratio, and when simulating systems with low bit error rates. In the past ten years, computer hardware and software have undergone significant changes. Powerful workstations such as the SUN, APOLLO, VAX workstations and personal computers like the IBM System 2 and Macintosh 2 offer friendly computing environments with highly visual and graphical user interfaces. Considerable effort is currently directed towards developing intelligent and user friendly simulation packages that take advantage of the latest developments in
workstations as well as PC technologies [12]. Two examples of simulation packages that take full advantage of the hardware/software environments offered by workstations are CAPSIM [21] and BOSS [22]. These provide an interactive framework for simulation-based analysis and design of communication systems with the capability for developing simulation models in a hierarchical fashion using graphical block diagram representations. For many applications, PC’s are preferred because they not only provide sufficient power for simulation-based analysis and design of communication systems, but they are also low cost and provide complete control over the simulation environment since they are rarely a shared resource. Some examples of PC-based software packages are the PC version of ICS [24] denoted as WCS, and MODEM [25].

Numerous simulation packages have been introduced, serving different levels of the communication systems. Desirable features of a simulation package are defined as follows [12]:

- **Modular Structure** — In order to provide maximum flexibility, simulation packages used to aid communication system analysis and design should have a modular structure.

- **Simulation Programming Language** — Fortran, Pascal, or C do not allow input via block diagrams. Therefore a pre-processor simulation language is needed, or object oriented languages such as C++ and objective C may be used.
• Topological Configuration — The model configurator of the simulation pack-
age should permit the design engineer to connect the functional blocks in
any desired topological interconnection, providing maximum flexibility.

• Model Library — The usefulness of a simulation package depends heavily
on the availability of a model library containing many models of various
functional blocks that make up transmission systems and networks.

• Time and Event Driven Simulation — For maximum flexibility, provisions
should be made for both time driven and event driven modes of processing
so that event driven blocks can be used interchangeable with time driven
blocks.

• User Interface — Finally, the whole simulation package should be made user
friendly.

2.2.3 Communication System Modeling

For the simulation of communication systems, it is convenient to model the system
using three levels as in Fig. 2.2.1.

The network level [26] is made up of switches (nodes), multiplexers, and con-
centrators connected by means of transmission systems. The transmission link is
composed of communication media (transmission channel and noise), modulators,
demodulators, encoders, decoders, and filters. The circuit level includes amplifiers,
Network Level CAAD Tools

Network level CAAD tools are based on computer aided analytic techniques, simulation, or emulation. In the first two techniques, which are sometimes combined, the network performance is evaluated using abstracted functional models [27]. In the third technique, the user specifies the network protocols in great detail [28], often supplying an actual program implementing the protocol, and these protocols are then simulated. Tools in this category are called emulators or testbeds [3].

The simulation model of a network includes the network topology, and proto-
The topology is normally specified in terms of nodes and links. Graphical techniques or tables are used to specify the nodes and interconnecting links. The modeling of the protocols involves specifying the data structures and the rules that govern the flow and modifications to the data structures. Three broad approaches are currently being used for this. The first approach consists of graph based models such as finite state machines, and petri-nets. These modeling approaches are typically used for protocol specification and validation. The second modeling approach uses an extended queuing paradigm. This model is suitable for analytic solution of network performance obtained via queuing theory. In the third class of models, the protocol functions are specified as a collection of interacting procedures. This approach is based on the fact that network protocols are algorithmic in nature, and a signal flow type block diagram can express the algorithmic nature of protocol functions clearly.

In network level CAAD tools, event driven simulation is used, since network functions process packages of data such as arrival and departure of messages, occurrences of transmission errors, etc [12]. Several simulation based CAAD tools for communication networks are available. $BONeS$ [29] is an example in this category.

**Link Level CAAD Tools**

Whereas network level CAAD tools use event driven simulations, link level CAAD tools use time driven waveform simulations. The communication link is modeled in
a signal flow block diagram form where each block performs a functional operation like modulation, coding, filtering, etc. The simulation approach consists of generating sampled versions of all the input signals, processing them through functional blocks in the model, generating sampled values of the output signals and analyzing them to extract performance measures. At the link level, the end-to-end performance [26] is measured in terms of waveform distortions, output signal-to-noise ratios, and error probabilities as a function of link level parameters such as filter bandwidths, power levels of input signals and noise, operating point of amplifiers and nonlinear circuits, and so forth.

Systems and signals are represented in the discrete time domain using sampling, and bandpass signals are usually represented by their complex lowpass equivalents [30]. Monte-Carlo techniques [31,32,33] are used to handle random variations in signals and systems and simulation is run long enough to accumulate enough samples to obtain statistically valid estimators of performance measures. However the computational burden of Monte Carlo simulations is considerable. For example, if Monte Carlo simulations are used to estimate the unknown error probability in a digital communication link, several million symbols will have to be processed during the simulation in order to obtain statistically accurate estimators of low probability (on the order of $10^{-6}$) [26]. The processing time for this may approach the order of several hours to days. A number of techniques are currently being developed to reduce the computational burden [34].
CAPSIM [21] and BOSS [22] are two examples of link level CAAD tools.

Circuit and Signal Processing Level CAAD Tools

One of the primary areas in which engineers are most involved is circuits. Until now, many software packages have been introduced such as *SPICE* [35,36] and *FREDA* [37]. The basic difference between link level and circuit level simulation packages is that link level CAAD tools are generally simulation environments for time driven waveform simulations whereas circuit level CAAD tools are based on solving a set of linear or nonlinear differential equations resulting from application of Kirchoff's voltage law and Kirchoff's current law using constitutive relations (i.e., the element characteristics). The methods for circuit level simulation may be characterized into three groups by the way in which the circuit elements are treated: *Time Domain Methods, Hybrid Methods, and Frequency Domain Methods* [37].

In time domain methods both linear and nonlinear elements are analyzed in the time domain. The numerical integration of the circuit parameters is used. The time domain methods are useful if the transient response is desired but inefficient when the circuit has widely varying time constants or the signal is widely separated in frequency. A widely used time domain circuit simulator is *SPICE* [35,36]. The hybrid methods use time domain analysis for the nonlinear elements, and frequency domain analysis for the linear elements [38,39]. These methods directly calculate the steady-state response of the nonlinear circuit. In frequency domain methods,
the entire circuit is analyzed in the frequency domain without transformations between the time and frequency domains. \textit{FREDA} is an example of a frequency domain circuit simulator [37].

The availability of low cost/high speed signal processing hardware has allowed the increasing use of digital signal processing in many areas. With the drive towards all digital communication networks, significant portions of the communication systems use complex signal processing techniques, and an increasing need exists for CAAD tools which can aid in the design of these systems. Many of the currently available signal processing simulation tools perform analysis and design of only specific DSP algorithms and functions, such as FIR and IIR filtering. One of the few examples in this category is \textit{SPW} (Signal Processing Worksystem) [3].

2.3 Discussion and Conclusion

In this chapter, computer-aided design, analysis, and simulation tools for communication networks, links and circuits were surveyed.

There is no doubt that the complexity of modern communication systems whether they are at the link, network or circuit level, require an extensive use of CAAD tools. While simulation is a very powerful tool for the analysis and design of communication systems, it imposes some problems to be addressed as well.

These problems can be classified as follows:
• *The heavy computational burden* — for example, Monte Carlo simulation of coded communication systems operating at low error rates may take several hours of cpu time.

• *The development of an expert system for simulation based design* — At the present time simulation is primarily used as an analysis tool, with the engineer making heuristic choices for design iterations on a trial and error basis which is time consuming and inefficient.

• *The integration of simulation tools* — There is a lack of integration between simulation tools within and between the various levels of the hierarchy shown in Fig. 2.2.1. There is a considerable amount of "hand transfer" of data between various CAAD tools which is the cause of numerous design errors and delays. An integrated environment for CAAD tools is essential for improving productivity and minimizing costly errors.

In this thesis, we are going to deal with the latter problem and present a simulation environment which integrates different level CAAD tools, namely, the link and circuit level. The practical use of this integrated simulation environment will be presented by a simulation example.
Chapter 3

Integrated Simulation Environment Design

3.1 Introduction

Conceptually, the need for amalgamating the link level simulation with that of the nonlinear RF circuit level led to the design of an integrated communication system simulation environment. The main goal for this integrated environment is the development of an hierarchical approach to simulation which permits the use of multiple tools within the environment and a hierarchy within each level of the system as well. A simulation tool at an intermediate level accepts data from the lower level modules and provides data to the models used in the next higher level. By loosely integrating the simulation tools at successive levels in the hierarchy through a flexible interface, an integrated environment has been achieved for the simulation and design of communication and signal processing systems. Also considered were the provisions for merging different circuit level simulators into the integrated simulation environment.
3.2 General Structure and Functions of the Integrated Simulation Environment

The integrated simulation environment is a mixed time domain and frequency domain simulator. Mostly frequency domain processing has been employed in the simulation environment. Frequency domain processing is more advantageous for linear functional blocks such as filters. It permits the use of measured values of filter characteristics such as amplitude and phase response [4].

The sampling rate is an important issue in communication system simulation. Especially for RF communication systems, the baseband signal can be many orders of magnitude lower than the frequency of the RF signal. This has led to the complex envelope representation of signals and baseband equivalents of subsystems to reduce the sampling rate [30,40,41]. However, when we are dealing with nonlinear systems, this technique is inappropriate since nonlinear characteristics involved in the frequency translation from RF to baseband are not captured. By the same token, any functional operation such as filtering or passing the signal through a nonlinear transmission medium has different effects on the baseband signal than it would have for the RF signal. This characteristic of the system precludes using the complex envelope and baseband equivalent representations of the bandpass signals. In time domain analysis, a remedy seems to be to increase the sampling rate, but this is very inefficient. For the proposed integrated simulation environment, this inefficiency is avoided by working in the frequency domain.
for bandpass signal processing. What is meant by representing the bandpass signals in frequency domain is that only the RF frequency part of the spectrum is retained since the spectrum elsewhere is redundant. Accordingly, all functional blocks process the RF signal spectrum at this RF carrier related frequency.

The designed CAAD tool can be divided into two parts, i.e., CAPSIM (link level simulator) and FREDA (nonlinear RF circuit level simulator). CAPSIM and FREDA have different approaches to simulation since they are dealing with different levels of the system. The integrated simulation environment is the appropriate interface of FREDA to CAPSIM. Before proceeding with the interface procedures, next two sections describe CAPSIM and FREDA.

3.2.1 Link level Simulation : CAPSIM

CAPSIM [21] is a complete interactive simulation environment for the design and analysis of communication systems. The integrated environment includes the capability to design models, subsystems, and systems in a hierarchical fashion using block diagrams. CAPSIM is a refinement and extension of the time-driven block diagram simulator BLOSIM [20]. The BLOSIM simulator was chosen because of its hierarchical block orientation and many other desirable features; however it has the drawback of being text oriented. CAPSIM was designed to allow BLOSIM to be used much more easily and efficiently.

CAPSIM uses a functional approach to simulation. A user specified block op-
erates on current inputs to produce current outputs. CAPSIM also allows the user to specify the design in a hierarchical manner using block diagrams which are familiar to system designers. CAPSIM provides a complete interactive environment for communication system design and engineering. The integrated environment includes the capability of defining blocks and models in a hierarchical fashion, configuring and executing a simulation, reviewing the results of simulation, and performing design iterations. Each of these capabilities is controlled through a user interface using interactive graphical techniques.

CAPSIM is essentially a time-driven simulator and each block is polled at each sampling instance. Multirate sampling techniques enable the simulation of many systems whose blocks have different bandwidths.

**CAPSIM Approach**

Philosophically, CAPSIM is closest in approach to writing a special-purpose simulation program, and offers similar advantages of efficiency and generality. At the same time it overcomes the deficiencies of many special-purpose simulation efforts by providing a structured environment for programming simulations. This structured environment encourages the programmer to use highly structured programming techniques, but without requiring any prior familiarity with these techniques. This means that CAPSIM frees the user from having to contend with simulation house-keeping duties. Instead, the user may concentrate on the functions of the simulation blocks. This structured simulation environment pays two major divi-
dends. First it encourages the writing of simulation code for small pieces of the system being simulated, and this code is written to a single carefully defined interface. As a result libraries of routines, re-usable in future simulations, can be developed. Second, multiprogrammer simulation efforts become relatively painless since routines written by different programmers readily interface to one another [20].

