ZSIM: A Table-Based Z-Domain Simulator
For Delta-Sigma Modulators

Gregory T. Brauns

Center for Communications and Signal Processing
Electrical and Computer Engineering Department
North Carolina State University

CCSP-TR-88/3
January 1988
ABSTRACT

BRAUNS, GREGORY T. ZSIM: A Table-Based Z-Domain Simulator For Delta-Sigma Modulators (Under the direction of Dr. John J. Paulos, Dr. Michael B. Steer, and Dr. Sasan H. Ardalan.)

Delta-sigma modulation is one of a class of systems which use oversampling and 1-bit quantization to achieve high resolution A/D conversion at a lower rate. However, their implementation and wide-spread use is limited by the inadequacy of analytic and simulation tools. A peculiarity of the delta-sigma modulator is that it contains a mix of continuous analog and sampled digital signals, as well as strong nonlinearities, and requires thousands of simulation cycles for performance evaluation.

The purpose of this work was to obtain a fast and accurate simulator for delta-sigma modulators. Speed is achieved by modeling the circuit in block diagram form and accuracy is achieved by the use of table models developed from SPICE or CAzM device-level simulations. The use of table-based methods allows for simulation of nonidealities with increased speed over conventional device-level simulators. Post-processing routines are included in ZSIM to perform digital signal processing for performance evaluation. The table-based simulator, which uses linear interpolation routines, has been tested on idealized circuits represented by difference equations and a second-order switched-capacitor circuit represented by SPICE and CAzM table models. Results show that system performance is dependent on circuit nonlinearities captured by the table models.
ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Paulos, for his confidence and guidance over the duration of my graduate program. Without his help I could not have pulled everything together. Thanks, Dr. Steer, for all the help with the programming and spending all that time just chatting about life. And without Dr. Ardalan, the man who always knew the right thing to do, I would never have figured out all that fancy DSP stuff. Special thanks are in order for Rick Bishop, who took time from his busy schedule to help me in the heat of things to produce ZSIM table models (just in the nick of time). Thanks to Don Erdman at Duke University for debugging those little errors in the CAzM simulation program. And thanks to my friends Marcia, Keyes, Darrel, David, and Scott for helping me get through this thesis.
# TABLE OF CONTENTS

LIST OF TABLES .................................................................................................................. v
LIST OF FIGURES .............................................................................................................. vi

Chapter 1: DELTA-SIGMA MODULATION ........................................................................ 1

Chapter 2: INTRODUCTION TO TABLE-BASED SIMULATION ..................................... 6
  2.1 Motivation .................................................................................................................. 6
  2.2 CAD Concept for Sampled Data Systems ................................................................. 8
  2.3 Development of an Efficient Simulator ................................................................. 11

Chapter 3: ZSIM: A NONLINEAR Z-DOMAIN SIMULATOR ........................................... 15
  3.1 Program Structure .................................................................................................... 15
  3.2 Input Features .......................................................................................................... 19
    3.2.1 Control Commands ............................................................................................ 19
    3.2.2 Circuit Descriptions ......................................................................................... 21
    3.2.3 Filtering Analysis ............................................................................................... 22
  3.3 Tables ....................................................................................................................... 24
    3.3.1 Set-up and Generation ....................................................................................... 24
    3.3.2 Interpolation ....................................................................................................... 26
  3.4 Output Features ....................................................................................................... 28
  3.5 Memory Requirement .............................................................................................. 33
    3.5.1 Table Storage .................................................................................................... 33
    3.5.2 Integrator Storage, Example of Modularity ..................................................... 34
  3.6 Computational Speed .............................................................................................. 37

Chapter 4: BENCHMARK SIMULATIONS ......................................................................... 40
  4.1 High-level Benchmark: Ideal Table vs. Difference Equation ................................. 41
    4.1.1 Voice Band Circuit ............................................................................................ 41
    4.1.2 Simulation Results ............................................................................................ 42
      4.1.2.1 Baseband Spectrum .................................................................................. 47
      4.1.2.2 Signal-to-Distortion Ratio ......................................................................... 49
      4.1.2.3 Interpolation Accuracy ............................................................................... 52
  4.2 Low-Level Benchmark: Device-Level vs. Difference Equation .............................. 53
    4.2.1 ISDN Circuit ...................................................................................................... 53
    4.2.2 Table Generation .............................................................................................. 55
    4.2.3 System Performance ......................................................................................... 65

Chapter 5: FUTURE INVESTIGATIONS ............................................................................. 69
  5.1 Tables and Interpolation ......................................................................................... 69
  5.2 ZSIM Additions ....................................................................................................... 71

LIST OF REFERENCES ........................................................................................................ 73

APPENDICES ..................................................................................................................... 76
  A: ZSIM User's Manual ................................................................................................. 76
  B: SPICE Simulation Example ...................................................................................... 102
  C: Linear Interpolation Formula Derivation ............................................................... 113
LIST OF TABLES

Table 3.1: Summary of control commands ................................................................. 20
Table 3.2: Run time comparison using tables, difference equations, and SPICE ......... 38
Table 4.1: SDR summary for ideal table and difference equation simulations .......... 49
Table 4.2: Summary of integrator simulation for cycle #249 .................................... 52
Table 4.3: SDR summary for ideal table, SPICE table, and CAzM table simulations ................................................................. 65
LIST OF FIGURES

Figure 1.1: Block diagram of a first-order delta-sigma modulator ............................................. 2
Figure 1.2: Sample set of signals for a first-order delta-sigma modulator .................................. 3
Figure 1.3: Block diagram of a first-order delta-sigma demodulator .......................................... 4
Figure 2.1: Flowchart of a CAD tool for delta-sigma modulators ............................................... 9
Figure 2.2: Second-order delta-sigma modulator ..................................................................... 12
Figure 2.3: Second-order delta-sigma modulator with fictitious sample-and-holds ....................... 13
Figure 2.4: Second-order z-domain delta-sigma modulator ....................................................... 13
Figure 3.1: Simulation flowchart for ZSIM .................................................................................. 16
Figure 3.2: Details of the Table Generator Module .................................................................... 17
Figure 3.3: Post-Processor Module flowchart ......................................................................... 18
Figure 3.4: A ZSIM integrator ................................................................................................. 25
Figure 3.5: Format of a table for N=3 and M=2 ....................................................................... 26
Figure 3.6: A planar view of table grid with interpolation points .............................................. 27
Figure 3.7: A sample collection of bins around the signal ......................................................... 31
Figure 4.1: Ideal delta-sigma modulator system for CODEC application .................................... 41
Figure 4.2: CODEC integrator #1 ideal table .......................................................................... 43
Figure 4.3: CODEC integrator #2 ideal table .......................................................................... 44
Figure 4.4: Graph representation of integrator #1 .................................................................... 45
Figure 4.5: Graph representation of integrator #2 .................................................................... 46
Figure 4.6: CODEC baseband spectrum for input = - 15 dB ..................................................... 47
Figure 4.7: CODEC baseband spectrum for input = 0 dB ......................................................... 48
Figure 4.8: SDR versus input amplitude for CODEC delta-sigma modulator ......................... 50
Figure 4.9: SDR differences (difference equation - table) versus input amplitude ................... 51
Figure 4.10: Second-order delta-sigma switched-capacitor circuit ............................................ 54
Figure 4.11: ISDN integrator #1 ideal table ............................................................................. 56
Figure 4.12: ISDN integrator #2 ideal table ............................................................................. 57
Figure 4.13: ISDN integrator #1 table generated from SPICE .................................................. 58
Figure 4.14: ISDN integrator #1 table generated from CAzM ................................................... 61
Figure 4.15: ISDN integrator #2 table generated from SPICE .................................................. 62
Figure 4.16: ISDN integrator #2 table generated from CAzM ................................................... 63
Figure 4.17: Circuit errors exhibited by SPICE and CAzM tables ......................................... 64
Figure 4.18: SDR versus input amplitude for ISDN delta-sigma modulator ............................. 66
Figure 4.19: ISDN baseband spectrum for input = - 10 dB ....................................................... 67
1. DELTA-SIGMA MODULATION

Delta-sigma modulation [1, 2] is one of a class of systems which use oversampling and 1-bit quantization to achieve high-resolution A/D conversion at a lower rate. Oversampling is attractive in that precise analog anti-alias filtering can be omitted. Instead, digital FIR lowpass filters, which are relatively insensitive to coefficient roundoff, are used after the modulator to perform decimation and anti-alias filtering. Decimation is required to achieve conventional Pulse Code Modulation (PCM) signals by reducing the sampling rate of the 1-bit data stream generated by the modulator. Another attraction is that delta-sigma modulators can be implemented with few precision circuits and precise component tolerances are not needed [3, 4, 5, 6]. Delta-sigma modulators can be easily implemented in digital MOS IC technologies [3] through the use of switched-capacitor circuits. This approach has recently gained increased attention [3, 7, 8, 9]. Digital MOS technology also lends itself to the implementation of complex decimating digital filters.

Figure 1.1 shows a block diagram of a first-order delta-sigma modulator. The circuit topology is that of a nonlinear, sampled-data, closed loop control system which is used as a signal tracking device. The system accepts an analog signal, $x(t)$, and encodes it as a digital pulse stream, $p(t)$, for transmission to a demodula-
tor. An example set of these signals is presented in Figure 1.2. The feedback works to minimize the error signal, $e(t)$, given by

$$e(t) = \int [x(t) - p(t)] dt.$$ (1.1)

The error signal is quantized by a comparator which samples and holds the binary code for one clock cycle in an attempt to track $x(t)$ with $p(t)$. Therefore, the output is driven so as to match the signal directly, at least in an integrated error sense. The maximum encodable input signal is equal to the amplitude of the output pulses [2]. Therefore, overloading of the system is independent of the signal frequency and the system has no significant problems in tracking the signal.

One advantage of delta-sigma modulation is that the corresponding demodulator is simple and does not require analog circuitry, as shown in Figure 1.3. Demodulation can be achieved by simply reshaping the received pulses and passing
Figure 1.2: Sample set of signals for a first-order delta-sigma modulator
them through a lowpass filter. The lowpass filter reduces the effects of quantization by removing the out-of-band noise and the signal images at multiples of the clock frequency.

Although delta-sigma modulation is conceptually simple, the system is difficult to analyze. The nature of the modulator's structure prohibits simple analysis - the quantizer is a nonlinear device, the sample and hold function causes the output pulses to be dependent on time and amplitude, and the feedback loop introduces stability problems. Also, if a random input is applied to the system, evaluation of the noise content in the output signal is difficult. Consequently, the implementation and wide-spread use of delta-sigma modulators is partly limited by the inadequacy of analytic and simulation tools. Only numerical simulations can provide the required confidence in design, final optimization of system performance, and investigation of novel circuit topologies. It was the intention of this work to design a simulator for delta-sigma modulators which includes the effect of
circuit nonlinearities in a fast and efficient manner.

The following chapters will introduce and demonstrate a novel simulator for delta-sigma modulators. ZSIM (a nonlinear Z-domain Simulator), uses table-based methods in a block-level configuration so that fast and accurate simulations can be obtained. Chapter 2 discusses the need for a new circuit simulator of this type and introduces the concept of table-based simulation for delta-sigma modulators. Chapter 3 goes into the details of the program structure, describes the table data structure, and presents computational speed comparisons of the table-based simulator versus conventional device-level simulators. Some sample results are presented in Chapter 4 which demonstrate the validity of the table method. Also presented are simulated results of a complete circuit intended for use in ISDN. Finally, Chapter 5 will present some conclusions and recommendations for future work.
CHAPTER 2

2. INTRODUCTION TO TABLE-BASED SIMULATION

2.1. MOTIVATION

Delta-sigma modulators have been successfully used in voice-band CODECS [6] and for the U-interface of an Integrated Services Digital Network (ISDN), where monolithic high resolution A/D conversion is required for echo cancellation [5]. However, the implementation and wide-spread use of delta-sigma modulators is partly limited by the inadequacy of analytic tools and simulation tools. A peculiarity of the delta-sigma modulator is that it contains a mix of continuous analog and sampled digital signals, as well as strong nonlinearities, which complicate the development of analytic and numerical design aids.

Recent developments in analytic techniques for delta-sigma modulators [10, 11, 12, 13, 14, 15] have increased the understanding of the signal-to-distortion ratio, quantization noise spectra and stability of these circuits. Unfortunately, these techniques do not include circuit nonidealities. Only numerical simulations can provide the required confidence in design, final optimization of system performance, and investigation of novel circuit topologies. Until now numerical simulation of delta-sigma modulators has been restricted to the use of time-consuming
circuit-level simulators (e.g. SPICE [16]), and numerically efficient difference equation simulators, which cannot capture circuit phenomena [17]. The speed of simulation is of overwhelming importance as the circuit must be simulated for a large number of clock cycles, often tens of thousands. While being rapid, difference equation simulations cannot easily include component effects such as slew-rate limiting, noise, hysteresis, clock feed-through, and many types of nonlinearity associated with some types of comparators.

The problem of simulation speed has been solved in various ways. For instance, utilizing a computer’s full capabilities by writing machine dependent code can improve simulation speed up to 20 times [18]. Bypassing a simulation cycle, if little change occurs in the terminal voltages of a device, is another method of increasing simulation speed. Also, the addition of theoretical circuit predictions aids in developing more efficient algorithms. Table look-up methods have also been used recently in the speed-up of computer-aided analysis, especially in the area of MOSFET modeling [19, 20, 21]

Table look-up methods offer several advantages over other analytical and numerical methods. Tables allow nonlinear as well as linear system equations to be modeled and solved. Also, computation time is saved for every simulation once a table is created and stored in memory. Tables allow for the re-use of data rather than solving the same system of equations over and over for each time point or for each different simulation run.
Table look-up methods are also economically practical. Today's low cost memory units allow for storage of virtually any number of tables of almost any size. Even if secondary storage devices are used rather than the computer semiconductor memory, the milliseconds of access time are still competitive with the minutes of CPU time needed for other simulation methods.