The terminology that is used in CAPSIM in describing the different types of blocks is, as in BLOSIM, modeled after cosmology: A block which is elementary or atomic in the sense that it is implemented directly by a user-provided routine is called a star. A block which is defined as a connection of other blocks, on the other hand, is called a galaxy. Finally, a special galaxy is defined which encompasses the entire systems, and which therefore has no inputs or outputs. Such a galaxy is known as the universe. By definition, unlike all other blocks, the universe can only have a single instance. The cosmology of blocks differs from ordinary cosmology in that a galaxy need not contain exclusively stars, but rather can contain other galaxies. There is no limit on the nesting of galaxies [20].

The Pre-processor

A star consists of code written in pseudo C-code. The statements themselves are legal C operations, but the star is not an executable C-program. Before the star can be used, it must be passed through the stargaze pre-processor which produces a C file. The resulting output is now a valid CAPSIM star. The stargaze pre-
processor uses *awk*, a pattern scanning and processing language, which translates pseudo C-code into an ordinary C-program. The user can easily define new stars to perform almost any function of interest. To facilitate the introduction of a new star into CAPSIM, a pre-processor called *pre-capsim* has been designed. This pre-processor creates the object code for the star, and after leaving a copy of this code in an individual user library, links this code with the remainder of the object modules related to CAPSIM. The result is the creation of a customized version of CAPSIM which contains only the stars needed by that specific user. This allows different users to have their own customized version of CAPSIM, containing the stars necessary for their individual simulations.

**Simulation Configuration**

CAPSIM supports a hierarchical definition of blocks. That is, it is simple to define new blocks which are made up of specified interconnections of other blocks. In subsequent simulations, these higher level blocks are indistinguishable from blocks implemented directly by a user program. In addition, multiple instances of blocks are supported. That is, the same block can appear as many times as desired in a given simulation (usually its functionality is specified by setting different values for its parameters). This is particularly convenient for simulation systems which have a replication of similar or identical functions (for example a systolic array of or a system with multiple filters). CAPSIM also makes many consistency checks during topology definition and execution, allowing it to check and detect many
user programming errors. The scheduling of the order of execution of the blocks in the universe is done without regard to whether the blocks are stars or galaxies. All the blocks are examined to find any which have no inputs. These blocks are scheduled to execute in arbitrary order. At step \( k \), all the blocks which have not been scheduled in steps 1 through \( k-1 \) are examined to find any which have inputs connected exclusively to blocks which have been scheduled in steps 1 through \( k-1 \). Blocks satisfying these criteria are scheduled next in arbitrary order. These steps are repeated until all blocks within a universe have been scheduled.

A user defined topology is constructed by selecting blocks from the menu and placing them on the work space. As each block is selected, the user will be prompted for parameters if needed.

**The Post Processor**

The results of simulation are displayed using signal plotting and processing techniques. The outputs of the simulation may be sent to a text file for manipulation by another program or can be used as inputs to subsequent CAPSIM simulation. The post processor uses the same interactive graphic interface to allow the user to select signals from the block diagram for display and manipulate the plotting of these signals.
3.2.2 Circuit Level Simulation: FREDA

FREDA is a nonlinear RF circuit simulator described in [37] and [42]. Although the methods for analyzing nonlinear RF circuits are varied, they are all based on solving a set of nonlinear differential equations resulting from application of Kirchhoff’s voltage law and Kirchhoff’s current law using the constitutive relations (i.e., the element characteristics). The methods fall into three groups according to the way in which the nonlinear elements are treated: time domain methods, hybrid (harmonic balance) methods, and frequency domain methods. Time domain methods analyze the nonlinear circuit by solving the nonlinear differential equations governing the circuit in the time domain using a method related to numerical integration [43]. They are generally regarded as unsuitable for RF and microwave circuit analysis. Microwave and RF circuits typically have widely separated time constants resulting in a set of stiff state equations. The consequence is that a small time step must be chosen and simulation must proceed for a large number of time steps leading to excessive computation time. Furthermore, it may be difficult to identify the steady-state solution when closely spaced frequencies are present.

The hybrid methods avoid the numerical integration of the state equations as required in the time domain. These methods directly calculate the steady-state response of the nonlinear circuit, and are generally referred to as harmonic balance methods. The modern version of the harmonic balance method was presented in [38]. In that study, the number of variables to be optimized was reduced by
partitioning the network into smaller sub-networks that are composed of either linear circuit elements or nonlinear elements. The linear sub-networks are solved in the frequency domain while nonlinear sub-networks are solved in the time domain. Several variations of the harmonic balance method in [38] have been proposed to handle nonharmonically related signals. The modified harmonic balance methods are presented in [39,44,45,46,47].

The frequency domain methods avoid explicit time domain calculations. This is accomplished through the use expansion of the input-output characteristics of the nonlinear elements in a set of basis functions. There are three types of basis function expansions: Volterra series, algebraic functional expansion, and power series. Volterra series techniques can only be used to analyze weakly nonlinear systems [37]. The algebraic functional expansion method is related to Volterra series, but the analysis procedure is quite different [48]. With frequency domain spectral balance analysis, power series based techniques are proving to be the most general methods for the analysis of nonlinear circuits with multi-frequency large signal excitations and are easily integrated into existing computer-aided design tools [37].

FREDA implements the harmonic balance and frequency domain methods of nonlinear RF circuit analysis [37,42]. Both methods implement the spectral balance procedure as follows. A nonlinear analog circuit is partitioned into linear and nonlinear subcircuits as depicted in simplified form in Fig. 3.2.1.
In this example the frequency domain form of the voltage $v = v'$ (i.e. a set of phasors) across both the linear and nonlinear elements is assumed, and the current $i$ (as a set of phasors) into the linear subcircuit is calculated using standard linear circuit techniques [43]. Then the current $i'$ (again as a set of phasors) flowing into the nonlinear circuit is calculated using the almost periodic Fourier transform technique [46,47], the multidimensional fast Fourier transform technique (NFFT) [49], or the arithmetic operator method (AOM) [37]. (Of these the NFFT method is preferred for the simplicity in nonlinear device modeling and the AOM technique for speed and memory usage when there are two or more incommensurable signals.) Whichever method is used in determining the nonlinear current $i'$, a balance of $i$ and $i'$ is achieved using a Newton iteration scheme to update the estimated voltage at the linear/nonlinear interface.
3.2.3 Interface of FREDA into CAPSIM

Fig. 3.2.2 illustrates the structure of the integrated simulation environment, particularly as it relates to the user interface. The user provides three types of routines: a system topology, *star* routines (could be system-provided as well), and a netlist for FREDA. These routines interface to CAPSIM (rather than directly to one another) which handles FIFO buffer management. The user-supplied "topology definition" program specifies the names of *stars*, the routine which implements each *star*, and how the *stars* are interconnected. The topology definition routine is also able to pass parameters to the *stars* to specialize their function. The *stars* get their input samples from input FIFO's, and put their generated output samples into output FIFO's.

The *functions*, in Fig. 3.2.2, are particular routines to be called by a *star* when they are needed. Those functions could be an FFT (Fast Fourier Transform), digital filters, or any other user defined mathematical functions. The convenience introduced by the *functions* is that the user can easily implement frequently needed routines in *stars* simply by calling *functions*. FREDA was interfaced into CAPSIM as one of these *functions*. More information as to the software details in the interface procedure is provided in Appendix A. Once the simulation is executed according to the user defined topology, FREDA takes over as the nonlinear RF circuit simulator at the appropriate point in the simulation. By interfacing FREDA into CAPSIM, a customized version of CAPSIM is obtained. Since
Figure 3.2.2: Integrated Simulation Environment
FREDA is a complete frequency domain simulator, all CAPSIM stars connected to the FREDA star are forced to process in the frequency domain in order to be consistent with FREDA through the interface. This transformation is managed by FFT stars in the CAPSIM system topology.

3.3 Summary

In summary, the incorporation of the link and the nonlinear RF circuit level simulation of the communications systems has resulted in a flexible integrated simulation environment which enables us to see the effects of link level perturbations on the nonlinear RF circuit level such as the effect of transmission media change on the received signal at physical level and vice versa. In addition, a great deal of efficiency is achieved in representing the filters and bandpass signals in the frequency domain, avoiding the inefficiencies introduced by high sampling rates.
Chapter 4

The Simulation System: Mobile Communications

4.1 Introduction

In choosing an example, we tried to cover a broad range of applications and be fairly general. The simulation of a mobile communication system was suited to this objective because of its complexity and the way that the different levels of the system affect each other. Another motivation in choosing the mobile communication system as an example is the practical need for an integrated simulation environment for the design and analysis of mobile communication systems. This is because, in industry, engineers carry out some field experiments which change the circuit designs or the modulation schemes accordingly. However, this is a very costly approach both financially and in terms of the time spent by the engineers to construct the experiment and take the data. Therefore, an integrated simulation environment is essential in the design and development of communication circuits and new modulation schemes for mobile communication systems.
4.2 Technical Problems in Mobile Communication Systems

The complexity of communication systems has been greatly increased by the presence of mobile radio technology, although some technical and regulatory issues still remained unsolved [50]. The allocation of new transmission channels at 900 MHz added many more problems to circuit complexity and analysis as well [51]. Various technical problems make the mobile communication systems difficult for designers to deal with. These technical problems can be divided into two groups: link level impairments, and circuit level impairments.

4.2.1 Link level Impairments

In the first place, there is a great deal of multipath transmission. Because of reflection from buildings, lamp posts, utility wires, and other obstructions, the signal may travel from the transmitter to the receiver by many different paths with different lengths. When one end of the channel is moving, as is usually the case in mobile communications, the predominant feature of the multipath is fading. The fading results because, in some relative positions, phases of the signals arriving from the various paths interfere constructively. Thus as a vehicle moves, the received signal strength varies erratically and unpredictably over a range of 20 to 30 decibels. Another set of problems that arise in the mobile channel is also the result of the mobile user’s motion. The first, of course, is Doppler shift. At
900 MHz, for example, speeds in the range of 50 miles per hour may result in Doppler spreading of approximately $\pm 100$ Hz [50]. This motion-related problem leads to other problems such as co-channel and adjacent channel interference. In a narrow band system the channels can be separated in frequency by 25, 30, or 50 kHz. Because of vehicle motion, a receiver and a transmitter operating on adjacent or nearly adjacent channels may be physically close together. The receiver selectivity must be extremely good to prevent substantial interference from the strong transmitter signal. Finally, some other problems like intersymbol interference which results from the arrival of signals at the receiver at different times are also due to the motion in mobile radio.

### 4.2.2 Circuit Level Impairments

The problems at the circuit level in mobile communication systems are mostly the large signal nonlinear distortion effects due to the RF receiver front-end. These distortions can be localized in the nonlinear RF mixer where the RF signal is converted down to the IF signal.

As suggested in [52] nonlinear distortion and interference effects in mixers are those effects which represent departure from the ideal performance when the input signal in the passband is large. The nonlinear distortion and interference effects can be classified as:
i) gain compression or enhancement — which is a variation of the mixer conversion gain, or loss, due to the amplitude of the desired input signal.

ii) intermodulation — refers to generation of spurious frequency components at the sum and difference frequencies of the desired and undesired input signals.

iii) desensitization — modulation of the desired signal by an undesired signal.

iv) detuning — generation of dc charge or dc current, due to a large input signal resulting in a change of the diode bias condition.

In addition to the large signal distortion effects, there is yet another distortion effect proportional to local oscillator power as well, which had been studied thoroughly in [53]. In [53], it was concluded that intermodulation will first increase, then reach a maximum, and finally decrease as local oscillator power increases. However, the conversion loss is inversely proportional to the local oscillator power, i.e., the conversion loss decreases as the local oscillator power increases. This fact suggests a compromise between the nonlinear distortion effects and the conversion loss.