The advantages of speed and accuracy with table-based simulation has appealed to many researchers [22]. Tables are being used for MOSFETs in transistor-level circuit simulation [19, 20, 21, 23, 24, 25, 26, 27] where more than half of the total simulation time is often required for evaluation of the MOSFET models. Microwave circuits are also being simulated with tables [28, 29]. The goal of this work is to use tables to represent mixed-mode (analog and digital) feedback sampled-data systems such as delta-sigma modulators.

2.2. CAD CONCEPT FOR SAMPLED DATA SYSTEMS

The concept of integrating analytical tools, difference equation simulation and table-based simulation, with appropriate postprocessing analysis is illustrated in Figure 2.1. Using the analytical tools developed in [10, 11, 12, 13, 14], the parameters of a candidate delta-sigma modulator are quickly derived for a given desired system performance such as signal-to-noise or signal-to-distortion ratio. These parameters include the oversampling ratio, the order of the modulator (first, second, or third order), the gains for stable operation, the required length of the decimation filter, the decimation weighting (uniform, triangular, or parabolic), and
As an example, Ardalan and Paulos [14] have expressed the signal-to-noise ratio of a sinusoidally excited modulator as a function of input amplitude. In addition, the variance at various nodes in the circuit both for the signal component and the noise components are calculated. It is shown that as the input signal ampli-
tude increases these variances rapidly increase leading to saturation in actual cir-
cuits. Analysis of these variances, in conjunction with algebraic expressions of the
specified performance requirements, can be used to determine the system paramet-
ters of the modulator such as the oversampling ratio, the order of the modulator,
and the integrator gains. Using the expressions for the frequency response of the
decimation filters [12], it is then possible to compute the aliased noise in the
baseband and subsequent reduction in signal-to-noise-plus-distortion ratio
\( \frac{S}{N+THD} \) for different weighting and tap lengths.

After the system parameters have been determined using the analytical tools,
difference equation simulations are combined with the actual decimation and
baseband filters to verify the performance of the modulator ignoring potential cir-
cuit limitations. At this stage, circuit-level implementations are considered, and
tables are generated, using circuit-level simulators such as SPICE, for the subsys-
tems that make up the modulator. Tables are generated for the subsystems that
make up the modulator. Using these tables, ZSIM captures circuit-level nonideali-
ties but still achieves rapid simulation at the subsystem level.

During each stage of the design and simulation process the designer can
iterate between the simulation systems and analytical tools to determine the circuit
which is most appropriate in terms of complexity, technology, and other con-
straints.
2.3. DEVELOPMENT OF AN EFFICIENT SIMULATOR

The simulation of delta-sigma circuits is complicated by oversampling and the presence of both analog and digital signals. This results in time-domain simulations being prohibitively time consuming. However, delta-sigma modulators are sampled data systems, so it is possible to model the performance of individual subsystems of the modulator at the sampling intervals. Continuous-time information, such as the circuit waveform between sampling intervals and the circuit state at internal subsystem nodes, are not required for accurate system-level simulation. Thus it is possible to use a z-domain description of the system. The utility of difference equation simulators is that they operate in the linear z-domain and so computations are kept to a minimum. Often, however, it is not possible to develop sufficiently accurate difference equations for practical delta-sigma circuits because of complex dependencies on nonlinearity, hysteresis, clock feed-through, slew-rate limiting, and finite gain-bandwidth-product of the subsystems. ZSIM, using table methods, is a natural extension of difference equation simulation and enables these effects to be modeled using a multi-dimensional table.

The development of ZSIM is analogous to that of the difference equation method. A second-order delta-sigma modulator is shown in Figure 2.2. With the addition of fictitious sample-and-holds, as in Figure 2.3, a delta-sigma modulator can be represented in the z-domain as in Figure 2.4. This representation enables the subsystems to be considered individually (although the input and output loading of the subcircuits must remain unchanged). Since practical monolithic imple-
Implementations of delta-sigma modulators use sampled data circuits and/or switched capacitor circuits, the addition of fictitious sample and holds, if placed appropriately, has no affect on circuit performance.

Consider the first integrator block of Figure 2.4. The output of this subsystem can be linearly modeled by the product of its input and its transfer function, as given by

\[ y[k] = \alpha_1 H_1[z] (x[k] - p[k]) \]  \hspace{1cm} (2.1)

where \( y[k] \) is the integrator output and the transfer function is given by

\[ H_1[z] = \frac{z}{z-1} . \]  \hspace{1cm} (2.2)

Since computer simulation uses discrete time steps, the function occurs in the form of sequences. Converting the combination of Equation 2.1 and Equation 2.2 to a sequence results in
\[ y[k] = \alpha_1 \sum_{n=-\infty}^{k} (x[n] - p[n]) \] (2.3)

which can be rewritten as
\[ y[k] = \alpha_1 \sum_{n=-\infty}^{k-1} (x[n] - p[n]) + \alpha_1 (x[k] - p[k]) \]  

or

\[ y[k] = y[k-1] + \alpha_1 (x[k] - p[k]) \]  

Equation 2.5 represents the difference equation implementation of a simple integrator used in most simulators. The limitation is that this function is linear. Physical circuits may contain gain errors, saturation of the output, and comparator voltage mismatches, which have no a linear relationship with the output. More generally, \( x[k], y[k-1], \) and \( p[k] \) are independent variables for \( y[k] \) with an unknown nonlinear relationship which can be described by

\[ y[k] = T(x[k], y[k-1], p[k]) \]  

where \( T \) is a table look-up function. Subsystems in ZSIM are described by Equation 2.6 where the table values may contain either linear or nonlinear data.
3. ZSIM: A NONLINEAR Z-DOMAIN SIMULATOR

ZSIM is a nonlinear Z-domain SIMulator designed for simulation of sampled data systems. ZSIM integrates analytic tools, a difference equation simulator, a novel table-based nonlinear z-domain simulator, and digital signal postprocessing into a workstation environment. The primary goal of ZSIM is development of an accurate and fast simulator for delta-sigma modulators.

ZSIM has a user-friendly input format but lacks a totally stand alone topology specification. Up to third order DSM’s can be simulated using difference equations, and first and second order DSM’s can be simulated using table methods. Program code is written in ANSI standard FORTRAN 77 and operation has been verified on a DEC MICROVAX running either the Ultrix 1.2 or MICROVMS operating system.

3.1. PROGRAM STRUCTURE

ZSIM is a completely modular program consisting of four major subdivisions - an Input Module, a Table Generator Module, a Simulator Module, and a Post-Processor Module. A high level flowchart of the program is shown in Figure 3.1. The capability exists to simulate variations in the circuit (either in topology or cir-
circuit parameters) or variations of external parameters (such as clock rate or input signal) in one computational run, as might be needed for a comparative analysis.

The Input Module's high level input capability allows for input to be taken from the keyboard and/or from a disk file. Standard English commands are used to diminish the need for a specific program environment language.

The Table Generator Module develops the tables that describe the input-output characteristics of each subsystem. The functionality of the Table Generator Module is depicted in Figure 3.2. Basically, a decision is made whether to use a difference equation representation of the circuit or a table representation of the circuit, as determined by the user. For table representations, a table is read from memory or created. The table may be an ideal table where data points are generated by a direct difference equation or a table constructed with data obtained from circuit-level simulations. Ideal table generation is performed in a stand alone routine not linked to ZSIM's work environment. Circuit-level table generation is
presently a manual chore which has not as yet been automated. Automatic table generation would involve running a circuit-level simulator, such as SPICE, for one clock cycle, extracting pertinent data, storing the data in a specific memory location, and repeating the process until each position of the table contains simulated data.

The Simulator Module performs simulations of the sampled data circuit using table descriptions of individual subsystems or using difference equations for rapid circuit investigations. Since topology specifications are currently ignored (except for internal storage allocation requirements), the user is only free to choose the
order of the system. The topology is therefore fixed according to the order of the modulator. The order of simulation for each subsystem at each clock cycle is as follows: the input signal generator, the integrators in the order of signal propagation, and the comparator. The voltages at the input and output nodes of each subsystem are stored in memory for processing during the next clock cycle.

The Post-Processor Module implements decimation and baseband filtering and calculates system performance characteristics (in the frequency domain) in a high-level fashion depicted in Figure 3.3. Flexibility is provided in that any element of this module can be bypassed. For instance, an FFT may be performed on the modulator bitstream before decimation occurs or baseband filtering may be excluded completely. It is even possible to use parallel structures of these elements. For example, two different filters may be defined to operate on the output of the decimator, with a different FFT analysis specified for each filter.

![Figure 3.3: Post-Processor Module flowchart](image)
A source file flowchart of the program can be found in the User’s Manual (Appendix A), with each routine separated into its corresponding module (Input, Table Generator, Simulator, and Post-Processor). Note that all modules stem from the file ZCMD; ZCMD handles all commands in a modular fashion. Additional source code can be integrated into ZSIM by simply adding a command recognition statement in ZCMD. Adding a new pre-processing or post-processing subsystem, such as a digital filter, would be just one example of a modular addition. If a new subsystem, such as a multiplier, is desired for inclusion in the Simulator Module, modification of the simulation command routine ZANA is also required to add a new fixed topology option. The ZSIM source code is published in [30], which may aid in the module addition process.

3.2. INPUT FEATURES

3.2.1. Control Commands

Program organization and execution is controlled through the Input Module. Two types of control commands exist: disk file input/output commands and circuit control commands. The control commands are summarized in Table 3.1. Disk file input/output commands are necessary to handle differences in interactive and file-read executions of ZSIM. Flexibility is enhanced in that program overhead may be included in the output, if so desired, with commands like ECHO ON, TITLE, EOF, and STOP. On the other hand, circuit control commands are definitions for
Table 3.1: Summary of control commands

<table>
<thead>
<tr>
<th>Disk file input/output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>open file for input</td>
</tr>
<tr>
<td>write</td>
<td>open file for output</td>
</tr>
<tr>
<td>echo on/off</td>
<td>echo input lines</td>
</tr>
<tr>
<td>eof</td>
<td>close input data file</td>
</tr>
<tr>
<td>end</td>
<td>temporarily close input data file</td>
</tr>
<tr>
<td>title</td>
<td>write line to output file</td>
</tr>
<tr>
<td>prompt</td>
<td>write line to terminal</td>
</tr>
<tr>
<td>stop</td>
<td>end ZSIM session</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circuit control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>circuit</td>
<td>define order of circuit</td>
</tr>
<tr>
<td>environment</td>
<td>define external circuit parameters</td>
</tr>
<tr>
<td>init</td>
<td>initialize quantities</td>
</tr>
<tr>
<td>clear</td>
<td>set all parameters to default</td>
</tr>
<tr>
<td>noise</td>
<td>define Gaussian white noise at circuit nodes</td>
</tr>
<tr>
<td>dump</td>
<td>store node values in a file</td>
</tr>
</tbody>
</table>

circuit and simulation parameters or specifications (CIRCUIT, ENVIRONMENT, INIT). ENVIRONMENT sets parameters such as sampling frequency and the number of clock cycles to execute. CLEAR is a user-friendly command which resets all circuit parameters and subsystem definitions to default so that more than one circuit can be simulated during one ZSIM session. NOISE is a Gaussian white noise generator available to test system response for noise present at various circuit nodes. Use of the generator has not been completely tested with respect to peak noise values, i.e., the noise probability density function is not scaled to fit variable power supply ranges. The DUMP command stores a record of node values, which
may be FFT bins or a time sequence of voltages.

3.2.2. Circuit Descriptions

Delta-sigma modulator systems are defined by an input generator (GEN), an integrator (SCINT), and a quantizer (QUANT). These subsystems are configured to form the block delta-sigma modulator structure as previously depicted in Figure 2.4. With each subsystem definition, node numbers must be specified (although circuit topology is hardwired) for memory allocation purposes.

Input signal types currently available include dc, ramp, sinewave, and step function. It is up to the user to insure that input signals do not exceed the circuit power supplies. All parameters, such as amplitude, frequency, phase, delay, and dc offset, are variables set by the GEN definition statement.

Since MOS IC technology is generally used for delta-sigma modulators [17], the most common topology for the integrators will be a switched-capacitor integrator. Therefore, the switched-capacitor integrator (SCINT) command is used for ZSIM's integrator. For the purposes of simulation, the output of a SCINT is a function of its inputs and its previous output and is described by a table. Each SCINT command should be followed by a TABLE command when table simulation is required. A discussion of tables will be presented in section 3.3.

Presently, the quantizer (QUANT), or comparator, is simulated as an ideal circuit, i.e., tables are not used to describe the input-output transfer function of a comparator. This prevents simulation of effects such as hysteresis. Output is
binary and is coded as +1 or -1. An offset may be specified for the comparator's switching threshold.

Before a time consuming table generation is performed for a switch-capacitor integrator, a difference equation model can be used. Simulation of the circuit with this model can be used to validate the choice of integrator gains. To define a difference equation model for the integrators, one EQUATION command replaces all SCINT, TABLE, and QUANT commands. Circuit topology is hardwired in equation mode; nodes are internally assigned numbers depending on the order of the system. Parameters of the model include output saturation voltage, gain, switched reference voltage, and finite-gain-bandwidth. See the User's Manual in Appendix A for command usage.

3.2.3. Filtering Analysis

Filtering of the modulator output is included since in most applications the oversampled data must be decimated to a workable frequency. Both FIR decimating filters and IIR baseband filters can be specified. FFT and other signal analysis are available to measure SNR and SDR of the complete system, including decimation and baseband filtering. The DECIMATE and SDR commands include the specification of the circuit nodes which define a subsystem-like structure as described earlier. This allows for flexibility of connections.

The DECIMATE command defines the type of FIR decimating filter and may include an IIR baseband filter. The FIR decimation filter implemented is simply a
canonical tapped-delay line. Command inputs include a decimating factor (INTDEC), the number of tapped delays (TAPS), and a windowing function (WINDOW = uniform, triangular, or parabolic) which describes the tap weighting scheme. The weighting scheme is congruent to the impulse response of the filter. Also, the decimation is performed by saving one sample out of every INTDEC samples. The IIR baseband filter inputs include a decimating factor (BBANDDEC), a special filter file (FILE), and the type of filter (TYPE = none, direct form, normalized lattice form, or cascade form). The special filter file is a file that contains IIR filter coefficients. Currently, these coefficients must be generated by the user through a separate routine that designs these IIR filters and produces the coefficient file (this routine is available from CCSP). Version ZSIM0a1 includes a file, VB.CAS, which specifies a cascade filter designed for decimation to voiceband. VB.CAS has a 3.4 kHz rolloff for an 8 kHz sampling frequency.