Another problem related to the circuit level is the inadequate bandpass filtering at the RF front-end which gives rise to the adjacent channel interference and the intermodulation distortion.
4.3 Description of Mobile Communication System

A block diagram is illustrated in Fig. 4.3.1 of a generic communication system which provides a model for the simulation example. The objective in designing the simulation example is to include the link and circuit level impairments, i.e., to define the individual blocks of the communication system incorporating the impairments mentioned in Section 4.2. The blocks into which these impairments are built, are the transmission medium block, i.e., the channel and additive noise, and the RF front-end block, i.e., the nonlinear RF mixer circuit and bandpass filter. In the following sections the functional blocks in each level are described.

4.3.1 Description of Link Level

The link level consists of transmitters and the transmission medium. Two transmitter units are used in order to simulate a multi-channel system. Each transmitter generates a two-tone signal. It is advantageous to use two-tone signals in the characterization of the channel and in measuring intermodulation distortion [54]. The purpose of the modulator/translator block in Fig. 4.3.1 is to map the two-tone signal into a waveform suitable for transmission over the transmission medium.

The most important block in the link level is the transmission medium block which is composed of a channel and additive white Gaussian noise. In choosing the channel model for the transmission block, it is important to provide a fairly
Figure 4.3.1: Generic Communication System
general capability which can support the impairments mentioned in the previous sections. A physical and practical channel model [40] which satisfies these requirements is a three component fading multipath channel plus additive noise. In particular, the fading multipath channel is assumed to consist of a single direct path, a specular multipath component, and a diffuse multipath component as illustrated in Fig. 4.3.2. The quantities $\xi_d$ and $\xi_s$ in Fig. 4.3.2 represent the specular and diffuse multi-path signal energies, respectively, each normalized to the energy in the direct path. $\phi$ is the relative phase of the specular multipath component relative to the direct path. The differential delay $\tau_o$ and the differential Doppler
$f_o$ are also associated with multipath components. The noise component, $n(t)$ in Fig. 4.3.2, is an additive white Gaussian noise (AWGN). The noise source is detailed in the implementation section, Chapter 5. This channel model includes all link level impairments mentioned in Section 4.2.1.

4.3.2 Description of Circuit Level

At the receiver front-end are the nonlinear RF mixer circuit, bandpass filter, and the other necessary demodulation blocks.

The nonlinear RF mixer circuit is a double-balanced ring diode mixer, as illustrated in Fig. 4.3.3, which is a widely used communication mixer circuit. The double-balanced ring diode mixer was chosen because of its practical use and its better performance on harmonic and intermodulation distortion and RF/IF isolation due to its balanced characteristic [55]. The ring diode mixers can also be used as modulators, phase detectors, and even voltage-controlled attenuators [55]. Therefore, they are very versatile components.

The bandpass filter block is represented in the frequency domain like the other blocks in the system. It uses measured or independently calculated values for amplitude and phase response [4]. This block has an essential role in simulating the adjacent channel interference because it determines the leakage power from the adjacent channel.
Figure 4.3.3: Double-Balanced Ring Diode Mixer Circuit
Chapter 5

Methodology and Implementation of Mobile Communication System Simulation

5.1 Introduction

This chapter illustrates by way of example the approach that a programmer using the proposed integrated simulation environment uses to develop a simulation, particularly the simulation of a multi-channel mobile communication system. The functional descriptions of blocks (*stars*) that build up the mobile communication system are elaborated. The C-programming codes for the *stars* are provided in Appendix B.

Next section presents an outline of the simulation procedure as it relates to the implementation of the simulation example.

5.2 Simulation Procedure

The CAPSIM user can configure a wide variety of digital communication systems from basic functional blocks, i.e., *stars*. In the mobile communication system simulation example, the CAPSIM *stars* include: signal sources, FFT blocks, modulators/frequency translators, a combiner, physical channel with additive white
Gaussian noise (AWGN), a bandpass filter, mixer, and IF filter. The mixer star effectively interfaces FREDA into CAPSIM.

The mobile communication system being simulated is illustrated in Fig. 5.2.1 in terms of its implementation in CAPSIM terminology. The detailed descriptions of the functional blocks follow in Section 4.3. For the purposes of simulation, as well as actual implementation, the system, i.e., the universe is partitioned into a number of galaxies. Those galaxies are further subdivided into stars. Each star is simulated by a user-defined C-program, and the integrated simulation environment combines these individual simulations into a simulation of entire system.

Before proceeding with the descriptions of the stars, we would like to restate and review the communication between CAPSIM and FREDA, and the communication within each simulator as well. As mentioned in Chapter 3, there are two different means to communicate with CAPSIM simulation environment. The first way is through an interactive graphical interface. The second way, which is less user-friendly, is to provide CAPSIM with a user-defined system topology. This approach is rather text-oriented. The graphical interface has the disadvantage that it requires the user to run the simulation on a workstation. Although the text-oriented approach is less friendly and less visual, it is possible to run the simulation on a computer terminal. In this simulation example, the latter way of communication with CAPSIM was employed.

In the simulation example, FREDA was used to analyze a nonlinear RF mixer
Figure 5.2.1: Mobile Communication System *Universe* in CAPSIM terminology
circuit. Particularly, a double-balanced ring diode mixer was chosen. As mentioned in Chapter 3, FREDA uses a circuit net-list to communicate with the user. The net-list specifies the topology, definitions, element values and node numbers of the particular circuit.

Finally the communication between CAPSIM and FREDA is achieved through the FREDA mixer \textit{star}.

5.3 Functional Description of the Simulation Example

The mobile communication system \textit{universe}, as illustrated in Fig. 5.2.1, consists of two transmitter \textit{galaxies} (since adjacent channel interference is of concern there are two transmitters which modulate the information signals up to two different carrier frequencies in the adjacent channels), a combiner \textit{star}, a transmission medium \textit{galaxy} and a receiver \textit{galaxy}.

The rest of this chapter is devoted to the functional descriptions of \textit{stars} which build up the \textit{galaxies} and finally the \textit{universe}.

5.3.1 Transmitter Galaxies

In \textit{transmitter galaxy I}, there are two \textit{signal stars} as shown in Fig. 5.3.1. Each of the \textit{signal stars} generates a sinusoidal signal which is the discrete sequence of the sampled analog waveform. The frequency and corresponding sampling rate of the time domain waveform are specified in the user-defined CAPSIM topology.
Figure 5.3.1: Transmitter Galaxies
through the parameters list. The *adder star* simply performs a sample-by-sample addition of the two input sample streams. Therefore, each *signal star* must have the same sampling rate while generating the sinusoidal waveform. After the adder *star*, a two-tone sinusoidal signal is obtained. The two-tone signal was chosen for its convenience in measuring the intermodulation distortion [54]. Following the *adder star*, the time domain two-tone signal is passed through the *FFT* (Fast Fourier Transform) *star* by which the information is transformed into the frequency domain. The next step is to modulate the baseband signal to a higher frequency suitable for transmission through the channel. This is achieved by the *modulator/translation star* which performs double-sideband suppressed carrier (DSBSC) modulation on the baseband signal [30]. The suppressed carrier frequency and the other parameters are specified in the user-defined CAPSIM topology. The description of the *transmitter galaxy II* is similar since it consists of identical blocks as the *transmitter galaxy I*.

In Fig. 5.2.1, the two *transmitter galaxies* are connected to the *combiner star*. As its name implies, the *combiner star* combines the two different spectrums and sends the resulting RF spectrum through the *transmission medium galaxy*.

### 5.3.2 Transmission Medium Galaxy

As mentioned in Chapter 4, the *transmission medium galaxy* in Fig. 5.3.2 is a multipath fading channel with additive white Gaussian noise. The parameters $\xi$. 
Figure 5.3.2: Transmission Medium Galaxy
and $\xi_d$ in Fig. 4.3.2 (which are the specular and diffuse multi-path signal energies — normalized to the energy in the direct path) are under direct user control. In addition, the user is allowed to adjust the relative phase $\phi$ of the specular multipath component relative to the direct path component. Other parameter choices include the nominal differential delay $\tau_o$ and the differential Doppler $f_o$ associated with the two multipath components. All of these parameters can be modified through the parameters list in the user-defined CAPSIM topology.

The noise star in the transmission medium galaxy is simply a white Gaussian noise generator operating in the frequency domain. The additive noise components are the sampled values of white Gaussian noise spectrum. The variance of the Gaussian noise is a parameter under user control.

5.3.3 Receiver Galaxy

The receiver galaxy, illustrated in Fig. 5.3.3, is composed of a bandpass filter star, a mixer (FREDA) star, an IF filter star, a IFFT (Inverse Fast Fourier Transform) star, and an IF demodulator star. The bandpass filter star is a bandpass filter represented in the frequency domain. It has lower and upper cut-off frequencies and roll-off factor in the transition band which are specified in the user-defined CAPSIM topology. Those parameters are effective in the adjacent channel interference measurements as explained in Section 4.3.2. The RF signal after the bandpass filter star is the desired signal plus the white Gaussian noise
Figure 5.3.3: Receiver Galaxy
added in the channel and the leakage signal from the adjacent channel which results from the inadequate filtering. The mixer star takes the RF spectrum and invokes FREDA to analyze the nonlinear RF mixer circuit illustrated in Fig. 4.3.3. The local oscillator power and the other circuit parameters are specified in the user-defined FREDA net-list. The output of the mixer star is taken by the IFFT star which carries out an inverse Fourier transform on the IF frequency spectrum. Hence, the time domain IF waveform is obtained after the IFFT star. As example, several IF waveforms and spectrums are presented in Chapter 6.

Finally, the IF demodulator star demodulates the IF signal using time domain techniques. The demodulation techniques are beyond the scope of this work. Thus, this particular star is reserved for further study.
Chapter 6

Simulation Results and Discussion

6.1 Introduction

This chapter presents the results of the mobile communication system simulation implemented in Chapter 5. Particularly, the results show the large signal nonlinear distortion effects at the receiver front-end — due to link level impairments such as multi-path fading channel with additive white Gaussian noise and adjacent channel interference.

The mobile communication system is examined for the main performance issues [56] such as harmonic distortion, intermodulation distortion, adjacent channel interference and signal to noise ratio for a desired signal and an undesired interfering signal. The carrier frequency is at 900 MHz which is one of the allocated RF spectrums by FCC for the mobile radio communications [57]. The channel separation is 100 kHz. Therefore, the adjacent channel carrier frequency is at 900.1 MHz. For the results presented, the channel attenuation is changed to increase or decrease the received RF signal power. Typically, while the channel attenuation is increased from 30 dB to 70 dB, the received RF signal power decreases from -20 dBm to -60 dBm. The specular and diffuse paths of the multi-path fading channel
have 10 dB and 20 dB, respectively, more attenuation relative to the direct path (refer to Fig. 4.3.2). The fixed delay in the direct path is 2 micro seconds. The differential delay in specular multipath is 1 micro second. The transmitted power from each transmitter is kept constant at 6.25 Watts which is typical for a medium range mobile radio [58,59]. The bandwidth and roll-off factor of the bandpass filter at the receiver front-end are varied to observe the effects of inadequate filtering on the signal-to-adjacent channel interference ratio. The local oscillator power for the receiver front end is $-4.0$ dBm (referred to 50 $\Omega$) and the diode saturation currents are 30 pA. The saturation currents for the diodes were calculated from the experimental data taken for the ring diode mixer used in Chapter 7. The transformers are virtually ideal. All of the numerical parameters mentioned above are under direct user control and can be modified through the user-defined FREDA and CAPSIM topologies.

6.2 Harmonic Distortion

The suppressed carrier is at 900 MHz, and the baseband signal frequency is 2 kHz. The resultant frequency spectrum and the local oscillator frequency are illustrated in Fig. 6.2.1.