The SDR command defines the digital signal post-processing. An SDR calculation, which is based on the fast Fourier transform (FFT), may be performed on any sequence of data as long as the number of bins in the sequence is a power of 2. This criterion is necessary for the FFT simulation. Thus, an FFT may be calculated for the modulator bitstream or the decimated and filtered output. If decimation is performed, the parameters in SDR must agree with those of DECIMATE. This allows the SDR and DECIMATE routines to operate completely independent of each other.

† Center for Communications and Signal Processing, North Carolina State University
For an SDR calculation, the number of points (NFFT) needed for the FFT must be specified. There is an NSKIP parameter that allows initial signal bins, which may represent transient circuit performance, to be ignored. Other parameters are sampling frequency (FS), passband frequency (FB), signal frequency (SIGNAL), and FFT window type (FFTWINDOW = uniform, Hamming, or Blackman-Harris). Parameters that correspond to the decimation routine are the number of FIR tapped delays (TAPS) and the decimation window (DECWINDOW). These two parameters are required to adjust signal frequencies and amplitudes which may be degraded by the filter. This problem is discussed later in the "Output Features" section. Specifying DECWINDOW=ignore will let SDR assume that the filter has no degradation problems.

3.3. TABLES

3.3.1. Set-up and Generation

The representation of circuit subsystems by tables allows discretization of input/output data for linear and nonlinear regions of operation. Thus, all timedomain circuit information is lost. Currently, ZSIM handles only switched-capacitor integrator tables.

A block diagram of an integrator implemented in ZSIM is shown in Figure 3.4 where the output is modeled by the following difference equation:

$$y[k] = y[k-1] + \alpha (x[k] - p[k]).$$  (3.1)
It is observed that the output of the integrator depends on three variables, denoted $T(x,y,p)$: the input signal $x[k]$ at the $k^{th}$ clock cycle, the integrator output $y[k-1]$ at the previous cycle, and the current comparator output $p[k]$. Since $p[k]$ is the output of a binary comparator, it can assume only two values. The signals $x[k]$ and $y[k-1]$, however, may range between each power supply rail, although $y[k]$ is generally limited by op-amp saturation and may not completely reach the power supply voltages. Thus, if $N$ discrete values are selected for $x[k]$ and $M$ discrete values for $y[k-1]$, a $2 \times N \times M$ table will describe the operation of the integrator. Therefore, the table is essentially divided into two two-dimensional tables, each representing one of the two possible values for $p[k]$.

A complete table contains definitions of the discretized inputs and the corresponding $T(x,y,p)$ values. The inputs are in monotonic order. An example

\[ \]
showing the table format is given in Figure 3.5 for $N=3$ and $M=2$. Variable $X1$ is the input $x[k]$, variable $X2$ is the previous integrator output $y[k-1]$, and variable $X3$ is the comparator output $p[k]$. The arrangement of $T(x,y,p)$ values is equivalent to FORTRAN 77 memory allocation for array elements. Variable $X3$ varies the slowest while variable $X1$ varies the fastest. Each column corresponds to constant $X1$. Each row corresponds to constant $X2$ and $X3$.

3.3.2. Interpolation

Linear table interpolation is used in ZSIM and is considered to be adequate for delta-sigma modulators where the system components should be designed to be approximately linear. Linear interpolation introduces discontinuities only in the derivatives of table values. This section describes the method of linear

```
* comment lines
* X1 = x[k] , X2 = y[k-1] , X3 = p[k]

X1 = x1(1), x1(2), x1(3)
X2 = x2(1), x2(2)
X3 = x3(1), x3(2)

begin y[k] = T(x,y,p)
    T(1,1,1) T(2,1,1) T(3,1,1)
    T(1,2,1) T(2,2,1) T(3,2,1)
    T(1,1,2) T(2,1,2) T(3,1,2)
    T(1,2,2) T(2,2,2) T(3,2,2)

done
```

Figure 3.5: Format of a table for $N=3$ and $M=2$
interpolation for a multi-dimensional space.

Since the comparator output value is a binary discrete word, no interpolation is needed between the \( T(x, y, -1) \) and \( T(x, y, +1) \) planes. Therefore, an integrator output becomes a function of two variables, \( T(x, y) \), and defines a plane as shown in Figure 3.6. Generally, the input values \( (X1) \) and the previous output values \( (X2) \) will not coincide with the discrete graph points, so interpolation is necessary.

Four \( y[k] \)'s are selected from the table: \( T(x_1, y_1) \), \( T(x_2, y_1) \), \( T(x_1, y_2) \), \( T(x_2, y_2) \). Using the actual \( x[k] \) value, linear interpolation between \( T(x_1, y_1) \) and \( T(x_2, y_1) \) produces \( T(y_1) \), and interpolation between \( T(x_1, y_2) \) and \( T(x_2, y_2) \) yields \( T(y_2) \). Then linear interpolation of \( T(y_1) \) and \( T(y_2) \) given \( y[k-1] \) is performed to

---

Figure 3.6: A planar view of table grid with interpolation points
obtain the desired approximation of $y[k]$. The overall formula is derived in Appendix C. A small error will occur at the output in nonlinear regions of the integrator operation due to the use of linear interpolation. With more elaborate interpolation routines and nonuniform interval tables, a more accurate model of the linear and nonlinear regions could be obtained with fewer table points.

3.4. OUTPUT FEATURES

The ZSIM output is chosen to allow the user to evaluate both system performance and simulator performance. Each output feature is tailored to aid in the design process of the modulator. Thus output can be divided into three categories as follows:

- program overhead data,
- subsystem (table) performance evaluation, and
- A/D encoding performance evaluation.

where the program overhead data is primarily an echo of the input data.

The subsystem evaluation data summarizes the performance of a circuit subsystem which is usually described by a table model. Information on the use of the table data is provided to aid in better table design, and several measures of the voltage levels at the subsystem nodes are provided to aid in the evaluation of the subsystems in the context of the overall system. Specifically, the HISTOGRAM command outputs the history of a table in histogram form. Each data point of a table’s output array is associated with a counter, which is updated each time that
point is used by the interpolation routine. (since the interpolation formula reads four (4) data points at a time, the sum of all table histogram amplitudes will be four times the number of clock cycles simulated). The histogram is displayed in chart form (like the table itself) for each table subsystem of the last simulation. The table designer can see which table points are used and which ones are not and, therefore, create a more efficient table or omit some table simulations. It is suggested that an ideal table be simulated for this purpose before the time consuming circuit-level simulations are performed. Along with histogram output is the subsystem external node voltage list, which is flagged by the PRINT PARAMETERS command. The minimum and maximum voltage levels of each node (specified as table input nodes) are recorded. This feature enables the designer to see if input overloading or output saturation has occurred.

The last category of output features is the A/D performance evaluation. Evaluation of the delta-sigma modulator entails post-processing by decimation and baseband filtering and Fourier analysis. Decimation and baseband filtering was described earlier in Input Features: Post-Processing. It is the intent here to describe the analysis of the filter output.

The filter output analysis is performed in the frequency domain. One result of these computations is signal-to-distortion ratio (SDR), where distortion is defined to be all energy not associated with the signal energy. SDR is important in that it shows overall system performance. However, distortion may be divided into two categories: white noise and harmonic noise. A signal-to-noise ratio (SNR) and a
signal-to-harmonic noise ratio (STHD) is also calculated. SNR is especially important in voiceband applications, since the human ear can hear low levels of white noise in conjunction with the signal. Switching problems and saturation problems in the delta-sigma modulator are often identified by STHD performance. Calculation of these three ratios (SDR, SNR, STHD) requires special care since no clear-cut method exists to separate harmonic and nonharmonic noise.

After a signal is modulated and filtered, it may be necessary to delete initial samples of the signal to remove the initial transient response of the modulator or filter, leaving only the steady-state response. After doing so, the signal is passed through a Fast Fourier Transform (FFT) routine to convert the time-domain signal to a frequency-domain signal. At this point, signal equalization is necessary. Inherent in the FIR tapped-delay decimation filter are magnitude errors which appear in the frequency spectrum. Equalization is performed to complement this attenuation so that a flat response is attained. The filter attenuation is calculated for each frequency bin of the FFT. At each frequency bin, the signal amplitude is multiplied by the inverse of the corresponding attenuation factor to restore (equalize) the signal content at that bin. Incorporated into ZSIM is equalization for voice-band (up to 4 kHz) applications only.

Using the SDR command, the user must specify the sampling frequency, the passband frequency, the signal frequency, and the type of window for the signal (uniform, Hamming, Blackman-Harris). The type of windowing determines the amount of signal energy leakage (spread) in terms of bins in the frequency domain.
Presently, only signals that coincide with a bin are handled by the routine, which is named SDRSUB and is found in Appendix D. The location of the signal bin, or integer bin counter \((iict)\), and the harmonic bins \((harm)\) is calculated from the different frequencies present in the system. Figure 3.7 is a sample collection of bins that demonstrate part of the SDR calculation. First, the bins must be separated into three categories: signal, white noise, and harmonic noise. This provides for the calculation of each of the three ratios mentioned earlier. Each bin that falls between \((iict - spread)\) and \((iict + spread)\) is considered as signal; each bin that falls between \((harm - spread)\) and \((harm + spread)\) is considered as harmonic noise; other bins are white noise. However, included in each signal/harmonic accumulator is an average white noise level per bin. To insure

---

Figure 3.7: A sample collection of bins around the signal
that all energy is accumulated in the appropriate category, the average white noise present in the signal region and each harmonic region is calculated individually for each region. The average white noise is subtracted from the corresponding signal/harmonic accumulator and added to the white noise accumulator. At this point, the signal, white noise, and harmonic noise accumulators are complete.

In determining average white noise for each region, it is necessary to average bins that are white noise only. Since no special detection routine is implemented to determine the number of bins present between any one harmonic and the next, special care is taken in simulating a system where white noise is present between each harmonic. The bins used to calculate average white noise are the bins adjacent to both sides of the signal/harmonic energy region. The term signal/harmonic energy region includes the harmonic bin and all bins falling in the spread region. Therefore, the average white noise has units noise/bin. Thus, average white noise is calculated by the formula

\[
\text{average white noise} = \frac{\text{bin}(iict - \text{spread} - 1) + \text{bin}(iict + \text{spread} + 1)}{2}
\]

for the signal bin. Replacing \(iict\) with \(harm\) calculates the corresponding average white noise for each harmonic. A final adjustment to the distortion accumulator and white noise accumulator is subtraction of the dc bins (the first two bins). Ratios are then calculated for signal-to-noise, signal-to-harmonic noise, and signal-to-distortion.
3.5. MEMORY REQUIREMENT

3.5.1. Table Storage

A table is stored in a one-dimensional vector named \textit{tables}(x), which contains every table read by the simulator in sequential order. Each individual table stack follows the order of X1 values, X2 values, to Xn values followed by the output values \(T(X_1, X_2, ..., X_n)\) in FORTRAN storage order as earlier described in Section 3.3.1 "Tables: Set-Up and Generation".

Each subsystem of the modulator that is described by a table has a corresponding three-dimensional array \(p_{table}(f_{bindx}, i, position)\) which is a table pointer to the \textit{tables} array, defining memory addresses and/or number of memory items. The \(f_{bindx}\) dimension is the functional block index that determines if the table describes an integrator, comparator, or other subsystem. The index number is located in a subsystem identification array such as \textit{scint} or \textit{quant}. The \(i\) dimension tells which functional block is considered, and \textit{position} is an integer value ranging from 1 to 9. Table information is arranged as follows: \(position=1\) (number of table dimensions), \(position=2\) (starting address in \textit{tables}), \(position=3\) (order of interpolating polynomial), \(position=4\) (number of X1 variables), \(position=5\) (number of X2 variables), and so on, up to \(position=9\) (number of X6 variables).
3.5.2. Integrator Storage, Example of Modularity

It is the purpose of this section to describe ZSIM in terms of variable names and modular capability in the event that modules need to be added in the future. Use of this section and the source code [30] is required for programming guidelines. Emphasis is on an integrator module — storage arrays, special functions, and connectivity to the main routine by subroutines. Memory allocation and circuit connection information is analogous to the GENERATOR, QUANTIZER, and other circuit subsystem modules to be added. Routines studied here are ZCMD, INTGTR, ZANA, ASCINT, and POLATE. See User’s Manual (Appendix A) for the subroutine tree structure.

ZCMD, found in Appendix D, is the main routine for handling commands and directing simulator execution. Adding new modules and their commands begins with ZCMD, where all memory arrays are accessible. Each type of module has its own set of memory arrays. Delta-sigma modulator subsystems, such as an integrator, have three such storage arrays - subsystem identification, subsystem flags, and subsystem table pointers.

From ZCMD, subroutine INTGTR is summoned to define an integrator subsystem, initializing the storage arrays. Specific integrator storage arrays are as follows: identification by scint, flags by intflg, and table pointers by ptable. During the input process, the flag fin acknowledges the absence of data and is used throughout the subroutine to ensure that sufficient data exists. Processing the integrator input command line begins with a HELP message. If help is not needed,
the initialization procedure begins. First, determine which integrator is being defined (let $i =$ integrator number) and check that the limits on the storage arrays are not exceeded, i.e., the maximum number of integrators ($mscint$) and/or subsystem blocks ($mzblk$) is not exceeded. If the integrator already exists ($intflg(i) =$ true), then a replacement is necessary, but the old integrator information ($scint$) must be saved in case an error occurs during the input process. Integrator information is then read into the identification array $scint$ as follows: $scint(i,1) =$ type of integrator (1:analog, 2:switched reference), $scint(i,2) =$ functional block index ($fbindx=1$, used for table selection), $scint(i,3) =$ analog input (node 1), $scint(i,4) =$ output (node 2), and $scint(i,5) =$ switch reference input (node 3).

If the integrator is described by a table already present in memory, the table pointer $ptable$ is simply duplicated for that integrator. Note that $ptable$ values are originally defined during a table read execution in the TABLRD routine. The last qualification for the input sequence is to restore the old integrator data into $scint$ if an error occurred during input.

The next aspect of an integrator subsystem is simulation, which involves the routines ZANA, ASCINT, and POLATE. ZANA is the topology routine which simply calls the ASCINT subroutine. A special feature of ZANA is the $node(k)$ and $nodep(k)$ arrays, which contain the voltage levels at node $k$ for the present and previous clock cycles, respectively. These special arrays are, in a sense, common blocks to be used by all subsystem simulation routines. ASCINT, along with other simulation routines, must first determine input nodes and output nodes.
Since an integrator table is used, this procedure involves equating X1 to the analog input, X2 to the previous output, and X3 to the present digital reference voltage. The next step is to call the interpolation routine POLATE, passing the X1, X2, X3, ptable, and tables variables. An integrator output value is returned from POLATE. It is also necessary to save special node values in nodep(k) for other clock cycles that require circuit memory. Special features, such as noise and voltage tracking, may also be added into a subsystem simulation. If noise is specified for an integrator, subroutine GAUSS is summoned and returns an additive white noise (in respective units) to the integrator output value. Also, the maximum and minimum integrator output is stored in arrays maxout(node) and minout(node), respectively.

The interpolation routine POLATE is somewhat complicated in structure yet simplistic in application. POLATE is divided into two sections: a one-dimensional system and a multi-dimensional system. A one-dimensional system is defined to have an X1 input variable only, i.e., the x and y relationship is strictly one-to-one. All other x and y relationships are multi-dimensional. Polynomial interpolation up to 6th order is possible for a one-dimensional system whereas linear interpolation is required for a multi-dimensional system.

Since each scheme has a similar procedure for interpolation, only the multi-dimensional system procedure will be reviewed. The first requirement is to use the table pointer ptable to locate both input and output table values in the vector tables. The next step is to identify the table points to be used in the interpolation.
formula. The two X1 points are denoted $s_1$ and $f_1$ and the two X2 points are denoted $s_2$ and $f_2$. The four output table points determined by the X1 and X2 limits are denoted by $pt_1$, $pt_2$, $pt_3$, and $pt_4$. Now the interpolation formula derived in Appendix C is executed. The output is denoted by $out$.

A special feature of POLATE is to create a histogram of table usage. Basically, each table point has its own integer counter. Thus, for each interpolation, four counters are updated and the sum of all counters is four times the number of clock cycles simulated. This histogram is useful in that it shows which table points are not used and need not be executed by a circuit-level simulator, thus reducing execution time.

3.6. Computational Speed

The primary goal of ZSIM is the fast simulation of delta-sigma modulators. Thus, the idea of table-based simulation is directed toward sampled-data systems that require a large number of clock cycles for circuit evaluation. In this section, simulation speed of ZSIM is compared to other simulation techniques (for a DEC MICROVAX II system running Ultrix 1.2 at about 1 MIP).

Table 3.2 presents a timing comparison between ZSIM and SPICE for a signal-to-distortion ratio calculation of a first-order delta-sigma modulator with an ideal quantizer. Simulation is for one input signal level and $2^{16}$ clock cycles. First notice that the total simulation time, when ZSIM reads a table from memory, is only twice as long as the difference equation time. This factor is not a disadvan-
tage since the table simulation is more accurate and includes all circuit nonlinearities.

A 120 point table (2x6x10) is assumed for the integrator. Using the integrator discussed later in Chapter 4, a SPICE simulation for one clock period takes 715 seconds. For 120 individual simulations, 1430 CPU minutes are required to generate a complete set of table values. ZSIM table simulation takes only 3 minutes and the SDR calculation takes only 2 minutes. Total time for a first-run ZSIM evaluation is then 1435 minutes (approximately 1 day). The estimated SPICE time is a prediction based on 715 seconds per clock cycle multiplied by the number of clock cycles ($2^{16}$), or 780,790 minutes (approximately 1.5 years). (This estimated time does not include the extra time required to simulate a complete circuit which includes the comparator and the feedback path.) Therefore, the ZSIM simulation shows approximately 550 times speed-up over SPICE. Note that once a table for a specific integrator is generated and stored in memory, additional simulations take

| Table 3.2: Run time comparison using tables, difference equations, and SPICE |
|---------------------------------|---|---|---|
|                                | ZSIM | SPICE |
|                                | Table look-up simulation | Difference equation simulation | Circuit-level simulation |
| first run                      | 1430 min | -- | -- |
| other runs                     | -- | 780,972 min | -- |
| generate table SPICE           | 3 min | 2 min | 2 min |
| read table                     | -- | 1 min | -- |
| simulation                     | 3 min | 3 min | 2 min |
| digital signal processing      | 2 min | 2 min | 2 min |
| TOTAL TIME                     | 1435 min | 6 min | 3 min | 780,972 min |
only 6 minutes to evaluate the system. However, for each additional SPICE simula-
tion, another 1.5 years is needed. Therefore, each subsequent run using ZSIM
results in a 200,000 times speed-up over SPICE.
CHAPTER 4

4. BENCHMARK SIMULATIONS

The purpose of this chapter is to demonstrate the program functionality and to show the accuracy of ZSIM. For the case of a voiceband Coder-Decoder (CODEC) delta-sigma modulator, an ideal table-based simulation is compared to a direct difference equation simulation to prove that the table method retains accuracy and is a valid simulation tool. These results are presented as a high-level benchmark in Section 4.1. An application to a real Integrated Services Digital Network (ISDN) circuit is presented as a low-level benchmark in Section 4.2. Tables are constructed using SPICE and CAzM (Circuit Analyzer with Macromodeling) for each integrator of the ISDN delta-sigma modulator. These simulations will show circuit dependencies related to noise and other circuit phenomena not modeled by difference equations but incorporated into the table representations.
4.1. HIGH-LEVEL BENCHMARK

4.1.1. Voice Band Circuit

This section will consider a delta-sigma modulator system for use in a voice-band CODEC application. Figure 4.1 shows the connectivity of the system. A second-order delta-sigma modulator is chosen with $\alpha_1 = 0.1$, $\alpha_2 = 0.5$, and $\beta_1 = 0.1$. The sampling frequency is $f_s = 1.024$ MHz. Assuming a 5 volt power supply, the comparator output is a binary ±2.5 V. The integrators are set to saturate (hard limit) at ±1.5 V. The modulator input will be a 1 kHz sinewave.

The digital output of the modulator is fed into an FIR decimation filter (tapped delay line) using parabolic weighting with 128 taps to provide the neces-
sary out of band noise rejection [12,32]. The sampling rate is reduced by a factor of \( M = 32 \) down to 32 kHz, and these samples are then filtered using an IIR (cascade) baseband filter to further attenuate and shape the frequency response. The coefficients of this filter are found in the file VB.CAS (see User’s Manual, Appendix A). After baseband filtering, the sampling rate is reduced from 32 kHz to 8 kHz.

4.1.2. Simulation Results

Simulation of the CODEC delta-sigma modulator was performed using both ZSIM’s direct difference equation simulator and table-based simulator. All simulations were executed with 65,536 \( (2^{16}) \) clock cycles. The difference equation set-up command is given by

\[
\text{EQ \ GAIN1=0.1 \ GAIN2=0.5 \ SAT=1.5 \ DELTA1=2.5 \ DELTA2=0.25.}
\]

These circuit parameters were then used to format a table for each integrator. The tables for Integrator 1 and Integrator 2 are shown in Figures 4.2 and 4.3, respectively. The tables can also be represented graphically as in Figures 4.4 and 4.5, where it is possible to see the effect of linear interpolation with hard saturation. Difference equation and table-based simulations were performed for the CODEC modulator. These simulations should produce identical results except near the saturation limit. A comparison of the baseband spectra and SDR results are presented in the next section.
* INTEGRATOR #1

- Integrator gain 0.1
- Switch voltage 2.5

begin

\[ x_1 = -2.5 -1.75 -1.0 -0.5 -0.1 0.1 0.5 1.0 1.75 2.5 \]
\[ x_2 = -1.5 -1.2 -0.9 -0.6 -0.3 0.3 0.6 0.9 1.2 1.5 \]
\[ x_3 = -1 1 \]

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>x3</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.5</td>
<td>-1.425</td>
<td>-1.35</td>
</tr>
<tr>
<td>-1.2</td>
<td>-1.125</td>
<td>-1.05</td>
</tr>
<tr>
<td>-0.9</td>
<td>-0.825</td>
<td>-0.75</td>
</tr>
<tr>
<td>-0.6</td>
<td>-0.525</td>
<td>-0.45</td>
</tr>
<tr>
<td>-0.3</td>
<td>-0.225</td>
<td>-0.15</td>
</tr>
<tr>
<td>0.3</td>
<td>0.375</td>
<td>0.45</td>
</tr>
<tr>
<td>0.6</td>
<td>0.675</td>
<td>0.75</td>
</tr>
<tr>
<td>0.9</td>
<td>0.975</td>
<td>1.05</td>
</tr>
<tr>
<td>1.2</td>
<td>1.275</td>
<td>1.35</td>
</tr>
<tr>
<td>1.5</td>
<td>1.500</td>
<td>1.50</td>
</tr>
<tr>
<td>-1.5</td>
<td>-1.500</td>
<td>-1.50</td>
</tr>
<tr>
<td>-1.4</td>
<td>-1.325</td>
<td>-1.25</td>
</tr>
<tr>
<td>-1.1</td>
<td>-1.025</td>
<td>-0.95</td>
</tr>
<tr>
<td>-0.8</td>
<td>-0.725</td>
<td>-0.65</td>
</tr>
<tr>
<td>-0.2</td>
<td>-0.125</td>
<td>-0.05</td>
</tr>
<tr>
<td>0.1</td>
<td>0.175</td>
<td>0.25</td>
</tr>
<tr>
<td>0.4</td>
<td>0.475</td>
<td>0.55</td>
</tr>
<tr>
<td>0.7</td>
<td>0.775</td>
<td>0.85</td>
</tr>
<tr>
<td>1.0</td>
<td>1.075</td>
<td>1.15</td>
</tr>
</tbody>
</table>

done

Figure 4.2: CODEC integrator #1 ideal table
* INTEGRATOR # 2

* Integrator gain 0.50
* Switch voltage 0.25

begin

\[
x_1 = -1.5 \quad -1.2 \quad -0.9 \quad -0.6 \quad -0.3 \quad 0.3 \quad 0.6 \quad 0.9 \quad 1.2 \quad 1.5 \\
x_2 = -1.5 \quad -1.2 \quad -0.9 \quad -0.6 \quad -0.3 \quad 0.3 \quad 0.6 \quad 0.9 \quad 1.2 \quad 1.5 \\
x_3 = -1 \\
\]

\[
\begin{array}{cccccccccccc}
-1.500 & -1.500 & -1.500 & -1.500 & -1.500 & -1.225 & -1.075 & -0.925 & -0.775 & -0.625 \\
-1.500 & -1.500 & -1.500 & -1.375 & -1.225 & -0.925 & -0.775 & -0.625 & -0.475 & -0.325 \\
-1.500 & -1.375 & -1.225 & -1.075 & -0.925 & -0.625 & -0.475 & -0.325 & -0.175 & -0.025 \\
-1.225 & -1.075 & -0.925 & -0.775 & -0.825 & -0.325 & -0.175 & -0.025 & 0.125 & 0.275 \\
-0.925 & -0.775 & -0.625 & -0.475 & -0.325 & -0.025 & 0.125 & 0.275 & 0.425 & 0.575 \\
-0.325 & -0.175 & -0.025 & 0.125 & 0.275 & 0.575 & 0.725 & 0.875 & 1.025 & 1.175 \\
0.025 & 0.125 & 0.275 & 0.425 & 0.575 & 0.875 & 1.025 & 1.175 & 1.325 & 1.475 \\
0.275 & 0.425 & 0.575 & 0.725 & 0.875 & 1.175 & 1.325 & 1.475 & 1.500 & 1.500 \\
0.575 & 0.725 & 0.875 & 1.025 & 1.175 & 1.475 & 1.500 & 1.500 & 1.500 & 1.500 \\
0.875 & 1.025 & 1.175 & 1.325 & 1.475 & 1.500 & 1.500 & 1.500 & 1.500 & 1.500 \\
-1.500 & -1.500 & -1.500 & -1.500 & -1.500 & -1.475 & -1.325 & -1.175 & -1.025 & -0.875 \\
-1.500 & -1.500 & -1.500 & -1.500 & -1.500 & -1.475 & -1.175 & -1.025 & -0.875 & -0.725 & -0.575 \\
-1.500 & -1.500 & -1.475 & -1.325 & -1.175 & -0.875 & -0.725 & -0.575 & -0.425 & -0.275 \\
-1.475 & -1.325 & -1.175 & -1.025 & -0.875 & -0.575 & -0.425 & -0.275 & -0.125 & 0.025 \\
-1.175 & -1.025 & -0.875 & -0.725 & -0.575 & -0.275 & -0.125 & 0.025 & 0.175 & 0.325 \\
-0.575 & -0.425 & -0.275 & -0.125 & 0.025 & 0.325 & 0.475 & 0.625 & 0.775 & 0.925 \\
-0.275 & -0.125 & 0.025 & 0.175 & 0.325 & 0.625 & 0.775 & 0.925 & 1.075 & 1.225 \\
0.025 & 0.175 & 0.325 & 0.475 & 0.625 & 0.925 & 1.075 & 1.225 & 1.375 & 1.500 \\
0.325 & 0.475 & 0.825 & 0.775 & 0.925 & 1.225 & 1.375 & 1.500 & 1.500 & 1.500 \\
0.625 & 0.775 & 0.925 & 1.075 & 1.225 & 1.500 & 1.500 & 1.500 & 1.500 & 1.500 \\
\end{array}
\]

done

Figure 4.3: CODEC integrator #2 ideal table
Figure 4.4: Graph representation of integrator #1 for (a) $p(k) = -1$, (b) $p(k) = 1$
Figure 4.5: Graph representation of integrator #2 for (a) $p(k) = -1$, (b) $p(k) = 1$
4.1.2.1. Baseband Spectrum

Simulations were performed for two cases - an input amplitude of -15 dB (with respect to the comparator reference 2.5V) and the maximum amplitude of 0 dB. Operation of the integrators for -15 dB is basically in the linear region and no limiting occurs. However, limiting does occur in the integrators for a 0 dB input amplitude.