In the harmonic response curve, Fig. 6.2.3, the channel attenuation is changed from 35 dB to 110 dB which results in a received RF signal power variation between 0 dBm and 85 dBm (referred to 50 $\Omega$). In Fig. 6.2.3, the horizontal axis, $RF$ power,
represents the received RF signal power at receiver front-end. The vertical axis, 

*normalized voltage amplitude*, is defined as,

\[ 20 \log \left( \frac{IF \text{ voltage amplitude}}{1 \text{ mV}} \right) \]

For this particular example there is only one sinusoidal signal on the carrier. The first observation is the insignificant amplitudes for the even numbered harmonics which is expected for the ring diode mixer as it produces only odd numbered harmonics. This can be described very simply if the diodes are treated as ideal LO-driven switches [55,60,61]. The mixer in Fig. 4.3.3, therefore, acts as a polarity reversing switch, connecting the RF port to the IF port but reversing its polarity every half LO cycle. The IF voltage is, therefore,

\[ v_{IF}(t) = s(t)v_{RF}(t) \]

where \( s(t) \) is a square waveform illustrated in Fig. 6.2.2, which represents ideal
switching. Consequently, 

\[ v_{IF}(t) = \sum_{n=\text{odd}}^{\infty} b_n \sin(nw_p t) v_{RF}(t) \]

where \( s(t) \) is represented as a Fourier series and \( w_p \) is the fundamental frequency. Since there are only odd numbered harmonics in the sum, this explains the isolation of even numbered harmonics at the IF band.

The second observation is the linear behavior of the ring diode mixer circuit for the low received RF signal powers, i.e., below -20 dBm. However as the received RF signal power increases, the nonlinear characteristic is observed around -15 dBm received RF signal power.

The third observation is the rapid decrease in the power of higher order harmonics compared to the lower order harmonics as the received RF signal power decreases. This agrees with [61]. All three observations are presented in Fig. 6.2.3.
Figure 6.2.3: Harmonic Response Curve, 1st Harmonic Solid Line, 3rd Harmonic Dashed–dotted Line, 5th Harmonic Dotted Line
6.3 Intermodulation Distortion

In many receiver applications, two input signals may enter a mixer and mix with the local oscillator signal to produce unwanted in-band intermodulation products. These two-tone products can be particularly troublesome in a receiver because they are separated from the desired output signals by the same amount as the two input signals. Because of this equally-spaced relationship, it is particularly difficult to provide adequate filtering to remove the unwanted products without affecting the desired outputs [62]. In this simulation example, those effects are examined, and the resultant IF waveforms and spectrums are presented.

To see the intermodulation distortion effects clearly, the baseband signal is chosen to be a two-tone signal at 2 and 3 kHz. The suppressed carrier and local oscillator frequencies are both 900 MHz as illustrated in Fig. 6.3.1. The two-tone intermodulation distortion test system presented in [54] is simulated, and
the resultant signal-to-intermodulation ratio curve is illustrated in Fig. 6.3.2. The signal-to-intermodulation ratio is by definition,

\[ SIR = 10 \log \left( \frac{\sum P_{IM}}{P_{IF}} \right) \]

where \( P_{IM} \) is the power of the individual intermodulation product in the IF band, and \( P_{IF} \) is the power of the desired IF signal component.

The simulation configuration has only one channel carrying a two-tone signal. The intermodulation distortion component is the difference between these two signals. The low received RF powers do not lead to any nonlinear characteristic in the intermodulation distortion curve. However, as the received RF power gets larger, the nonlinear characteristic becomes noticeable around -25 dBm.

The IF waveforms are presented for 45 dB and 55 dB channel attenuation in Fig. 6.3.3 and Fig. 6.3.5 respectively. The IF spectrums are also provided for the same channel attenuations in Fig. 6.3.4 and Fig. 6.3.6. The intermodulation distortion is insignificant for 55 dB channel attenuation which is observed in Fig. 6.3.6 as the intermodulation products are almost 40 dB less compared to the desired signal. Therefore, the waveform in Fig. 6.3.5 is not distorted. However, for the 45 dB channel attenuation there is some distortion which can be observed in Fig. 6.3.3. In this case the intermodulation products are 20 dB less compared to the desired signal. The rapid decrease in the power of intermodulation products is as expected since the decrease in the power of fundamental component causes higher order decreases in the power of intermodulation products as studied in [61].
Figure 6.3.2: Intermodulation Distortion Curve, SIR
Figure 6.3.3: IF Waveform for 45 dB Channel Attenuation
Figure 6.3.4: IF Spectrum for 45 dB Channel Attenuation
Figure 6.3.5: IF Waveform for 55 dB Channel Attenuation
Figure 6.3.6: IF Spectrum for 55 dB Channel Attenuation
6.4 Noise Distortion

There are 23 incommensurable (i.e., nonharmonically related) frequencies in the RF spectrum as illustrated in Fig. 6.4.1. The white Gaussian noise bandwidth is 8 kHz — determined by the bandwidth of the bandpass filter at the receiver front-end. The baseband signal is at 2 kHz.

The signal-to-noise ratio is by definition,

\[ SNR = 10 \log \left( \frac{P_{\text{noise}}}{P_{\text{IF}}} \right) \]

where \( P_{\text{noise}} \) is the total noise power in the IF band, and \( P_{\text{IF}} \) is the total IF signal power.

The SNR curve in Fig. 6.4.2 illustrates the response of the nonlinear RF mixer circuit to different values of received RF signal power in the presence of additive white Gaussian noise. The noise components are represented by 20 incommensurable frequencies in a bandwidth of 8 kHz. The amplitudes of different noise
components are the sampled values of white Gaussian frequency spectrum. In this simulation, the total RF noise power density which is precisely the front-end receiver noise, is kept constant at -68 dBm (for a bandwidth of 100 kHz). The IF noise power density is -74 dBm for the same bandwidth. Therefore, the RF ring diode mixer introduces a 6 dB conversion loss for the bandpass noise. The 6 dB conversion loss is typical for the double-balanced mixers as it is between 5 dB and 7 dB in [60,63]. The SNR curve is again linear for the low received RF signal powers (below -25 dBm), i.e., the SNR increases at the same rate as the received RF signal power increases. However, around -20 dBm the curve bends down showing the large signal saturation effects in the mixer circuit.

The IF waveforms are shown for 45 dB and 80 dB channel attenuations in Fig. 6.4.3 and Fig. 6.4.5 respectively. The corresponding IF spectrums are also illustrated in Fig. 6.4.4 and Fig. 6.4.6. For the 45 dB channel attenuation, the IF signal level is 40 dB larger than the noise level as observed in Fig. 6.4.4. Therefore the IF waveform is not distorted, and the information signal can be recovered precisely. However for the 80 dB channel attenuation, the RF signal is severely attenuated. Consequently, the noise level is 5 dB smaller than the signal level at the IF band as illustrated in Fig. 6.4.6. Therefore, there is a great deal of distortion in the IF waveform as shown in Fig. 6.4.5. In this case, the information carrying signal cannot be recovered precisely, and some special demodulation techniques are needed for information extraction.
Figure 6.4.2: Signal-to-Noise Ratio Curve, SNR
Figure 6.4.3: IF Waveform for 45 dB Channel Attenuation
Figure 6.4.4: IF Spectrum for 45 dB Channel Attenuation
Figure 6.4.5: IF Waveform for 80 dB Channel Attenuation
Figure 6.4.6: IF Spectrum for 80 dB Channel Attenuation
6.5 Adjacent Channel Interference

Two adjacent channels — separated by 100 kHz — are placed side by side in the RF spectrum as illustrated in Fig. 6.5.1. Two-tone signals at different frequencies are used in each channel. The two-tone signal with 60 kHz bandwidth is carried in the main channel while the adjacent channel carries another two-tone signal with 110 kHz bandwidth. The 110 kHz bandwidth is deliberately chosen to give rise to adjacent channel interference. This excess bandwidth may also result from the inadequate filtering at the transmitter side.

The curve in Fig. 6.5.2 presents the characteristic of the IF signal power as the adjacent channel received RF signal power changes. The vertical axis which is the signal-to-adjacent channel interference ratio is by definition,

$$SAR = 10 \log \left[ \left( \sum P_{ADJ} \right) / P_{IF} \right]$$
where $P_{\text{adj}}$ is the individual adjacent channel interference component, and $P_{\text{IF}}$ is the signal power in the IF band. It should be noted that the horizontal axis in Fig. 6.5.2 is the received adjacent channel RF signal power.

The adjacent channel interference results when the two-tone signals in the adjacent channels mix with each other. For this particular example, the interference components in the IF stage are at 15 kHz, 25 kHz, and 35 kHz. Those components result from the cross product of 20 and 30 kHz two-tone signal with 45 and 55 kHz two-tone signal in the adjacent channel. The SAR measurements are taken for the various values of the received adjacent channel RF signal power. For this, the channel attenuation for the adjacent channel has been varied while it is held constant for the channel carrying the desired signal. In Fig. 6.5.2 the nonlinear characteristic is not very significant because the leakage power does not contain all of the received adjacent channel RF signal power. Therefore, it takes more power to drive the mixer circuit into the nonlinear region only by the leakage power from the adjacent channel. Nevertheless, the nonlinear response can still be observed around -25 dBm as the curve bends down slightly.

In Fig. 6.5.3 and Fig. 6.5.5, the IF waveforms are presented for 55 dB and 80 dB channel attenuation for the adjacent channel. Also, the IF spectrums for the same adjacent channel attenuation are shown in Fig. 6.5.4 and Fig. 6.5.6. The IF waveform in Fig. 6.5.3 for 55 dB adjacent channel attenuation is quite distorted due to 35 kHz and 45 kHz frequency components in Fig. 6.5.4 which are 20 dB and 3
dB smaller in magnitude than the desired signal respectively. Although the 45 kHz component is not an interference component, it comes into the picture when the IF filter has an excess bandwidth. In this particular example, the excess bandwidth is 15 kHz. Therefore, at the IF filter output the 45 kHz adjacent channel signal is present together with 15, 20, 25, 30, and 35 kHz signals. For the 80 dB adjacent channel attenuation, the IF waveform is relatively less distorted as observed in Fig. 6.5.5. This is mainly because of the 25 dB difference in magnitude between the desired signal and the interfering adjacent channel signal, as illustrated in Fig. 6.5.6.
Figure 6.5.2: Signal-to-Adjacent Channel Interference Curve, SAR
Figure 6.5.3: IF Waveform for 55 dB Channel Attenuation
Figure 6.5.4: IF Spectrum for 55 dB Channel Attenuation
Figure 6.5.5: IF Waveform for 80 dB Channel Attenuation
Figure 6.5.6: IF Spectrum for 80 dB Channel Attenuation
This simulation example is presented to demonstrate the full capability of the integrated simulation environment. There are totally 29 incommensurable (non-harmonically related) frequencies in the RF spectrum as illustrated in Fig. 6.6.1. The RF spectrum at the receiver front-end includes the main and adjacent channel with two-tone signals, and additive white Gaussian noise with 60 kHz bandwidth. The IF spectrum in Fig. 6.6.2 presents the main two-tone IF signal at 20 kHz and 30 kHz together with the adjacent channel interference components at 15 kHz, 25 kHz and 35 kHz, the intermodulation distortion component at 10 kHz and the additive white Gaussian noise components elsewhere.
Figure 6.6.2: IF Spectrum for 29 Incommensurable Input Frequencies
Chapter 7

90 kHz Experiment

7.1 Introduction

This chapter presents the actual implementation of the receiver front-end — which was simulated in Chapter 6 — and the related experimental results. The receiver front-end consists of the bandpass filter and the ring diode mixer. The frequency of operation in the experimental setup was chosen to be 90 kHz. Although this frequency does not match the frequency used in the simulation, this choice has an advantage in experimental accuracy. The measurements are more precise at low frequencies because 1) high order active filters are available, and 2) the use of opamp buffers removes the loading effects of different circuits in the receiver front-end. Also, the frequency of operation does not have a significant effect on the results since the mixer circuit is resistive. The frequency independency of mixer response is verified by the experimental results as well.

7.2 Experimental Setup

The experimental setup used to measure the harmonic response of the ring diode mixer circuit is shown in Fig. 7.2.1. Two signal sources are used: one for RF
Figure 7.2.1: Experimental Setup
signal which is an HP3312A and the other one for the local oscillator which is also an HP3312A. The local oscillator frequency is 90 kHz while the RF signal is 95 kHz. Effectively, a DSBSC (double sideband suppressed carrier) modulated 5 kHz information signal is obtained. The same modulation type was used in the simulation. The white Gaussian noise source in Fig. 7.2.1 is an ELGENCO series 624 A. The noise bandwidth is 5 MHz with flat spectral density. The white Gaussian noise is added on the RF signal by the opamp adder which is illustrated in Fig. 7.2.2. The RF signal with additive noise is then passed through the bandpass filter which is a WAVETEK model 852 dual HI/LO filter. The bandpass filter is obtained by connecting the highpass filter output to the lowpass filter input. Therefore, the bandwidth of the filter is determined by the cut-off frequencies of
### Table 7.2.1: Element Values Used in the Lowpass Filter

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>1.5 nF</td>
</tr>
<tr>
<td>$C$</td>
<td>1.5 nF</td>
</tr>
<tr>
<td>$R_1$</td>
<td>5.1 Ω</td>
</tr>
<tr>
<td>$R_2$</td>
<td>8.6 Ω</td>
</tr>
<tr>
<td>$R_3$</td>
<td>27 Ω</td>
</tr>
<tr>
<td>$R_4$</td>
<td>27 Ω</td>
</tr>
</tbody>
</table>

The lowpass and high pass filters. The center frequency of the bandpass filter is 90 kHz with a 15 kHz bandwidth. The amplitude response of the bandpass filter is shown in Fig. 7.2.3.

The mixer component of the receiver front end is a ring diode double-balanced mixer as simulated in Chapter 6. This component was selected (the model number is SRA–3) from an RF/IF Signal Processing Components Guide [63]. Right after the mixer circuit is a lowpass filter to extract the information signal which is the difference between RF signal frequency and the local oscillator frequency. The lowpass filter is a second order Butterworth active filter [64] illustrated in Fig. 7.2.4. The component values for the lowpass filter are given in Table 7.2.1. The resultant lowpass cut-off frequency is 15 kHz.
Figure 7.2.3: Amplitude Response of Bandpass Filter
The buffers used in the various parts of the setup in Fig. 7.2.1 are opamp voltage followers. These buffers are used to remove the loading effects between the filters, mixer circuit, and the measurement devices.

7.3 RF Power vs. IF Power Measurements

In these measurements, the local oscillator frequency is 90 kHz, and the RF signal frequency is 95 kHz. That is, the information signal is at 5 kHz, and it is modulated with double-sideband suppressed carrier modulation at 90 kHz. The local oscillator power is 2.7 dBm (referred to 50 Ω) in the first set of measurements. For the second set of measurements, it is 6.9 dBm (referred to 50 Ω).
Table 7.3.1: RF Power vs IF Power for 2.7 dBm LO Power

<table>
<thead>
<tr>
<th>RF Power (dBm)</th>
<th>IF Power (dBm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-44.3</td>
<td>-51.0</td>
</tr>
<tr>
<td>-37.3</td>
<td>-43.9</td>
</tr>
<tr>
<td>-31.5</td>
<td>-38.3</td>
</tr>
<tr>
<td>-23.2</td>
<td>-30.0</td>
</tr>
<tr>
<td>-17.6</td>
<td>-24.6</td>
</tr>
<tr>
<td>-12.3</td>
<td>-19.0</td>
</tr>
<tr>
<td>-6.1</td>
<td>-12.7</td>
</tr>
<tr>
<td>-1.9</td>
<td>-9.0</td>
</tr>
<tr>
<td>-0.1</td>
<td>-7.0</td>
</tr>
<tr>
<td>2.2</td>
<td>-6.1</td>
</tr>
<tr>
<td>4.0</td>
<td>-5.6</td>
</tr>
<tr>
<td>6.9</td>
<td>-5.6</td>
</tr>
<tr>
<td>8.9</td>
<td>-5.6</td>
</tr>
</tbody>
</table>

For the first set of measurements where the local oscillator power is 2.7 dBm, the conversion loss is around 5.7 dB at low RF signal powers, as given in Table. 7.3.1. However as the RF power increases, the gain compression occurs due to nonlinear saturation effects in the mixer. Consequently, the conversion loss increases to 14.3 dB for the 8.9 dBm RF signal. This phenomenon can be explained by the saturation of power delivered from the local oscillator.

For the second set of measurements, as shown in Table. 7.3.2, the conversion loss is the same as in the first set of measurements for the low RF signal powers. However in this case the gain compression decreases to 11 dB for 8.1 dBm RF signal power. This decrease is reasonable since the local oscillator power had been increased, which in turn increases the available IF power. Also, the curve for the
<table>
<thead>
<tr>
<th>RF Power (dBm)</th>
<th>IF Power (dBm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-44.1</td>
<td>-49.8</td>
</tr>
<tr>
<td>-37.9</td>
<td>-43.5</td>
</tr>
<tr>
<td>-31.7</td>
<td>-37.4</td>
</tr>
<tr>
<td>-24.8</td>
<td>-30.4</td>
</tr>
<tr>
<td>-18.6</td>
<td>-24.3</td>
</tr>
<tr>
<td>-12.5</td>
<td>-18.1</td>
</tr>
<tr>
<td>-6.7</td>
<td>-12.1</td>
</tr>
<tr>
<td>-1.5</td>
<td>-7.3</td>
</tr>
<tr>
<td>2.0</td>
<td>-4.4</td>
</tr>
<tr>
<td>4.0</td>
<td>-3.5</td>
</tr>
<tr>
<td>6.0</td>
<td>-3.2</td>
</tr>
<tr>
<td>7.0</td>
<td>-3.0</td>
</tr>
<tr>
<td>8.1</td>
<td>-2.9</td>
</tr>
</tbody>
</table>

Table 7.3.2: RF Power vs IF Power for 6.9 dBm LO Power

IF power as a function of the RF power is presented for 2.7 dBm and 6.9 dBm local oscillator powers as illustrated Fig. 7.3.1. In this curve the increase in the available IF signal power as the local oscillator power increases is observed for each RF signal power level.

This experiment was repeated for different RF signal frequencies; the relationship between the IF signal power and the RF signal power remained the same verifying the assumption about the frequency independency of the mixer circuit.

### 7.4 Noise Measurements

In the noise measurements, the local oscillator and the suppressed carrier frequencies are kept the same at 90 kHz. The bandpass filter has a bandwidth of 15
Figure 7.3.1: IF Power vs. RF Power, Solid Line for 6.9 dBm Local Oscillator Power, Dashed-dotted Line for 2.7 dBm Local Oscillator Power
kHz with the center frequency of 95 kHz. Consequently, the white Gaussian RF noise becomes bandlimited by the bandpass filter. The local oscillator power is at 6.9 dBm (referred to 50 Ω). During the experiment the total RF noise power density was constant at -60.10 dBm for 1 Hz bandwidth. The results for the IF noise power density as a function of RF signal power are tabulated in Table 7.4.1. The total IF noise power density was measured for 1 Hz bandwidth as well. The curve for the noise conversion loss as a function of RF signal power is presented in Fig. 7.4.1. As observed, the conversion loss gets larger as the RF signal power increases. The explanation is similar to that in the previous section since the available power delivered from the local oscillator gets saturated as the RF signal power increases.

<table>
<thead>
<tr>
<th>RF Signal Power (dBm)</th>
<th>IF Noise Power (dBm) (1 Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-35.7</td>
<td>-61.0</td>
</tr>
<tr>
<td>-15.7</td>
<td>-62.8</td>
</tr>
<tr>
<td>-5.0</td>
<td>-63.7</td>
</tr>
<tr>
<td>2.0</td>
<td>-65.5</td>
</tr>
<tr>
<td>4.0</td>
<td>-68.2</td>
</tr>
<tr>
<td>6.1</td>
<td>-70.2</td>
</tr>
<tr>
<td>7.6</td>
<td>-73.2</td>
</tr>
</tbody>
</table>

Table 7.4.1: Noise Measurements
Figure 7.4.1: Noise Conversion Loss vs. RF Signal Power
7.5 Conclusion and Discussion

This chapter has presented the results of a 90 kHz experiment for the RF front-end simulated in Chapter 6. Both the simulation results in Chapter 6 and the experimental results in this chapter adequately show the large signal nonlinear distortion effects in the double-balanced ring diode mixer circuit.

The simulated and experimental results for the fundamental harmonic response are illustrated in Fig. 7.5.1. Although the overall characteristics of the curves are similar, there is not an exact agreement between the numerical values. A 10 dB difference in magnitude exists between the fundamental components of the IF signals for each corresponding received RF signal. That is, the simulation results for the fundamental IF signal are 10 dB less in magnitude compared to the experimental results. This is merely because the local oscillator power in the simulation (could only be increased up to $-4$ dBm) was lower than the local oscillator power (2.7 dBm and 6.9 dBm) used in the experiment. To match the experimental results, the local oscillator power used in the simulation should have been increased. However, we could not achieve convergence in FREDA for the high local oscillator powers. The difference between the results can be explained by the fact that the local oscillator power is inversely proportional to the conversion loss. That is, as the local oscillator power increases, the conversion loss decreases. The discrepancy in Fig. 7.5.1 is introduced by the difference in the conversion loss which in turn depends on the local oscillator power.
Figure 7.5.1: IF Power vs. RF Power for the 1st Harmonic, Solid line for Experimental Result (2.7 dBm LO Power), Dashed-dotted line for Simulation Result
The convergence problem in FREDA is apparent when the mixer circuit in Fig. 4.3.3 is examined. The local oscillator signal is directly applied across the two diodes connected in series. Therefore, once the amplitude of the local oscillator signal is greater than twice the threshold voltages of the diodes, the current increases exponentially. With the increasing current flowing through the diodes, the numerical solution for the nonlinear diode mixer circuit cannot converge. As a result, the local oscillator power was kept below this high power region during the simulation in Chapter 6.

As observed in Fig. 7.5.1, the simulated results are incomplete for the received RF signal powers greater than 3 dBm. This is also due to the convergence problem in FREDA. The necessary remedies in order to solve the convergence problem are suggested in Chapter 8.
Chapter 8

Conclusion

8.1 Summary

The specific focus of this study was the development of an integrated simulation environment to aid in the design and analysis of mobile communication systems from the link level down to the circuit level. This work consists of three parts: the development of the simulation tool, the verification of its operation by way of simulation example — the mobile communication systems, and the 90 kHz receiver front-end experiment for the comparison of the simulation results.

In the first part, the integrated link level and circuit level simulation environment was developed. The simulation technique supports block diagram representations, multirate sampling, and an interactive graphical user interface. It enables the incorporation of sophisticated user models of individual blocks in a mixed time-domain and frequency-domain environment. The integrated simulation environment is sufficiently flexible to enable the merging of other circuit level simulators.

In the second portion of this work, the mobile communication system simulation example was introduced. The link and circuit level impairments were de-
scribed. The incorporation of these impairments into the functional blocks of the simulation environment was detailed. The implementation concepts for the integrated simulation environment were discussed in terms of the simulation example. Finally, the simulation results were presented and discussed.

The third part presented the 90 kHz RF front-end experiment. The simulation results were compared with the experimental ones. Along with the comparisons, the discrepancies between the results were discussed, and the reasons were presented.

8.2 Discussion

Frequency domain processing is employed in representing the filters and the band-pass signals. This choice has two major advantages: 1) Frequency domain processing permits the use of measured or independently calculated values of filter characteristics such as amplitude or phase response; and 2) The inefficiency introduced by high sampling rate is avoided in representing the bandpass signals.