Figure 4.6 shows the baseband spectra for a -15 dB input amplitude. The table method simulations closely match the results using a difference equation.
The slight deviations may be due to the limited accuracy of the table storage. This accuracy can be improved by extending the length of the precision of the data.

The baseband spectra for a 0 dB input amplitude is shown in Figure 4.7. The table result is close to the difference equation spectrum with some deviation in power across the band. Distortion components are present in both methods, but with some difference in level. The difference in levels can be attributed to pure hard limiting in the direct difference equation as compared to softer limiting in the tables. The latter is caused by interpolation between the linear and saturation regions given a finite number of table points. Since a pure hard limiter does not

![Figure 4.7: CODEC baseband spectrum for input = 0 dB](image)
exist in a true circuit, this harmonic disagreement may be of minimal concern.

4.1.2.2. Signal-to-Distortion Ratio

Table 4.1 compares the signal-to-distortion ratio (distortion includes noise plus total harmonic distortion) for both methods of simulation. The SDR as a function of input amplitude is plotted in Figure 4.8. Close agreement is obtained at low signal levels whereas deviations at large signal levels occur due to integrator saturation, as described earlier. Figure 4.9 is a plot of the differences in SDR for the two simulation methods. Notice that this difference is highly random. The slight differences are attributed to the accuracy of the interpolation routine and the finite precision of the Fourier transform routine. Better accuracy can be

<table>
<thead>
<tr>
<th>Input</th>
<th>Diff. Equat.</th>
<th>Ideal Table</th>
<th>Input</th>
<th>Diff. Equat.</th>
<th>Ideal Table</th>
<th>Input</th>
<th>Diff. Equat.</th>
<th>Ideal Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>40.9661</td>
<td>40.5639</td>
<td>-11</td>
<td>71.8130</td>
<td>70.8580</td>
<td>-25</td>
<td>56.8837</td>
<td>57.6987</td>
</tr>
<tr>
<td>-1</td>
<td>65.5709</td>
<td>65.2184</td>
<td>-12</td>
<td>70.2594</td>
<td>70.4663</td>
<td>-28</td>
<td>53.0780</td>
<td>52.6831</td>
</tr>
<tr>
<td>-2</td>
<td>71.8916</td>
<td>72.8751</td>
<td>-13</td>
<td>67.6239</td>
<td>69.4333</td>
<td>-30</td>
<td>52.4806</td>
<td>53.3591</td>
</tr>
<tr>
<td>-3</td>
<td>73.1701</td>
<td>73.7618</td>
<td>-14</td>
<td>69.2771</td>
<td>68.6706</td>
<td>-33</td>
<td>48.6391</td>
<td>51.1124</td>
</tr>
<tr>
<td>-4</td>
<td>75.4289</td>
<td>74.4117</td>
<td>-15</td>
<td>67.3672</td>
<td>67.1737</td>
<td>-35</td>
<td>48.2805</td>
<td>47.3207</td>
</tr>
<tr>
<td>-5</td>
<td>72.8379</td>
<td>73.8041</td>
<td>-16</td>
<td>66.2523</td>
<td>66.5136</td>
<td>-38</td>
<td>44.9667</td>
<td>45.3714</td>
</tr>
<tr>
<td>-6</td>
<td>73.5745</td>
<td>74.7970</td>
<td>-17</td>
<td>67.1231</td>
<td>66.3335</td>
<td>-40</td>
<td>42.7303</td>
<td>43.0408</td>
</tr>
<tr>
<td>-7</td>
<td>74.9449</td>
<td>73.1008</td>
<td>-18</td>
<td>65.8062</td>
<td>64.4709</td>
<td>-45</td>
<td>36.2685</td>
<td>38.0489</td>
</tr>
<tr>
<td>-8</td>
<td>76.3002</td>
<td>72.4187</td>
<td>-19</td>
<td>64.4583</td>
<td>64.9478</td>
<td>-50</td>
<td>32.1171</td>
<td>31.5382</td>
</tr>
<tr>
<td>-9</td>
<td>71.8306</td>
<td>73.2247</td>
<td>-20</td>
<td>63.6741</td>
<td>61.9838</td>
<td>-55</td>
<td>28.2827</td>
<td>28.0385</td>
</tr>
<tr>
<td>-10</td>
<td>70.4733</td>
<td>70.9682</td>
<td>-23</td>
<td>59.2016</td>
<td>58.8462</td>
<td>-60</td>
<td>25.0812</td>
<td>24.1595</td>
</tr>
</tbody>
</table>
Figure 4.8: SDR versus input amplitude for CODEC delta-sigma modulator
Figure 4.9: SDR differences (difference equation-table) versus input amplitude

attained through use of larger tables and a better interpolation process.
4.1.2.3. Interpolation Accuracy

The interpolation errors in the above example can be understood by considering two cases with the integrator operating in the linear region and in the saturation region. Table 4.2 is a summary of interpolation and difference equation results for a specific clock cycle during a simulation using an input signal level of 0 dB. The first integrator is operating in the linear region whereas the second integrator is operating in the saturation region. The interpolation output is taken directly from a node voltage listing and the difference equation output is calculated using Equation 3.1. Recall that the reference voltage is 2.5 V for the first integrator and 0.25 V for the second integrator. Notice that for the first integrator the interpolation routine generates an output value that is exactly as calculated by the difference equation. This result is as expected since the integrator operation is linear and the interpolation routine is linear. However, the interpolated result for the second integrator is slightly different. The result given is the exact answer to

<table>
<thead>
<tr>
<th>INTEGRATOR #1</th>
<th>Interpolation formula</th>
<th>Difference equation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>0.619197</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INTEGRATOR #2</th>
<th>Interpolation formula</th>
<th>Difference equation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>1.43547</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INTEGRATOR #2</th>
<th>Interpolation formula</th>
<th>Difference equation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>1.51914</td>
</tr>
</tbody>
</table>

Table 4.2: Summary of integrator simulation for cycle #249
the difference equation, although a hard limit of 1.5 V was specified, and 1.5 V would be the actual result. For this specific cycle a 4.3 % difference exists between the table result and the difference equation saturation level. In effect, the interpolation scheme sees the saturation earlier than it actually occurs. Therefore, a soft limiting effect is introduced into the integrator. The overall performance of the delta-sigma modulator is not degraded to a large extent, however, as shown in the previous section.

4.2. LOW-LEVEL BENCHMARK

4.2.1. ISDN Circuit

This section considers a delta-sigma modulator system for use in ISDN applications. The connectivity of the circuit is the same as in the previous example (see Figure 4.1) except that an IIR filter is not used. A second-order delta-sigma modulator is chosen with $\alpha_1 = 0.1$ and $\alpha_2 = 0.5$ and $\beta_1 = 0.1$. The selection of $\beta_1$ insures that the range of voltages at both inputs of the second integrator are equivalent. The sampling frequency is $f_s = 5.12$ MHz. The decimation filter has parabolic weighting with 128 taps, and the sampling rate is reduced by a factor of $M = 32$ to 160 kHz with an 80 kHz baseband frequency. For the benchmark study, the modulator input is a 30 kHz sinewave.

Figure 4.10 shows the second-order switched-capacitor circuit designed to implement the modulator. The topology for each switched-capacitor integrator
insures that its transfer function is independent of parasitic capacitances between any node and ground [33, 34]. The comparator is ideal whereas the operational amplifiers are not; a class AB op-amp (schematic in Appendix B) is designed for a 1.0 μm CMOS process. \( \Phi_1 \) and \( \Phi_2 \) are nonoverlapping clocks, each with a duty cycle close to 50 percent. Integrator #1's output is stable at the end of \( \Phi_1 \) and Integrator #2's output is valid at the end of \( \Phi_2 \). Note that the clock timing causes the first integrator to be a summing integrator with negative gain and the second integrator to be a difference integrator with positive gain.
4.2.2. Table Generation

A DC analysis of the class AB op-amp was performed using SPICE. The op-amp was powered by +2.5 V and -2.5 V supplies, so all signals are with respect to 2.5 V. Results of the analysis show that the upper saturation limit is +1.25 V and the lower saturation limit is -1.45 V. However, an AC analysis of the op-amp with the addition of the switching circuits showed that the output swings from -1.5 V to 1.5 V. Therefore, ZSIM's difference equation set-up line is given by:

\[ \text{EQ GAIN1=0.1 GAIN2=0.5 SAT=1.5 DELTA1=2.5 DELTA2=0.25} \]

Difference equation simulations were performed for an input range of -60 dB to 0 dB. Results show that the output voltage of the first integrator never exceeds 0.65 V, but that the second integrator does in fact reach saturation. Therefore, the tables representing each integrator can be reduced in size since one variable (X2 for Int.#1 and X1 for Int.#2) does not fluctuate over the entire range of possible voltages.

For the first integrator, with the gain as given, the analog input ranges between -2.5 and 2.5 V, while the output swings between ± 0.65 V. Therefore, the discretization points for the table were chosen to be \( X_1 = -2.5, -1.75, -1.0, -0.5, -0.1, 0.1, 0.5, 1.0, 1.75, 2.5 \) and \( X_2 = -0.7, -0.5, -0.3, -0.1, 0.1, 0.3, 0.5, 0.7 \). The continuous input to the second integrator ranges from -0.65 to 0.65 V, and the output swings between -1.5 and 1.5 V. For the second table, we choose \( X_1 = -0.7, -0.5, -0.3, -0.1, 0.1, 0.3, 0.5, 0.7 \) and \( X_2 = -1.5, -1.3, -0.8, -0.4, -0.1, 0.1, 0.4, 0.8, 1.3, 1.5 \). Zero volts was avoided as a table entry to prevent distortion of small sig-
nals due to finite precision.

The ideal tables for each integrator are shown in Figures 4.11 and 4.12. The integrators are the same as those used in the CODEC application although the ISDN tables use different discretization points. Notice that a complete table for the second integrator is not necessary since \( X_2 \) can never be negative when \( X_3 \) is positive and vice versa. However, the two \( X_2 \) values surrounding the zero point

---

* INTEGRATOR #1
* Integrator gain 0.1
* Switch voltage 2.5

begin

\[
\begin{array}{ccccccccccccccccccccccccccc}
\text{x1} &=& -2.5 &-1.75 &-1.0 &-0.5 &-0.1 &0.1 &0.5 &1.0 &1.75 &2.5 \\
\text{x2} &=& -0.7 &-0.5 &-0.3 &-0.1 &0.1 &0.3 &0.5 &0.7 \\
\text{x3} &=& -1.0 &-0.8 &-0.6 &-0.4 &-0.2 &0.2 &0.4 &0.6 &0.8 &1.0 \\
\end{array}
\]

* \( \text{x1} = \text{x2} \)

---

Figure 4.11: ISDN integrator #1 ideal table
* INTEGRATOR #2
* Integrator gain  0.50
* Switch voltage  0.25

begin
x1= -0.7 -0.5 -0.3 -0.1 0.1 0.3 0.5 0.7
x2= -1.5 -1.3 -0.8 -0.4 -0.1 0.1 0.4 0.8 1.3 1.5
x3= -1 1

\[
\begin{array}{cccccccccc}
-1.500 & -1.500 & -1.500 & -1.425 & -1.325 & -1.225 & -1.125 & -1.025 \\
-1.500 & -1.425 & -1.325 & -1.225 & -1.125 & -1.025 & -0.925 & -0.825 \\
-1.025 & -0.925 & -0.825 & -0.725 & -0.625 & -0.525 & -0.425 & -0.325 \\
-0.825 & -0.525 & -0.425 & -0.325 & -0.225 & -0.125 & -0.025 & 0.075 \\
-0.325 & -0.225 & -0.125 & -0.025 & 0.075 & 0.175 & 0.275 & 0.375 \\
-0.125 & -0.025 & 0.075 & 0.175 & 0.275 & 0.375 & 0.475 & 0.575 \\
-0.575 & -0.475 & -0.375 & -0.275 & -0.175 & -0.075 & 0.025 & 0.125 \\
-0.375 & -0.275 & -0.175 & -0.075 & 0.025 & 0.125 & 0.225 & 0.325 \\
-0.075 & 0.025 & 0.125 & 0.225 & 0.325 & 0.425 & 0.525 & 0.625 \\
0.325 & 0.425 & 0.525 & 0.625 & 0.725 & 0.825 & 0.925 & 1.025 \\
0.825 & 0.925 & 1.025 & 1.125 & 1.225 & 1.325 & 1.425 & 1.500 \\
1.025 & 1.125 & 1.225 & 1.325 & 1.425 & 1.500 & 1.500 & 1.500 \\
\end{array}
\]
done

Figure 4.12: ISDN integrator #2 ideal table

must have table values so that the interpolation is valid between zero volts and the next X2 value. This reduces the number of points that must be simulated in a circuit-level simulator from 160 to 96 for the second integrator. On the other hand, the table for the first integrator uses four X2 values surrounding the zero point, since the comparator decision is dependent on the second integrator and not
the first. The possibility may arise where a negative X2 value does exist when X3 is positive and vice versa. The number of table points for the first integrator is reduced from 160 to 120 points. Missing table points are replaced with 9.9 so that the program will return an interpolation error if this part of the table is ever used.

ZSIM table simulations were performed for two sets of tables. One set was generated by SPICE runs and one set was generated by CAzM runs. The purpose of two table sets is to compare the end results and describe potential problems related to different circuit-level simulators. A LEVEL 2 MOSFET model is used in both the SPICE and CAzM simulations. Specifically, SPICE uses capacitive-based device models whereas CAzM uses charge-based device models. It is a well known fact that capacitive models do not accurately simulate some dynamic circuits, resulting in a nonconservation of charge in which the charge stored in a node is not equal to the integrated net current flowing into the node [35,36]. Therefore, CAzM models will show better performance in switched-capacitor circuit simulations.

A series of individual transient circuit simulations were performed to build up the tables one point at a time. Some precautions must be considered during these simulations. First, the integrator output and other signal path nodes must be correctly initialized for each run. Second, the timing of the switched-capacitor integrator during table simulation must match the timing of the complete modulator. In particular, the critical time points for each node must be correctly identified.
Each simulation begins 4 ns before a clock cycle and continues for one clock period, ensuring that node initialization occurs during the correct part of the clock phase (nodes will not be correctly initialized if transient simulation begins between clock phases). The integrator nodes are initialized by forcing an initial condition on the feedback capacitor. Forcing the integrator output to a desired value with a voltage source causes incorrect initialization since the virtual ground node is not simultaneously initialized to the offset voltage of the amplifier. However, forcing an initial voltage across the capacitor correctly initializes both the output and virtual ground nodes. Since the virtual ground node is not actually at zero volts, the output voltage will not exactly match the capacitor initialization voltage. This poses no problem since the X2 discretization points in the table can be adjusted to this output value. Appendix B contains a sample SPICE output file, showing initial voltages and clock timing for a one period simulation that produces one table point. The CAzM file is similar but is not included. Figures 4.13 and 4.14 show the first integrator tables generated by SPICE and CAzM, respectively. Figures 4.15 and 4.16 show the second integrator tables generated by SPICE and CAzM, respectively. Note that the CAzM table values, when compared to the SPICE table values, are closer to the ideal table values. Both integrators modeled by SPICE exhibit differences in output of up to 20 mV when compared to the ideal difference equation solution, whereas CAzM showed differences of up to 4 mV.

To compare SPICE and CAzM overall performance, a sample of data points was chosen from the tables representing the first integrator. The data point set
* INTEGRATOR #1
* Integrator gain 0.1
* Switch voltage 2.5

begin
xl= -2.5 -1.75 -1.0 -.5 -.1 .1 .5 1.0 1.75 2.5
x2= -.7018 -.5018 -.3018 -.1018 .0982 .2981 .4981 .6981
x3= -1 1

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-.8930</td>
<td>-.6186</td>
<td>-.5440</td>
<td>-.4886</td>
<td>-.4822</td>
<td>-.4361</td>
<td>-.3958</td>
<td>-.3465</td>
<td>-.2855</td>
<td>-.1919</td>
</tr>
<tr>
<td>-.4874</td>
<td>-.4269</td>
<td>-.3455</td>
<td>-.3028</td>
<td>-.2565</td>
<td>-.2367</td>
<td>-.1968</td>
<td>-.1404</td>
<td>-.0737</td>
<td>.0830</td>
</tr>
<tr>
<td>-.2939</td>
<td>-.2275</td>
<td>-.1451</td>
<td>-.1041</td>
<td>-.0572</td>
<td>-.0369</td>
<td>.0018</td>
<td>.0583</td>
<td>.1260</td>
<td>.1931</td>
</tr>
<tr>
<td>-.0940</td>
<td>-.0273</td>
<td>.0530</td>
<td>.1028</td>
<td>.1420</td>
<td>.1625</td>
<td>.2021</td>
<td>.2449</td>
<td>.3254</td>
<td>.3924</td>
</tr>
<tr>
<td>.0981</td>
<td>.1847</td>
<td>.2522</td>
<td>.2948</td>
<td>.3485</td>
<td>.3625</td>
<td>.3946</td>
<td>.4435</td>
<td>.5303</td>
<td>.6044</td>
</tr>
<tr>
<td>.3036</td>
<td>.3840</td>
<td>.4519</td>
<td>.5007</td>
<td>.5468</td>
<td>.5536</td>
<td>.5937</td>
<td>.6494</td>
<td>.6981</td>
<td>.6981</td>
</tr>
<tr>
<td>-.7018</td>
<td>-.7018</td>
<td>-.6406</td>
<td>-.5981</td>
<td>-.5578</td>
<td>-.5247</td>
<td>-.4920</td>
<td>-.4425</td>
<td>-.3684</td>
<td>-.2940</td>
</tr>
<tr>
<td>-.5826</td>
<td>-.5156</td>
<td>-.4477</td>
<td>-.3921</td>
<td>-.3582</td>
<td>-.3320</td>
<td>-.2989</td>
<td>-.2368</td>
<td>-.1750</td>
<td>-.0940</td>
</tr>
<tr>
<td>-.3837</td>
<td>-.3094</td>
<td>-.2409</td>
<td>-.1989</td>
<td>-.1586</td>
<td>-.1316</td>
<td>-.0998</td>
<td>-.0430</td>
<td>.0136</td>
<td>.1048</td>
</tr>
<tr>
<td>-.1977</td>
<td>-.1186</td>
<td>-.0429</td>
<td>.0060</td>
<td>.0532</td>
<td>.0735</td>
<td>.1068</td>
<td>.1622</td>
<td>.2291</td>
<td>.3034</td>
</tr>
<tr>
<td>.0148</td>
<td>.0894</td>
<td>.1587</td>
<td>.1999</td>
<td>.2482</td>
<td>.2658</td>
<td>.3054</td>
<td>.3809</td>
<td>.4349</td>
<td>.4958</td>
</tr>
<tr>
<td>.2142</td>
<td>.2753</td>
<td>.3556</td>
<td>.3986</td>
<td>.4388</td>
<td>.4650</td>
<td>.5047</td>
<td>.5539</td>
<td>.6348</td>
<td>.8953</td>
</tr>
</tbody>
</table>

done

Figure 4.13: ISDN integrator #1 table generated from SPICE
* INTEGRATOR #1
* Integrator gain  0.1
* Switch voltage  2.5

begin

x1= -2.5  -1.75  -1.0  -0.5  -0.1  0.1  0.5  1.0  1.75  2.5
x2=  -0.7013  -0.5013  -0.3013  -0.1013  0.0987  0.2987  0.4987  0.6987
x3=  -1  1  

-0.7006  -0.8257  -0.5507  -0.5008  -0.4608  -0.4408  -0.4008  -0.3508  -0.2759  -0.2010
-0.5007  -0.4257  -0.3508  -0.3008  -0.2608  -0.2408  -0.2008  -0.1509  -0.0760  -0.0011
-0.3008  -0.2258  -0.1509  -0.1009  -0.0609  -0.0409  -0.0010  0.0490  0.1239  0.1987
-0.1009  -0.0259  0.0490  0.0990  0.1390  0.1590  0.1989  0.2488  0.3237  0.3986
  0.0990  0.1740  0.2489  0.2988  0.3388  0.3588  0.3987  0.4487  0.5235  0.5985
  0.2989  0.3738  0.4487  0.4987  0.5386  0.5586  0.5985  0.6484  0.6987  0.6987
-0.7013  -0.7013  -0.6504  -0.6005  -0.5605  -0.5405  -0.5005  -0.4506  -0.3758  -0.3008
-0.6005  -0.5254  -0.4505  -0.4006  -0.3606  -0.3406  -0.3006  -0.2507  -0.1758  -0.1009
-0.4006  -0.3255  -0.2506  -0.2007  -0.1607  -0.1408  -0.1007  -0.0508  0.0241  0.0990
-0.2006  -0.1255  -0.0507  -0.0007  0.0393  0.0593  0.0992  0.1491  0.2240  0.2989
-0.0007  0.0743  0.1493  0.1992  0.2392  0.2592  0.2991  0.3490  0.4239  0.4988
  0.1993  0.2741  0.3490  0.3990  0.4390  0.4590  0.4989  0.5488  0.6237  0.6986
done

Figure 4.14: ISDN integrator #1 table generated from CAzM
* INTEGRATOR #2
* Integrator gain 0.50
* Switch voltage 0.25

begin
x1= -0.7 -0.5 -0.3 -0.1 0.1 0.3 0.5 0.7
x2= -1.5011 -1.3017 -.8018 -.4018 -.1018 .0982 .3981 1.2975 1.4964
x3= -1 1

-1.5011 -1.5011 -1.5011 -1.4154 -1.3284 -1.2199 -1.1209 -1.0218
-1.5318 -1.4336 -1.3347 -1.2359 -1.1487 -1.0399 -0.9409 -0.8415
-1.0337 -0.9261 -0.8349 -0.7360 -0.6486 -0.5494 -0.4502 -0.3512
-0.8254 -0.5349 -0.4352 -0.3273 -0.2486 -0.1402 -0.0409 0.0488
-0.3228 -0.2238 -0.1242 -0.0160 0.0624 0.1615 0.2702 0.3691
-0.1251 -0.0259 0.0649 0.1727 0.2512 0.3504 0.4590 0.5489
-0.5824 -0.4834 -0.3751 -0.2846 -0.1879 -0.0980 0.0102 0.1004
-0.3824 -0.2746 -0.1837 -0.0846 0.0121 0.1110 0.2103 0.3004
-0.0736 0.0186 0.1182 0.2243 0.3119 0.4112 0.5101 0.6003
0.3172 0.4253 0.5249 0.6238 0.7029 0.8110 0.9100 1.0004
0.8261 0.9165 1.0163 1.1151 1.2112 1.3097 1.4078 1.4964
1.0260 1.1164 1.2248 1.3229 1.4098 1.4964 1.4964 1.4964
done

Figure 4.15: ISDN integrator #2 table generated from SPICE
* INTEGRATOR #2
* Integrator gain 0.50
* Switch voltage 0.25

begin

\begin{align*}
\text{x1} &= -0.7 -0.5 -0.3 -0.1 0.1 0.3 0.5 0.7 \\
\text{x2} &= -1.4996 -1.3004 -0.8013 -0.4002 -0.1014 0.0986 0.3985 0.7985 1.298 1.4998 \\
\text{x3} &= -1 1
\end{align*}

\begin{tabular}{cccccccccccc}
-1.4996 & -1.4996 & -1.4996 & -1.4217 & -1.3234 & -1.2249 & -1.1262 & -1.0274 \\
-1.4996 & -1.4207 & -1.3225 & -1.2240 & -1.1253 & -1.0265 & -0.9275 & -0.8280 \\
-1.0242 & -0.9252 & -0.8257 & -0.7262 & -0.6268 & -0.5272 & -0.4278 & -0.3283 \\
-0.6241 & -0.5247 & -0.4252 & -0.3257 & -0.2263 & -0.1267 & -0.0273 & 0.0721 \\
-0.3251 & -0.2255 & -0.1260 & -0.0265 & 0.0730 & 0.1725 & 0.2720 & 0.3714 \\
-0.1252 & -0.0255 & 0.0740 & 0.1735 & 0.2729 & 0.3724 & 0.4719 & 0.5713 \\
0.1749 & 0.2744 & 0.3739 & 0.4733 & 0.5728 & 0.6723 & 0.7717 & 0.8712 \\
-0.8736 & -0.7742 & -0.6747 & -0.5752 & -0.4758 & -0.3763 & -0.2768 & -0.1774 \\
-0.5745 & -0.4750 & -0.3755 & -0.2760 & -0.1766 & -0.0770 & 0.0225 & 0.1219 \\
-0.3748 & -0.2751 & -0.1756 & -0.0761 & 0.0234 & 0.1229 & 0.2224 & 0.3218 \\
-0.0747 & 0.0248 & 0.1243 & 0.2238 & 0.3233 & 0.4228 & 0.5223 & 0.6217 \\
0.3252 & 0.4247 & 0.5241 & 0.6237 & 0.7231 & 0.8226 & 0.9220 & 1.0213 \\
0.8247 & 0.9243 & 1.0236 & 1.1229 & 1.2221 & 1.3209 & 1.4194 & 1.4998 \\
1.0276 & 1.1268 & 1.2261 & 1.3249 & 1.4235 & 1.4998 & 1.4998 & 1.4998 \\
\end{tabular}

done

Figure 4.16: ISDN integrator #2 table generated from CAzM

represents X1 = -1.0 V over the entire range of X2 and X3 values (one randomly selected column from the table). The differential error is calculated for each data point as compared to the ideal difference equation solution for each input combination. The results are depicted in Figure 4.17. Note that SPICE error is an order of magnitude greater than CAzM error. Also note that the SPICE error lacks mono-
This nonmonotonic error is present for each column of the SPICE table, but it occurs at the lower end of the $p[k] = -1$ curve when the $X1$ value is positive. The magnitude of these errors is such that the error is asymmetrical with respect to zero volts. ZSIM simulations using these tables are presented in the next section and will show performance degradation following the magnitude of the above described errors.
4.2.3. System Performance

Simulations were performed for the ISDN circuit using ideal tables, SPICE tables, and CAzM tables. The purpose of these tests is to show that circuit non-linearities are inherently modeled by the tables and therefore produce a degradation in overall system performance by the inclusion of amplifier distortion in the simulation.