It has been shown that the link level impairments can have significant effects on the nonlinear RF circuit level. A simulation environment which amalgamates the different levels of the communication system is essential to analyze such a system. The results obtained in Chapter 6 clearly show that the integrated simulation environment which simulates both the link level and the circuit level interactively was able to capture the large signal nonlinear distortion effects such as harmonic
distortion, intermodulation distortion, noise distortion and adjacent channel interference. Along with these results, the regions of linear and nonlinear operation were also predicted by the integrated simulation environment.

The purpose of the simulation example was to verify the operation of the proposed integrated simulation tool. Only by integrating a time domain link simulator with a frequency domain nonlinear RF circuit simulator is it practical to simulate the nonlinear effects.

Although the overall characteristics of the curves obtained experimentally and from the simulation are similar, there are numerical differences. As explained in Chapter 7, the discrepancy is due to the convergence problem in FREDA for the high local oscillator powers. There are two factors affecting the convergence of numerical solution technique used in FREDA: the Newton iteration scheme and the diode model. First, a smaller step size with higher precision should be used for the Newton iteration. Because a large step in diode voltage increases the current indefinitely due to the exponential characteristic. Second, the diode model should be modified to limit the available current for the voltages greater than the threshold voltage of the diode. Incorporation of these two solutions would be the best approach to solve the convergence problem in FREDA.
8.3 Suggestions for Further Study

As mentioned in the previous discussion section, the numerical solution technique and the diode model used in FREDA should be modified to achieve convergence for the high local oscillator powers applied to the ring diode mixer circuit. This will enable us to realize the simulation of more general nonlinear circuits in communication systems.

In this study, only two-tone signals with double sideband modulation were considered. Future work should investigate different modulation schemes such as PSK (Phase Shift Keying), QPSK (Quadriphase Shift Keying) and others. The problem introduced with more complex modulation schemes is the difficulties in representing these signals properly in the frequency domain. The method employed in the simulation example was the frequency domain sampling. This method is quite appropriate since the two-tone sinusoidal signals are merely finite impulses in the frequency domain. Therefore by sampling these impulses in the frequency domain, the exact signal can be recovered in the time domain. However the other complex modulation schemes are continuous functions in the frequency domain. Therefore, the sampling in the frequency domain introduces an aliasing problem in the time domain [65]. The aliasing problem can be reduced by increasing the sampling rate. However the processing of large number of samples requires an excess amount of memory and computation time. Therefore some compromise must be made between the aliasing of the signal in the time domain and the excessive computation
time required for the simulation.

Another area to be addressed is the incorporation of different level circuit simulators into the integrated simulation environment. This feature will enable the simulation of communication systems in more depth.
References


[63] Mini-circuits, "RF/IF signal processing components guide".


Appendix A

Detailed Software Structure of Integrated Simulation Environment

A.1 Introduction

This appendix presents the necessary software details in order to interface FREDA into CAPSIM. The integrated simulation environment is executed on a MicroVAX II (Digital Equipment Corporation). The operating system is UNIX. Therefore, the computer commands used hereafter are UNIX system commands.

A.2 CAPSIM Software

As mentioned in Chapter 3, CAPSIM has a modular structure. The CAPSIM software structure includes, FIFO Management (the source code), Model Library (the library stars), Functions, and Pre - processor. The modular structure is embedded into four different directories in the CAPSIM software environment. The directories are :

- SRC — FIFO management (Source code).
- STARGAZE — Pre-processor.
• **SUBS** — Reference functions.

• **STARS** — Model library.

The compilation of CAPSIM is done by the `make` command in the `SRC` directory. This command executes the `Makefile` which compiles each directory and links them to the source code. For the sake of convenience and modularity, each directory is put in a library structure by `ar -r filename.o` command in UNIX. Consequently, each directory has a library structure associated with it. After this, each library, i.e., `libsrc.a` (the source code library), `libstars.a` (the `stars` library), `libsub.a` (the functions library), is placed in the `Makefile` to be compiled and linked to the main source code.

The `functions` in `SUBS` directory are frequently used C programs by the `stars`. Therefore instead of rewriting the same programs in the `stars`, they are called from the `functions` directory. The interface of FREDA into CAPSIM was achieved through these `functions`.

### A.3 FREDA Software

The software structure of FREDA is composed of a single directory where all the source code resides. The source code is divided into different C-programs which are grouped into:

- The input functions — *FREDA_input.c*,
• The process functions — \textit{FREDA\_process.c},

  i) \textit{PROC\_association.c}  

  ii) \textit{PROC\_ckt\_anal.c}  

  iii) \textit{PROC\_aom\_model.c} — for arithmetic operator method  

  iv) \textit{PROC\_hb\_nonlinear.c} — for harmonic balance method  

• The output functions,

  i) \textit{OUT\_prt\_ckt.c}  

  ii) \textit{OUT\_prt\_soli.c}  

• The main function — \textit{main.c} which combines these three different functions.

Similarly, FREDA is compiled by the \textit{make} command in the source directory. This command executes the \textit{Makefile} by which all of the C-programming code is compiled.
A.4 Interface

As mentioned in Chapter 5, a net-list is used to communicate with FREDA. This net-list is given as a file which is read by the \texttt{freda_input()} function which is contained in \texttt{FREDA_input.c}. However, once the interface is established, it is not possible to change any parameter in the net-list which is essential in our simulation environment. Therefore, a modification program — \texttt{modify()} — was needed to transfer the spectrum obtained in the CAPSIM link level simulation to the FREDA net-list. The \texttt{modify()} function is given at the end of this section. This function essentially writes the received RF signal voltage, amplitude, and phase information in the net-list of FREDA while the simulation is running. By this, a dynamic communication link between CAPSIM and FREDA is established. The \texttt{modify()} function is linked to the FREDA source code as well.

The next step in the interface procedure is to remove the \texttt{main.c} so that FREDA can be used as a function by another main program, i.e., CAPSIM. After that, all FREDA object code except \texttt{main.c} which was removed, is placed in a library structure named as \texttt{libfreda.a}. The reason for this is the convenience in linking FREDA into CAPSIM source code. The \texttt{libfreda.a} library is placed in the CAPSIM source directory and added in the CAPSIM \textit{Makefile} as well.

Before recompiling the CAPSIM source code, the last step is to include FREDA functions — \texttt{freda_process()}, \texttt{freda_input()}, \texttt{OUT_prt_ckt()}, and \texttt{modify()} in the
sys_calls.c. The sys_calls.c program is also in the CAPSIM source directory. It simply serves to reference the external system, i.e., all of the mathematical or other functions which are used by stars should be included in this program.

Finally, to complete the interface procedure, the user should write a CAPSIM star which makes use of FREDA functions linked to the CAPSIM source code. The fredastar.s was written for this purpose. The C-programming code of this CAPSIM star is presented in Appendix C. In fredastar.s, the nonlinear RF mixer circuit net-list is read by the freda_input() function. Then, according to the received RF signal spectrum coming from link level simulation, the net-list is modified by the modify() function. The freda_process() function analyzes the nonlinear mixer circuit with the modified net-list and finally the resultant frequency spectrum is output by the OUT_prt_ckt().
Modify()

/*
 */

#include "FREDA_common.h"

void modify(name_elem, name_parm, new_val)
    char name_elem[FRE$P_BUF], name_parm[FRE$P_BUF];
    double new_val[];
{
    element *element_ptr;
    parm_list *parm_ptr;
    int ctr, nu_val;

    for(element_ptr = FRE$element_head->next_elem;
      element_ptr != NULL;
      element_ptr = element_ptr->next_elem)
    
      if(!strcmp(element_ptr->name,name_elem))
        { 
          for(parm_ptr = element_ptr->parm;
            parm_ptr != NULL;
            parm_ptr = parm_ptr->next_parm)
            
              if(!strcmp(name_parm,parm_ptr->name))
                { 
                  nu_val=parm_ptr->num_values;
                  ctr=0;
                  while(nu_val)
                    {
                      --nu_val;
                    
                    } /* nu_val */
                } /* if */
        } /* if */
    } /* for */
} /* void */

*/

mod_val.c

This routine changes the value of a specified parameter
with the command "modify". The first for loop goes through
the element structure looking for the specified element.
Once the element is found, the type of data is determined
and the change is made.

Date: October 22, 1988
Programmer: Jeyhan KARAOGUZ
*/
if(parm_ptr->value_type == FRE$K_REAL)
    parm_ptr->value[ctr]->_real = new_val[ctr];

else if(parm_ptr->value_type == FRE$K_COMPLEX)
    parm_ptr->value[ctr]->_complex[0] = new_val[ctr];

else if(parm_ptr->value_type == FRE$K_INTEGER)
    parm_ptr->value[ctr]->_int = new_val[ctr];

else if(parm_ptr->value_type == FRE$K_DOUBLE)
    parm_ptr->value[ctr]->_double = new_val[ctr];

ctr++;```
Appendix B

User's Guide

B.1 Introduction

As mentioned in Chapter 3, the integrated simulation environment consists of two different simulators, CAPSIM and FREDA. Therefore, the user should have adequate knowledge about constructing the system topologies for both simulators in realizing any simulation. In this appendix, both simulation environments are detailed as to the construction of a simulation example in general.

B.2 CAPSIM Environment

The CAPSIM simulation consists entirely of blocks and their interconnections. A block can be a *star*, in which case it refers to simulation programming, or it can be a *galaxy*, which means that the block's function is described in terms of more primitive blocks and the interconnections between them. The *stars* in CAPSIM always refer to a file that contains the programming for the *star*. Several examples of *stars* are given in Appendix C.

While each *star* refers to a simulation program segment, every *galaxy* has associated with it a specification of what blocks it contains and how they are in-
terconnected. This specification is known in CAPSIM as a topology, and stored in a topology file. The system topology consists of definitions of stars and galaxies, their corresponding parameter values, and interconnections of the individual galaxies or stars.

At the very top of the tree is a node which consists of the top-level blocks of the simulation, i.e., the galaxies. This node is called the universe.

A parameter is a variable argument to a star or a galaxy that can be set from within CAPSIM prior to running a simulation. Simply by changing the parameter values, many copies of a single star or galaxy can be used in very different ways.

Next section presents the CAPSIM user-defined topology for the mobile communication system simulation. More information as to constructing the galaxies or writing programming codes for the stars can be obtained from [66,67].

B.2.1 CAPSIM System Topology

For the mobile communication system simulation, the top level topology (i.e., the universe) is shown as the mobile communication system universe at the end of this section. The universe contains two transmitter galaxies, a receiver galaxy, a combiner star, and a channel star. Each galaxy topology is presented after the universe topology as well.

The parameters declared in the universe are used in the galaxy topologies as arg 0, arg 1, etc. The arg parameters of the galaxies are in the same order with
the declaration of parameters in the universe topology. Particularly, there are four parameters for each transmitter galaxy in the universe topology. Consequently, the same number of arg's appear in the transmitter galaxy topologies. The corresponding definitions of the parameters are given in the galaxy topologies. The remaining parameters can be found in the corresponding C-programming codes for the stars in Appendix C.
Mobile Communication System Universe

# universe.t
# This universe uses transmitter galaxies, i.e., transmitter1.t and
# transmitter2.t, transmission medium galaxy, and receiver galaxy,
# i.e., receiver.t
#
# Date: March 30, 1989
# Programmer: Jeyhan KARAOGUZ
#

param int 256
param float 128000
param int 8
param float 900000000
galaxy transmit1 transmitter1.t

param int 256
param float 128000
param int 8
param float 900100000
galaxy transmit2 transmitter2.t

#
# Combiner star
#

star combine combiner.o

#
# This is multi-path fading channel. First parameter is relative
# phase difference of specular multipath second parameter is fixed
# time delay for the path, third parameter is the attenuation in
# the direct path.
#

param float 60.0
param float 2.5
param float 30
star channel0 channel.o

param float 850000000
param float 950000000
param float 0.5
param float 100000
galaxy receiver receiver.t

connect transmit1 0 combiner 0
connect transmit2 0 combiner 1
connect combiner 0 channel 0
connect channel 0 receiver 0

Transmitter Galaxys

#
# transmitter1.t
# This galaxy simulate a two-tone transmitter system.
# First baseband signal is obtained, then transformed
# into frequency domain, and modulated with DSBSC.
# arg 0 - (int) number of points to generate
# arg 1 - (float) sampling rate
# arg 2 - (float) fft length
# arg 3 - (float) modulation carrier frequency
#
# Date: March 30, 1989
# Programmer: Jeyhan KARAOGUZ
#
param arg 0
param float 25
param arg 1
param float 20000
param float 0
star signal0 sine.o

param arg 0
param float 25
param arg 1
param float 30000
param float 0
star signal1 sine.o

param arg 2
star fft cmxfft.o

param float 0.64
param float 30000.0
param arg 3
star modulator trans.o

star adder add

connect signal0 0 adder 0
connect signal1 0 adder 1
connect adder 0 fft 0
connect fft 0 modulator 0
This galaxy simulates a two-tone transmitter system. Two-tone signal is at 45 and 55 kHz. First baseband signal is obtained, then transformed into frequency domain, and modulated with DSBSC.