Table 4.3 compares the SDR for each type of simulation for an input range of -60 dB to 0 dB. Figure 4.18 uses this data and graphically compares SDR versus

Table 4.3: SDR summary for ideal table, SPICE table, and CAzM table simulations

<table>
<thead>
<tr>
<th>Input</th>
<th>Ideal Table</th>
<th>CAzM Table</th>
<th>SPICE Table</th>
<th>Input</th>
<th>Ideal Table</th>
<th>CAzM Table</th>
<th>SPICE Table</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>45.9876</td>
<td>42.7910</td>
<td>42.9484</td>
<td>-17</td>
<td>54.4488</td>
<td>55.4130</td>
<td>38.3206</td>
</tr>
<tr>
<td>-1</td>
<td>59.8002</td>
<td>59.8692</td>
<td>56.2801</td>
<td>-18</td>
<td>54.1800</td>
<td>54.2184</td>
<td>37.3219</td>
</tr>
<tr>
<td>-2</td>
<td>64.8946</td>
<td>64.7157</td>
<td>58.3784</td>
<td>-19</td>
<td>53.3141</td>
<td>52.7361</td>
<td>36.8485</td>
</tr>
<tr>
<td>-3</td>
<td>65.7073</td>
<td>65.3697</td>
<td>57.0753</td>
<td>-20</td>
<td>51.9250</td>
<td>51.9363</td>
<td>37.4184</td>
</tr>
<tr>
<td>-4</td>
<td>64.8560</td>
<td>64.8847</td>
<td>53.7449</td>
<td>-23</td>
<td>48.8842</td>
<td>48.9576</td>
<td>34.4917</td>
</tr>
<tr>
<td>-5</td>
<td>64.3137</td>
<td>64.4671</td>
<td>51.1329</td>
<td>-25</td>
<td>46.3904</td>
<td>46.6596</td>
<td>39.2423</td>
</tr>
<tr>
<td>-6</td>
<td>64.2173</td>
<td>63.9290</td>
<td>49.2058</td>
<td>-28</td>
<td>43.8861</td>
<td>43.4973</td>
<td>36.0918</td>
</tr>
<tr>
<td>-7</td>
<td>63.3018</td>
<td>63.4182</td>
<td>48.2367</td>
<td>-30</td>
<td>41.7764</td>
<td>41.8049</td>
<td>38.6269</td>
</tr>
<tr>
<td>-8</td>
<td>62.0797</td>
<td>62.6921</td>
<td>49.9208</td>
<td>-33</td>
<td>38.5196</td>
<td>38.4169</td>
<td>31.3868</td>
</tr>
<tr>
<td>-9</td>
<td>61.9208</td>
<td>62.0810</td>
<td>52.0792</td>
<td>-35</td>
<td>37.2634</td>
<td>35.9685</td>
<td>25.7691</td>
</tr>
<tr>
<td>-10</td>
<td>61.7608</td>
<td>61.5106</td>
<td>49.0343</td>
<td>-38</td>
<td>32.6229</td>
<td>32.8922</td>
<td>28.2288</td>
</tr>
<tr>
<td>-11</td>
<td>60.2181</td>
<td>60.5481</td>
<td>44.4672</td>
<td>-40</td>
<td>29.0808</td>
<td>31.3950</td>
<td>25.4819</td>
</tr>
<tr>
<td>-14</td>
<td>58.0728</td>
<td>58.2841</td>
<td>37.0555</td>
<td>-55</td>
<td>15.5312</td>
<td>16.3829</td>
<td>11.6745</td>
</tr>
<tr>
<td>-15</td>
<td>56.8528</td>
<td>57.2770</td>
<td>37.1518</td>
<td>-60</td>
<td>9.1073</td>
<td>12.2133</td>
<td>6.8413</td>
</tr>
<tr>
<td>-16</td>
<td>55.7005</td>
<td>56.0783</td>
<td>37.0567</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
input signal amplitude. The actual performance of the circuit shows a reduction in SDR over the entire input range when compared with its ideal performance. Notice that the performance of the circuit is dramatically higher using CAzM tables rather than SPICE tables. This is due to more noise present in the SPICE simulations, a direct result of the errors previously shown in Figure 4.17. The degradation is attributed to the different device models, i.e., CAzM is free of charge conservation problems present in SPICE MOSFET models.

Further investigation of system performance shows an increase in circuit noise for frequencies less than the signal frequency when using SPICE tables rather than

![Figure 4.18: SDR versus input amplitude for ISDN delta-sigma modulator](image)

**Figure 4.18:** SDR versus input amplitude for ISDN delta-sigma modulator
CAzM tables. This result can be seen in the baseband frequency spectra of Figure 4.19. The limited accuracy of the tables may possibly be the cause of the additional low frequency noise and the nonmonotonicity of the errors in the SPICE tables creates the second harmonic. Other nonlinearities captured by the circuit-level simulations include clock feed-through, saturation, and finite slewing. However, most of the SDR degradation for the SPICE simulation is determined to be numerical noise, finite precision, and charge conservation problems. The spectrum for difference equation simulations has been excluded from Figure 4.19 since it is essentially the same as the CAzM result. Overall, CAzM has given better ZSIM

Figure 4.19: ISDN Baseband spectrum for input = -10 dB
CHAPTER 5

5. FUTURE INVESTIGATIONS

This chapter describes the limitations encountered during the course of research and some ideas for future investigations of delta-sigma simulation using ZSIM. Specific problems, such as interpolation and table generation, will be addressed. Also, additions to ZSIM code will be suggested.

5.1. TABLES AND INTERPOLATION

On the basis of this work, it is seen that table-based techniques provide a practical means for simulating circuit-level circuits at the system level. The tables both capture circuit nonlinearities and increase simulation speed as compared to circuit-level simulations. Tables may require large amounts of computer memory to store the data, however, memory is relatively inexpensive and should not pose a problem. However, present in ZSIM are some limitations which are related to the structure of tables and in the interpolation process.

Currently, a table is stored as a rectangular matrix, or grid. This implementation may be practical for general linear functions, but the tables will become large as a result of the rectangular spacing in each dimension when very nonlinear functions are modeled. Rectangular spacing also creates table points that are
never used. If ZSIM is to remain as a general purpose sampled-data simulator, rectangluar grid spacing should do well because it has flexibility to describe any function, even hysteresis (hysteresis demands a table for each direction of operation). The only real limitation would be memory space, but this should not be a problem because memory is relatively inexpensive. A grid pattern that is not rectangular (NxM) could be implemented in the future if and only if the system requires high precision modeling for very nonlinear functions. Implementation of nonrectangular grids will probably require much research time in determining how to store and interpolate tables.

Table interpolation is also an issue. Linear interpolation seems to work well since the circuits have been carefully designed to avoid saturation, which is the greatest form of nonlinearity in a delta-sigma modulator. Higher order polynomial fitting (Newton's method) is possible for a function of one input variable only; linear polynomial fitting is used for multi-variable functions since simulation involves interpolation of planar surfaces. Higher order interpolation might be implemented for multi-variable functions if the routine had intelligence to locate the planar surfaces in an N-dimensional table so that more than four data points could be used for interpolation. In this case, a study should be constructed for intersecting lines and surfaces [37]. A separate issue would be interpolating in a nonrectangular grid structure as previously mentioned.

In applications to delta-sigma modulators, current ZSIM table structure and interpolation routine is more than adequate. However, the above suggestions
should reduce table storage size and increase simulation accuracy, but computational speed may be degraded if different table structures and more complicated interpolation schemes are used. This tradeoff must be considered in the context of the type of system to be simulated.

5.2. ZSIM ADDITIONS

The primary addition contemplated for ZSIM is the development of a capability for general topologies. As the program stands, each subsystem definition includes nodal connections. These node numbers are used only for internal storage of subsystem information and not for arbitrary circuit configurations. Given a facility for automated circuit connectivity analysis, the program would also need to check for signal flow through the circuit and to order the subsystem simulations. It would still be up to the user to generate a subsystem table that agrees with all loading and timing considerations of the connected circuit.

Input/output could be improved with a few more options. The input format should include a loop structure so that many commands need not be repeated. For example, an SDR versus input signal amplitude analysis currently requires changing the signal level as well as repeating the SIMULATE, DECIMATE, and SDR commands for each signal level. Output options could include numeral precision length (which is not a variable in FORTRAN) and some graphing capabilities (possibly for the DIGITAL LN03 laser printer).
Another addition to ZSIM would be comparator tables. Presently, the quantizer model is ideal and neglects such circuit problems as hysteresis, switching delay, and asymmetrical switching levels. This addition would involve modifying the quantizer input routine (COMPAR) and adding the interpolation option to the comparator simulator (AQUANT).

The Post-Processor Module could also use some additions and changes. The first consideration should be including an SDR calculation for signals not coinciding with an FFT bin. An extension of this work is determining uneven signal spread and separating it from the random noise. Also, incorporation of the IIR filter design routines, available through CCSP, is suggested.
LIST OF REFERENCES


APPENDIX A

ZSIM User's Manual
1. VERSION NOTES

ZSIM0a1 is the first distribution of ZSIM. This version is tailored to the simulation of Delta Sigma Modulators (DSM). It has a user-friendly input format but lacks a totally stand alone topology specification. Currently up to third order DSM can be simulated using difference equations and first and second order DSM can be simulated using table methods.

2. INSTALLATION GUIDE

ZSIM0a1 is written in ANSI standard FORTRAN 77 and operation has been verified on DEC MICROVAXES running Ultxrix 1.2 or MICROVMS operating systems.

ZSIM0a1 is distributed as two sets of files - ZSIMALL.FOR and the individual module files.

ZSIMALL.FOR is just all the routines below plus fdate.f concatenated together.

** ZSIMALL file **
Zsimall should be compiled and linked.

** individual module files **
The files below should be compiled and linked together.

agen aquant ascint calfbw calsdr
casfil cassec compar decimate desim
dffil dtable envirn fft fread
gauss gendec gentor init intgtr
nlfil nodept nodset noiset polate
ranf sdrsub tabled zana sbdec
scalsdr scmd sdec zdump sequat
shelp zinput sprint zedr zset
zzsim

The program calls a routine fdate using
   call fdate(date)
where date is dimensioned as
   character*24 date
On UNIX systems this returns the date. On nonunix machines the above routine can be provided or the dummy file fdate.f (included in this distribution) can be included in the above list.
3. I/O CONVENTIONS

All input to ZSIM is converted to uppercase at input. On some systems, e.g. unix based, user's need to be aware that files are always accessed in uppercase by the program.

There is a high level input capability so that input can be taken from the keyboard and/or from a disk file - see user's guide for more information.

Comments can be included by preceding an input line by ' * '.

4. PROGRAM ORGANIZATION

The program is organized by sets of modules with a common set of high level input routines contained in the file zinput.f.

The program is block oriented. A high level view of the program is as follows:

Level of program: top next

ZSIM

INPUT Input parameters

SIMULATION Perform simulation of sampled data system

DECIMATE Perform decimation

SDR Perform SDR calculations

Greater detail is given on the next page.

Description of a circuit must be in terms of nodes however in ZSIM0a1 arbitrary topology has not been completely implemented. ZSIM needs to be informed as to the interconnection of blocks in the sampled data system. At present only first and second order delta sigma modulators can be modeled (indicated by the parameter CIRCUIT) using table modeling techniques and first second and third order delta sigma modulators (again indicated by the parameter CIRCUIT) using difference equation techniques.

The distinction between a table based simulation and a difference equation simulation is indicated by the command to do the simulation (the command SIMULATE performs a table based simulation and DESIMULATE does a difference equation simulation).
ZSIM: A Nonlinear Z-domain SIMulator

Input Routines: ZINPUT, FREAD
This guide is organized by commands. ZSIM is largely self-documenting and help on each command is available by typing "COMMAND" HELP.

### Commands Available:

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>KEYWORD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMULATE</td>
<td>SIM</td>
<td>Perform z-domain simulation (tables)</td>
</tr>
<tr>
<td>DESIMULATE</td>
<td>DES</td>
<td>Perform s-domain simulation (diff eq)</td>
</tr>
<tr>
<td>GEN</td>
<td>GEN</td>
<td>Define generator input</td>
</tr>
<tr>
<td>SCINT</td>
<td>SCINT</td>
<td>Define integrator input</td>
</tr>
<tr>
<td>QUANT</td>
<td>QUANT</td>
<td>Define quantizer input</td>
</tr>
<tr>
<td>TABLE</td>
<td>TABLE</td>
<td>Define table description</td>
</tr>
<tr>
<td>CIRCUIT</td>
<td>CIRC</td>
<td>Define circuit type</td>
</tr>
<tr>
<td>ENVIRONMENT</td>
<td>ENV</td>
<td>Define simulation features</td>
</tr>
<tr>
<td>HISTOGRAM</td>
<td>HIST</td>
<td>Specify histogram for the tables</td>
</tr>
<tr>
<td>CLEAR</td>
<td>CLEAR</td>
<td>Erase all parameters to default</td>
</tr>
<tr>
<td>DUMP</td>
<td>DUMP</td>
<td>Dump contents of stored node values</td>
</tr>
<tr>
<td>STOP</td>
<td>STOP</td>
<td>End session</td>
</tr>
<tr>
<td>INIT</td>
<td>INIT</td>
<td>Initialize quantities</td>
</tr>
<tr>
<td>DECIMATE</td>
<td>DEC</td>
<td>Perform a decimation</td>
</tr>
<tr>
<td>SDR</td>
<td>SDR</td>
<td>Signal-to-Distortion calculation</td>
</tr>
<tr>
<td>EQUATION</td>
<td>EQ</td>
<td>Specify difference equation parameters</td>
</tr>
<tr>
<td>NOISE</td>
<td>NOISE</td>
<td>Define Gaussian white noise at a node</td>
</tr>
</tbody>
</table>

The following input/output commands are available:

- `read filename`: the file filename is opened for input
- `read`: the previously opened file is now used for input
- `write filename`: the file filename is opened for output
- `eof`: the input data file is closed
- `end`: the input data file is temporarily closed
- `title`: write line to output file
- `prompt`: write line to terminal
- `echo on`: the flag prt is set
- `echo off`: the flag prt is cleared

If prt is set all lines input are echoed.
Usage:

CIRCUIT HELP

: This Message

CIRCUIT = n

: Set order of modulator to "n"

Usage:

CLEAR HELP

: This Message

CLEAR

: Set everything to default value

Usage:

DECIMATE HELP

: This Message

DECIMATE n1 n2 TYPE=ZBDEC

This performs a decimation on the values of node n1 and outputs the result (usually the baseband) at n2 TYPE indicates the type of canned decimation.