Date: March 30, 1989
Programmer: Jeyhan KARAOGUZ

param arg 0
param float 40
param arg 1
param float 45000
param float 0
star signal0 sine.o

param arg 0
param float 40
param arg 1
param float 55000
param float 0
star signal1 sine.o

param arg 2
star fft cmxfft.o
param float 1.28
param float 55000
param arg 3
star modulator trans.o

star adder add
connect signal0 0 adder 0
connect signal1 0 adder 1
connect adder 0 fft 0
connect fft 0 modulator 0
Receiver Galaxy

# receiver.t
# This galaxy simulates an RF front-end
# First, the RF spectrum is filtered with a
# bandpass filter, then converted down to IF
# by a mixer circuit.
# arg 0 - (float) lower cut-off frequency
# arg 1 - (float) upper cut-off frequency
# arg 2 - (float) roll-off factor
# arg 3 - (float) transition bandwidth
#
# Date: March 30, 1989
# Programmer: Jeyhan KARADGUZ
#
param arg 0
param arg 1
param arg 2
param arg 3
star filter bandpass.o

star freda fredastar.o

param file fv.out
star readfile readfile

star wform waveform.o

star node fork

param file output1
param int 256
star plot1 plot.s

param file output2
param int 256
star plot2 plot.s

connect filter 0 node 0
connect node 0 plot1 0
connect node 1 freda 0
connect readfile 0 wform 0
connect wform 0 plot2 0
B.3 FREDA Topology

As mentioned in Chapter 3, FREDA uses a net-list to communicate with the user. The net-list consists of four sections specified as *def* (definition), *ckt* (circuit), *spectrum*, and *output*. In the *def* section, the user specifies some programming variables such as number of frequencies, type of numerical solution algorithm to be employed, maximum error in the numerical solution, etc. The *ckt* section consists of types of elements used in the circuit, their values and node connections. The frequencies of operation, i.e., the frequencies for which the circuit is to be analyzed are given in *spectrum* section. Finally in the *output* section, type of the output, i.e., current, voltage or power and the output node are specified. More information about the net-list can be found in [68].

As an example, Section B.3.1 presents the FREDA net-list for the double-balanced mixer circuit simulated in Chapter 6. The actual mixer circuit used in the net-list is as shown in Fig. B.3.1. This circuit is the equivalent of the circuit in Fig. 4.3.3 assuming a 1:2 transformer ratio. The multi-valued voltage sources in the FREDA net-list represent the RF spectrum after the bandpass filter at the RF front-end. It should be noted that the voltage sources have the actual values for the RF spectrum in the given FRADA net-list. However, prior to execution of simulation those values are reset to zero and modified by the *modify()* function after the execution of mixer (FREDA) *star* (*fredastar.s*).
Figure B.3.1: Mixer Circuit Specified in the FREDA Net-list
B.3.1 FREDA Net-list

DEF
N = 53,
CHORD = off,
_ERROR = 1.0e-12
*********

CKT
vaclo1 N1 GND f = 9e+08 v = 0.4 ;
vaclo2 GND N3 f = 9e+08 v = 0.4;

mvac1 NIF N4

mf = 8.99970e+08 8.99980e+08 9.00020e+08 9.00030e+08
9.00045e+08 9.00055e+08 9.00145e+08 9.00155e+08
8.99970e+08 8.99980e+08 9.00020e+08 9.00030e+08
9.00045e+08 9.00055e+08 9.00145e+08 9.00155e+08
8.99970e+08 8.99980e+08 9.00020e+08 9.00030e+08
9.00045e+08 9.00055e+08 9.00145e+08 9.00155e+08

mv = 0.07029 0.07029 0.07029 0.07029 0.00125
0.00125 0.00125 0.00125 0.02222 0.02222
0.02222 0.02222 0.00039 0.00039 0.00039
0.00039 0.000702 0.00702 0.00702 0.00702
0.00012 0.00012 0.00012 0.00012

mphase = -12887.5 -12888.9 -12894.7 -12896.1
-12898.3 -12899.7 -12912.6 -12914
-12827.5 -12828.9 -12834.7 -12836.1
-12838.3 -12839.7 -12852.6 -12854
-6443.76 -6444.47 -6447.34 -6448.05
-6449.13 -6449.84 -6456.29 -6457.01;

mvac2 N2 NIF

mf = 8.99970e+08 8.99980e+08 9.00020e+08 9.00030e+08
9.00045e+08 9.00055e+08 9.00145e+08 9.00155e+08
8.99970e+08 8.99980e+08 9.00020e+08 9.00030e+08
9.00045e+08 9.00055e+08 9.00145e+08 9.00155e+08
8.99970e+08 8.99980e+08 9.00020e+08 9.00030e+08
9.00045e+08 9.00055e+08 9.00145e+08 9.00155e+08

mv = 0.07029 0.07029 0.07029 0.07029 0.00125
0.00125 0.00125 0.00125 0.02222 0.02222
0.02222 0.02222 0.00039 0.00039 0.00039
0.00039 0.000702 0.00702 0.00702 0.00702
0.00012 0.00012 0.00012 0.00012
mphase = -12887.5 -12888.9 -12894.7 -12896.1
       -12898.3 -12899.7 -12912.6 -12914
       -12827.5 -12828.9 -12834.7 -12836.1
       -12838.3 -12839.7 -12852.6 -12854
       -6443.76 -6444.47 -6447.34 -6448.05
       -6449.13 -6449.84 -6456.29 -6457.01;

rif1 WIF GND r = 2000;

r1 W1 W2
    model=PNjunction
    I0=20.0e-12;

r2 W2 W3
    model=PNjunction
    I0=20.0e-12;

r3 W3 W4
    model=PNjunction
    I0=20.0e-12;

r4 W4 W1
    model=PNjunction
    I0=20.0e-12;

************

SPECTRUM

df = 0.000000e+00  1.000000e+04  1.500000e+04  2.000000e+04
       2.500000e+04  3.000000e+04  3.500000e+04  4.500000e+04
       5.500000e+04  1.450000e+05  1.550000e+05  2.100000e+05
       9.000000e+08  8.999700e+08  8.999800e+08  9.000200e+08
      9.000300e+08  9.000450e+08  9.000550e+08  9.001450e+08
      9.001550e+08  1.799970e+09  1.799980e+09  1.800020e+09
     1.800030e+09  1.800045e+09  1.800055e+09  1.800145e+09
     1.800155e+09  1.799950e+09  1.799990e+09  1.800000e+09
     1.800015e+09  1.800025e+09  1.800115e+09  1.800125e+09
     1.800019e+09  1.800035e+09  1.800135e+09  1.800100e+09
     1.800190e+09  1.800200e+09  1.800210e+09  1.800050e+09
     1.800065e+09  1.800075e+09  1.800085e+09  1.800165e+09
     1.800175e+09  1.800185e+09  2.700000e+09  3.600000e+09
     4.500000e+09  5.400000e+09

************

OUTPUT
voltage out WIF GND;
outpwr1 rif1;
B.4 Terminal Session

MicroVaxII<jk> capsim

Capsim/Blosim 3.0
Copyright (c) 1989 L. J. Faber, S.H. Ardalan, CCSP.
All rights reserved.

Topology file: universe.t
Capsim-> load universe.t
Capsim-> run
freda finished parsing on line 32

initialize DC variables

**Linear DC matrix is reduced to 6 X 6.
**Linear AC matrices are reduced to 6 X 6.

summation of errors = 4.98638

iteration number = 1 (Block Newton's Method)
summation of errors = 0.0771379

iteration number = 2 (Chord Method)
summation of errors = 0.0312852

iteration number = 3 (Chord Method)
summation of errors = 0.00639673

iteration number = 4 (Chord Method)
summation of errors = 0.00235939

iteration number = 5 (Chord Method)
summation of errors = 0.000565226

iteration number = 6 (Chord Method)
summation of errors = 0.000185516

iteration number = 7 (Chord Method)
summation of errors = 4.80989e-05

iteration number = 8 (Chord Method)
summation of errors = 1.47982e-05

iteration number = 9 (Chord Method)
summation of errors = 3.99812e-06
max. error = 4.53659e-07

*** final solutions ***
AC input:
  \text{vac} 1v(\text{peak})

Multi-value source input:
\begin{verbatim}
mvac 0.395285 0.395285 0.395285 0.395285 0.395285 0.395285 0.395285 0.395285 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.0395285 0.0395285 0.0395285 0.0395285 0.0395285 0.0395285 0.0395285 0.0395285
mnac 0.8 0.8v(\text{peak})
\end{verbatim}

(1) \text{voltage1} 4 GND;
  **freq. = 0.00000000e+00 : 
    V = -0.0166288 V
  **freq. = 1.00000000e+01 : 
    V = 0.00162394 -j3.91657e-05 V(\text{peak})
  **freq. = 2.00000000e+01 : 
    V = 0.00820049 -j3.79314e-05 V(\text{peak})
  **freq. = 3.00000000e+01 : 
    V = 0.00220601 -j9.31989e-06 V(\text{peak})
  **freq. = 4.00000000e+01 : 
    V = 0.00513463 +j1.24949e-05 V(\text{peak})
  **freq. = 5.00000000e+01 : 
    V = 0.00220601 -j9.31989e-06 V(\text{peak})
  **freq. = 6.00000000e+01 : 
    V = 0.00513463 +j1.24949e-05 V(\text{peak})
  **freq. = 7.00000000e+01 : 
    V = 0.00820049 -j3.79314e-05 V(\text{peak})
  **freq. = 8.00000000e+01 : 
    V = 9.82937e-06 -j1.80753e-07 V(peak)
  **freq. = 9.00000000e+01 : 
    V = -5.39224e-06 +j1.21541e-07 V(peak)
  **freq. = 1.00000000e+02 : 
    V = -1.42243e-05 +j3.45231e-07 V(peak)
  **freq. = 2.00000000e+02 : 
    V = 0.119932 +j0.000558674 V(peak)
  **freq. = 3.00000000e+02 : 
    V = -0.0952607 +j5.37233e-06 V(peak)
  **freq. = 4.00000000e+02 : 
    V = -0.0952607 +j5.37233e-06 V(peak)
  **freq. = 5.00000000e+02 : 
    V = 9.99997000e+00 V(peak)
  **freq. = 6.00000000e+02 : 
    V = -0.01215 -j0.0586366 V(peak)
  **freq. = 7.00000000e+02 : 
    V = -0.0122931 -j0.0586106 V(peak)
  **freq. = 8.00000000e+02 : 
    V = -0.0122931 -j0.0586106 V(peak)
  **freq. = 9.00000000e+02 : 
    V = 9.00000000e+02 V(peak)
  **freq. = 1.00000000e+03 : 
    V = -0.0130083 -j0.0584776 V(peak)
  **freq. = 1.10000000e+03 : 
    V = -0.0130083 -j0.0584776 V(peak)
  **freq. = 1.20000000e+03 : 
    V = -0.00455663 -j0.00215222 V(peak)
(2) outpwr1 rl;
  **freq. = 0.00000000e+00 :  
P =  0.00553031mw (-22.5725dbm)
**freq. = 1.00000000e+01 :  
P =  2.63872e-05mw (-45.7861dbm)
**freq. = 2.00000000e+01 :  
P =  0.000672494mw (-31.7231dbm)
**freq. = 1.00000000e+03 :  
P =  0.000263646mw (-35.7898dbm)
**freq. = 2.00000000e+03 :  
P =  4.86657e-05mw (-43.1278dbm)
**freq. = 3.00000000e+03 :  
P =  4.88776e-05mw (-43.1089dbm)
**freq. = 4.00000000e+03 :  
P =  6.60821e-05mw (-41.7992dbm)
**freq. = 5.00000000e+03 :  
P =  0.000264898mw (-35.7692dbm)
**freq. = 6.00000000e+03 :  
P =  6.65027e-05mw (-41.7716dbm)
**freq. = 8.00000000e+03 :  
P =  9.66493e-10mw (-90.148dbm)
**freq. = 9.00000000e+03 :  
P =  2.9091e-10mw (-95.3624dbm)
**freq. = 1.00000000e+04 :  
P =  2.0245e-09mw (-86.9368dbm)
**freq. = 9.00000000e+08 :  
P =  0.1436mw (-8.42845dbm)
**freq. = 9.00000010e+08 :  
P =  0.0907468mw (-10.4217dbm)
**freq. = 9.00000020e+08 :  
P =  0.0918228mw (-10.3705dbm)
**freq. = 8.99997000e+08 :  
P =  0.0358587mw (-14.454dbm)
**freq. = 8.99998000e+08 :  
P =  0.0358632mw (-14.4535dbm)
**freq. = 9.00000000e+08 :  
P =  0.0358833mw (-14.4511dbm)
**freq. = 9.00003000e+08 :  
P =  0.0358865mw (-14.4504dbm)
**freq. = 1.80000000e+09 :  
P =  4.83966e-05mw (-43.1519dbm)

*******************************************************************************
Capsim-> quit
MicroVaxII<jk>
Appendix C

Stars

/*
 */

sine()

This star generates a sinusoidal signal which is a discrete sequence of the sampled analog waveform.

Parameters:
1- (int) Number of samples in the waveform
2- (float) Magnitude of the signal
3- (float) Sampling frequency
4- (float) Frequency of the signal
5- (float) Phase of the signal

Date: November 11, 1988
Programmer: Jeyhan KARAOGUZ

*/

#include <math.h>
declare PI 3.1415926

output_buffers
  float y;
end

parameters
  int length = NUMSAMPLES;
  float magnitude = 1.0;
  float fs = 8000.0;
  float freq ;
  float phase;
end

states
  int samples_out = 0;
end

declarations
  int i;
end
main_code
   for (i=0 ; i<NOSAMPLES ; i++) {
      if (samples_out >= length )
         return(0) ;
      it_out(0) ;
      y(0) = magnitude*cos(2.0*PI*freq*samples_out/fs +phase);
      samples_out += 1;
   }
   return(0) ;
end

/*
 * trans()
 *
 * This star modulates the input signal which is in the frequency
 * domain, with a carrier frequency given as a parameter.
 *
 * Parameters ;
 *  1 - (float) Sampling frequency
 *  2 - (float) Bandwidth of the input signal
 *  3 - (float) Carrier frequency
 *
 * The sampling frequency and bandwidth of the signal are needed
 * because only the positive frequency information is available
 * at the fft output. Therefore in order to take the negative
 * frequencies into account in the modulation, one must use these
 * parameters.
 *
 * Date:  March 17, 1989
 * Programmer: Jeyhan KARAOGUZ
 *
 */

#include <math.h>
#include <stdio.h>

parameters
   float FS ;
   float BW ;
   float FO ;
end

input_buffer
   float x ;
output_buffer
  float y;
end

declarations
  int counter1 = 0,i;
  double PWR, tmp, FREQ;
  double ar[257],bw;
end

main_code
  while( it_in(0) ) {
    tmp = x(0);
    counter1++;
  
  /*
   * Place the input fft in an array. Only 256 of
   * 512 points are needed, since it is symmetric.
   */
    ar[counter1]=tmp;
    if(counter1==256) {
  
    /*
     * Here, I calculate the relative bandwidth in
     * terms of n of fft to consider the negative
     * frequencies coming from the first signal.
     */
    bw=((256.0*BW*2.0)/(100000.0*FS)+1)+2;
    for(i=bw;i>=1;i--)
  
    /*
     * With this for loop negative frequencies are
     * taken into account.
     */
    if(ar[i]>=20) {
      PWR = ar[i]/256.0;
      FREQ = (F0/10.0)-((i-1)*(FS/2.0
       *100000.0) /256.0) ;
      it_out(O);
      y(0) = FREQ ;
      it_out(O);
      y(0) = PWR ;
    }
  
  /*
   * Here the positive frequencies are handled
   */
for(i=1;i<=266;i++)
if(ar[i]>=20){
PWR = ar[i]/256.0;
FREQ = (FD/10.0)+((i-1)*(FS/2.0
*100000.0)/256.0) ;
it_out(0) ;
y(0) = FREQ ;
it_out(0) ;
y(0) = PWR ;
} ;

/*
* -31 is a flag indicating the end of the spectrum
*/

it_out(0) ;
y(0) = -31 ;

}
/ * 
* combiner()
* 
* This star concatenates the two incoming frequency spectrum, 
* and their powers. It has two inputs for different 
* spectrums. This number can be increased. 
* 
* Date: March 21, 1989 
* Programmer: KARAOGUZ, JEHANY
* */

#include <math.h>
#include <stdio.h>

parameters
end

input_buffer
  double x ;
  double y ;
end

output_buffer
  double z ;
end

declarations
  int flag=0,i=0,j;
  double xarr[20];
  double yarr[20];
end

main_code
  while( min_avail() ) {
    it_in(0);
    it_in(1);
    i++ ;

    /*
    * Here, put the incoming spectrums in different arrays
    */
    xarr[i-1] = x(0);
    if(x(0)==31)
      flag++ ;
    yarr[i-1] = y(0);
    if(y(0)==31)
      flag++ ;
/* 
* Both spectrums are placed in different arrays 
* then output as one spectrum. Flag is used 
* to make sure that both spectrums are received. 
*/

if( flag == 2 ) {

  /* 
  * i-1 is put to eliminate the flag (-31) 
  * coming from first signal. 
  */
  
  for(j=0;j<i-1;j++) {
    it_out(0);
    z(0)=xarr[j];
  };
  for(j=0;j<i;j++) {
    it_out(0);
    z(0)=yarr[j];
  };

end
Parameters:
1 - (float) Relative phase of specular path relative to direct path.
2 - (float) Delay in any path
3 - (float) Attenuation in direct path

Date: March 27, 1989
Programmer: Jeyhan KARAOGUZ

#include <math.h>
#include <stdio.h>

parameters
float phs;
float delay; /* In microseconds */
float atten; /* In dB */
end

input_buffer
double x;
end

output_buffer
double y;
end

declarations
float tmp1, tmp2, freq[40], power[40], phase[40];
int i=0, j;
end

main_code
while( min_avail() ) {
  tmp1 = pow(10.0, atten/10.0);
tmp2 = sqrt(tmp1);
for(it_in(0); x(0)!=-31; i++) {
    freq[i]=x(0);
    it_in(0);
    power[i]=x(0);
    it_in(0);}

/*
 * this is the direct path
 */

for(j=0 ; j<i ; j++) {
    power[j] = power[j]/tmp2 ;
    phase[j] = -(freq[j]*1.0e-6*delay)*180.0/3.141529 ;
}

/*
 * this is the specular multipath with
 * an additional constant phase
 */

for(j=i ; j<(2*i) ; j++) {
    freq[j] = freq[j-i];
    power[j] = power[j-i]/3.1622777 ;
    phase[j] = phs - (freq[j]*1.0e-6*delay) *180.0/3.141529 ;
}

/*
 * this is the diffuse multipath
 */

for(j=(2*i) ; j<(3*i) ; j++) {
    freq[j] = freq[j-i];
    power[j] = power[j-i]/3.1622777 ;
    phase[j] = - (freq[j]*1.0e-6*(delay/2.0)) *180.0/3.141529 ;
}

for(j=0 ; j<(3*i) ; j++) {
    it_out(0);
    y(0)=freq[j];
    it_out(0) ;
    y(0)=power[j];
    it_out(0) ;
    y(0)=phase[j];
    it_out(0);
    y(0) = -31.0 ;
}

end
bandpass()

This star is simply a bandpass filter with unity gain in its passband region, and infinity attenuation in its stop band.

Parameters:
1 - (float) Lower cutoff frequency of bandpass filter
2 - (float) Upper cutoff frequency of bandpass filter
3 - (float) Roll-off factor in transition band
4 - (float) Transition bandwidth

Date: March 30, 1989
Programmer: Jeyhan KARAOGUZ

#include <math.h>
#include <stdio.h>

parameters
float lowcutoff;
float highcutoff;
float roll_off;
float band;
end

input_buffer
double x;
end

output_buffer
double z;
end

declarations
int flag;
double freq, power, phase;
end

main_code
while( min_avail() ) {
it_in(0);
flag=x(0);
freq=x(0);
it_in(0);
power=x(0);
it_in(0);
phase=x(0);

if( (freq>=lowcutoff-band) &&
  (freq<=lowcutoff) ) {
  it_out(0);
  z(0)=freq;
  it_out(0);
  z(0)=power-(roll_off*(lowcutoff-band)/freq);
  it_out(0);
  z(0)=phase;
}

if( (freq>=lowcutoff) && (freq<=highcutoff) ) {
  it_out(0);
  z(0)=freq;
  it_out(0);
  z(0)=power;
  it_out(0);
  z(0)=phase;
}

if( (freq>=highcutoff-band) &&
  (freq<=highcutoff) ) {
  it_out(0);
  z(0)=freq;
  it_out(0);
  z(0)=power-(roll_off*(highcutoff-band)/freq);
  it_out(0);
  z(0)=phase;
}

if(flag == -31) {
  it_out(0);
  z(0)= -31;
}

end
fredastar()

* This star is written to call freda functions from capsim program.
* Date: January 18, 1989
* Programmer: Jeyhan KARAOGUZ
*
*/

#include <math.h>
#include <stdio.h>

input_buffer
double x;
end

declarations
int counter1 = 0, flag;
int counter2 = 0, temp;
float PWR, FREQ, PHASE;
char *inp_file_ptr;
double new_freq[40], new_pwr[40], new_phase[40];
double tmp1, tmp2;
end

main_code
while( it_in(0) ) {
    FREQ = x(0);
    flag = x(0);
    it_in(0);
    PWR = x(0);
    it_in(0);
    PHASE = x(0);

    if ( !counter1 ) {
        inp_file_ptr = "in1";
        freda_input(inp_file_ptr);
    };
    counter1++;
    counter2++;
    new_freq[counter2-1] = (double)FREQ+(double)8.1e+08;
    new_pwr[counter2-1] = (double)PWR;
    new_phase[counter2-1] = (double)PHASE;

    if ( flag == -31 ) {
        while( counter2 <= 41 ) {
            if ( !flag ) {
                ...
I put 41 & counter2-1 instead of 40 & counter2-1 to eliminate -31 flag in the last cell.

```
tmp1 = new_freq[0];
tmp2 = new_pwr[0];
new_pwr[counter2-1] = (double)0.0;
new_freq[counter2-1] = (double)0.0;
new_phase[counter2-1] = (double)0.0;
++counter2;
}
new_freq[0] = tmp1;
new_pwr[0] = tmp2;
```

Here modify() function is called to modify frequency, power, phase information of the received RF spectrum.

```
modify("mvac","mf",new_freq);
modify("mvac","mv",new_pwr);
modify("mvac","mphase",new_phase);
```

freda_process() function is called to analyze the mixer circuit with the modified values.

```
freda_process();
```

Finally, OUT_prt_ckt() is called to write the output in a file.

```
OUT_prt_ckt();
```

end