DECIMATE n1 n2 TYPE=GENDEC NUMBER=nn WINDOW=sss TAPS=nn
INTDEC=nn BBANDDEC=bb FILE=filename FILTER=sss

This performs a decimation on the values of node n1 and outputs the result (usually the baseband) at n2. NUMBER can be to base 2, e.g.: NUMBER=nn BASE 2

WINDOWS available: UNIFORM TRIANGLE PARABOLIC

TAPS : length of the decimation filter. INTDEC: FIR decimation factor

BBANDDEC : baseband decimation factor. FILE specifies file with filter coefficients

FILTERS available: DIRECT_FORM NORMALIZED_LATTICE CASCADE_FORM

Included in this distribution of ZSIM is the file VB.CAS which specifies a cascade filter designed for decimation to voiceband. VB.CAS has a 3.4 kHz rolloff for an 8kHz sampling frequency. Coefficients for other filters must be generated by the user. A program for doing this is available from CCSP.
DIFFERENCE EQUATION SIMULATION

Usage:

DES HELP
  : This Message

DES
  : Simulate a circuit described by difference equations

DUMP

Usage:

DUMP HELP
  : This Message

DUMP file n no
  : Dump first "no" values at node "n" to "file"

INITIALIZE

Usage:

INIT HELP
  : This Message

INIT NODE n x
  : Initializes NODE n to x. This is required as several blocks use previous node values in calculations

NOISE

Usage:

NOISE HELP
  : This Message

NOISE n x1 x2
  : Add Gaussian white noise to node "n" with mean "x1"
  : and standard deviation "x2"

** NOT TESTED OR FULLY IMPLEMENTED **
#### EQUATION

**Usage:**

**EQ HELP**
: This Message

**EQ GAIN#=xxx1 DELTA#=xxx2 OFINT#=xxx4 SAT#%=xxx5**
: # = 1, 2, or 3 for respective integrator but may be
: left out to describe all integrators.
: This inputs ideal circuit parameters:
: GAIN#: gain of integrator
: DELTA#: reference voltage of quantizer
: OFINT#: integrator offset referred to the output
: SAT#%: integrator saturation
: % = '+' for upper limit
: % = '-' for lower limit

**NOTE** Finite Gain Bandwidth is not implemented.

**Default Values are:**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>gain1</td>
<td>1.00</td>
<td>gain2</td>
<td>1.00</td>
<td>gain3</td>
</tr>
<tr>
<td>delta1</td>
<td>1.00</td>
<td>delta2</td>
<td>1.00</td>
<td>delta3</td>
</tr>
<tr>
<td>ofint1</td>
<td>0.0</td>
<td>ofint2</td>
<td>0.0</td>
<td>ofint3</td>
</tr>
<tr>
<td>sat1</td>
<td>5.00</td>
<td>sat2</td>
<td>5.00</td>
<td>sat3</td>
</tr>
</tbody>
</table>

**Difference equation implemented:**

\[ y#(new) = y#(old) + \text{gain#} \times (x# \times \text{bit} \times \text{delta#}) + \text{ofint#} \]

\[ y#(new) = \text{is limited to} + \text{or} - \text{sat#} \]

where

- \( y# \) is the output of the \# integrator
- \( y#(new) \) is the new output of the integrator
- \( y#(old) \) is the output of the integrator at the previous clock cycle
- \( x \) is the analog input
- \( \text{gain#}, \text{delta#} \) and \( \text{ofint#} \) are as defined above
- \( \text{bit} \) is the output of the quantizer (+ or - 1)

---

---
### GENERATOR

#### Usage:

**GEN HELP**

: This Message

**GEN no node TYPE = DC AMP=x2**

: generator number "no" at node "node"
: DC generator of amplitude "x2" and

**GEN no node TYPE = RAMP THR[ESHOLD]=x1 AMP=x2**

: generator number "no" at node "node"
: ramp generator of amplitude "x2" and
: threshold of "x1"

**GEN no node TYPE = SINE FREQ= freq OFFSET=x1 AMP=x2**

: generator number "no" at node "node"
: sinewave generator of frequency "freq",
: amplitude "x2", offset "x1", and zero phase

**GEN no node TYPE = SINE FREQ= freq AMP=x1 PHASE = x2 DEG**

: generator number "no" at node "node"
: sinewave generator of frequency "freq",
: amplitude "x1", and phase of "x2" degrees

**GEN no node TYPE = SINE FREQ= freq AMP=x1 DB WRT x3 PHASE = x4 RAD**

: generator number "no" at node "node"
: sinewave generator of frequency "freq",
: amplitude "x1" With Respect To "x3" V and
: phase of "x4" radians

**GEN no node TYPE=STEP INIT=x1 AMP=x2 DELAY=x3**

: step generator with initial value "x1", amplitude
: "x2" and delay of "x3" seconds
QUANTIZER

Usage:

QUANT HELP
: This Message

QUANT n n1 n2 THRES=xxx1 TYPE=ss1 BAND=xxx2 SAME=nnn1
: Quantizer "n" at input node "n1" and output node "n2"
: has THRESHOLD of "xxx1" and TYPE = NONHysteresis or
: HYSTeresis.
: options 1. previously defined quantizer number "nnn1" uses the same data table
: 2. "xxx2" is a deadzone region where the quantizer output zero.
: NOTE:: option 1 is presently disabled since the quantizer is ideal
: and not characterized by a table as of now.
: TYPE must be specified NONHysteresis until a hysteresis table
: is programmed.
: Integrator table must include x3=0 (see section 5 on generating
: tables) when BAND option is used.

SIMULATION USING TABLES

Usage:

SIM HELP
: This Message

SIM
: Simulate a circuit described by tables
SDR HELP

This Message

SDR n1 n2 TYPE=CALSDR NPOINTS=nnn1 NFFT=nnn2 NSKIP=nnn3
: FFTWINDOW=sss1 DECWINDOW=sss2 FB=xxx1 TAPS=nnn4
: This performs a Signal-to-Distortion calculation on
: the values of node n1. The calculated spectrum is
: stored as node n2. Options available:
: NPOINTS = total number of input bins
: NFFT = number of bins to use for fft
: NSKIP = number of initial bins to ignore
: (NPOINTS = NFFT + NSKIP)
: FFTWINDOW = UNIFORM HAMMing blackman-HARRIs
: DECWINDOW = IGNORE UNIFORM TRIAngular PARAbolic
: (DECWINDOW should be the same used in decimation.)
: The following parameters can be ignored if
: decimation not used (DECWINDOW=IGNORE)
: FB : baseband sampling frequency
: TAPS : number of taps in decimation filter must be the
: same as that used in the decimate command.

Warning: The parameters in SDR must agree with those in
the decimate command.

We suggest that FFTWINDOW = UNIFORM be used as only this has
be debugged and tested. This amounts to not using a window and
is satisfactory provided the input signal coincides with a bin.

The SDR calculation determines the signal level SIG, the
total distortion level DIST, the noise level NOIS, and the
harmonic level HARM. DC is excluded from the calculation.

DIST = NOIS + HARM

Since very few bins (of the frequency spectrum) are typically available,
the noise in the location of the harmonics is estimated by averaging the
noise on either side of the harmonics and subtracting it from the total
signal level at the harmonic bins. The FFTWINDOW used determines
the signal spread and the number of bins over which the signal levels
should be determined.
### SCINT - INTEGRATOR

#### Usage:

**SCINT HELP**

: This Message

**SCINT n TYPE=SWITCH n1 n2 n3 SAME=nnn1**

: switched capacitor integrator number "n" with positive input node "n1"

: and negative input node "n2" and output node "n3".

: option- > previously defined integrator number "nnn1" uses the same data table

**SCINT n TYPE=ANALOG DIM=nnn1 n1 n2 SAME=nnn2**

: analog integrator number "n" with table DIMENSION "nnn1"

: (excluding input) and input node "n1" and output node "n2".

: option- > previously defined integrator number "nnn2" uses the same data table

**WARNING**: Since arbitrary circuit topology has not been completely implemented, the ANALOG integrator should not be used.

### STOP

#### Usage:

**STOP HELP**

: This Message

**STOP**

: End ZSIM session

### TABLE

#### Usage:

**TABLE HELP**

: This Message

**TABLE SCINT n file**

: "file" is the input file which contains the TABLE

: that describes the switched capacitor integrator number "n".

**TABLE QUANT n file**

: "file" is the input file which contains the TABLE

: that describes the quantizer number "n".

**NOTE**: quantizer table not yet operational!
6. GENERATING YOUR OWN TABLE

Currently tables can only be used to describe the characteristics of switched capacitor integrators.

Integrator

\[
\begin{align*}
\text{analog input (x1)} & \quad \longrightarrow \quad \text{feedback bit (x3)} \quad \longrightarrow \quad \text{output of integrator (x2)} \\
\end{align*}
\]

\(x1\) and \(x2\) can be any real value
\(x3\) must be an integer

This is described by the difference equation

\[x2(n) = x2(n-1) + G(x1(n) - x3(n-1))\]

where \(n\) and \(n-1\) refer to cycle numbers and \(G\) is gain.

format of table: comments:

* comment lines
\[x1(1) \ldots x1(i) \ldots x1(l)\] ! This is the input to the integrator at the current ! cycle. Values of \(x1\) are in monotonic order.

\[x2(1) \ldots x2(j) \ldots x2(J)\] ! This is the output of the integrator at the previous ! cycle. Values of \(x2\) are in monotonic order.

\[x3(1) \ldots x3(k) \ldots x3(K)\] ! This is the binary feedback bit for the integrator at ! the current cycle. Values of \(x3\) are in monotonic ! order and must be integers.

begin ! instruction initiating table

! The tables are arranged so that \(x3\) varies slowest, ! then \(x2\) and \(x1\) varys the fastest each line of ! the table corresponds to constant \(x2\) and \(x3\) and ! the numbers on a line are for the output \(x2\) at ! the current cycle for each value of \(x1\).
The purpose of the tables is to represent an output y as an arbitrary function of inputs X1, X2, ... Xn. i.e. \( y = T(X_1, X_2, \ldots X_n) \). Let \( y = T(x_1, y_2) \) represent the predefined points in the table. Crude table interpolation is incorporated in ZSIM at present but this is adequate for simulating delta-sigma modulators (DSMs). The output of an integrator in a DSM is the function of two continuous variables - the analog input X1 and the previous value of the integrator output X2 - and one discrete variable - the fed back bit X3 which usually is binary. Thus in DSMs the table describes the function \( y = T(X_1, X_2, X_3) \). The discrete value reduces this to a 2 dimensional function \( y = T(X_1, X_2) \). This is a plane which can be depicted as shown. Generally the X1 and X2 values will not coincide with the quantization of the graph and so it will be necessary to interpolate to determine y. This is done by first using the X1 value and linearly interpolating along lines of constant X2 to reduce the function to be determined to \( y = T(X_2) \). Now linear interpolation with respect to X2 yields the desired value of the function. This concept can be extended to multiple dimensions.
Table based simulation of a second-order Delta-Sigma Modulator clocked at 1.024 MHz with an input signal of 1 kHz. The SDR is calculated for voiceband.

Circuit:

Input files: example_circuit, TABLE1, TABLE2, VB.CAS
Output files: example_output, NODE4, NODE5, NODE6
These files are printed below together with a keyboard transcript of the session.

KEYBOARD TRANSCRIPT (EXAMPLE)

\%
ZSIM version ZSIM0a1
Nonlinear Z Domain Simulator
Brauns, Steer, Ardalan, ECE Dept
North Carolina State University

* read example_circuit

"example_circuit"
"example_output"

Decimation Parameters From File: " VB.CAS "
32768 points decimated to 256
Signal-to-Distortion Ratio (dB) 54.3512
Signal-to-Noise Ratio (dB) 55.2068
Signal-to-Harmonic Distortion Ratio (dB) 61.8269
DUMP file: "NODE4 "
no. values output = 100 node number = 4
DUMP file: "NODE5 "
no. values output = 256 node number = 5
DUMP file: "NODE6"

no. values output = 128 node number = 6

? replacing old generator?

Decimation Parameters From File: " VB.CAS"

32768 points decimated to 256

Signal-to-Distortion Ratio (dB) 63.6741
Signal-to-Noise Ratio (dB) 63.8709
Signal-to-Harmonic Distortion Ratio (dB) 77.2103

INPUT FILE: example_circuit

* file: example_circuit
print parameters on
echo on
write example_output
histogram

* set up environment: sampling frequency and the number of clock cycles
environ freq=1.024 MHZ
environ cycles=32768

* since the topology is not completely automatic, specify that we are
* simulating a second order circuit
circuit=2

* describe generator: the level is with respect to 2.5 V
* we could have used AMP = 0.079
gen 1 1 type=sine freq=1000 amp=-30 db wrt 2.5

* describe the first switched capacitor integrator with tables
scint 1 type=switch 1 4 2
table scint 1 table1

* describe the second switched capacitor integrator with tables
scint 2 type=switch 2 4 3
table scint 2 table2

* describe the quantizer as ideal; output is +1 or -1
quant 1 3 4 threshold=0 type=nonhys
* initialize node values - not completely necessary by this will
* eliminate some warning messages
init node 1 0
init node 2 0
init node 3 0
init node 4 1

* now start the simulation
simulate

* ***************************************************************
* VOICEBAND PERFORMANCE parabolic decimation weighting
* sdr calculated using 256 baseband bins
* sdr calculated with equalization
* sdr ignores first 128 transient bins

* input: -30 dB wrt 2.5 V

decimate 4 5 type=gendec number=32768 window=para taps=128 intdec=32
    bbanddec=4 file=vb.cas filter=cascade

sdr 5 6 type=calsdr npoints=256 nskip=128 nfft=128 fftwin=unif
    decwin=para fb=4000 fs=8000 taps=128 signal=1000

* print out first 100 bits of bitstream
dump node4 4 100

* print out the baseband signal
dump node5 5 256

* print out the spectrum for the second set of 128 baseband points
dump node6 6 128

* -------------------------------------------------------------
* now simulate with difference equations for input
* signal level of -20 dB wrt 2.5 V

gen 1 1 type=sine freq=1000 amp=-20 dB wrt 2.5
eq gain1=0.1 gain2=0.5 sat=1.5 delta1=2.5 delta2=0.25

init node 1 0
init node 2 0
init node 3 0
init node 4 1

desimulate
decimate 4 5 type=gentdec number=32768 window=para taps=128
intdec=32 bbanddec=4 file=vb.cas filter=cascade

deewin=para
cb=4000 (8=8000 taps=128 signal=1000

INPUT FILE: TABLE 1

* INTEGRATOR #1
* Integrator gain : 0.10000
* Switch voltage : 2.50000

begin
x1= -2.50 -1.75 -1.00 -0.50 -0.10 0.10 0.50 1.00 1.75 2.50
x2= -0.90 -0.70 -0.50 -0.30 -0.10 0.10 0.30 0.50 0.70 0.90
x3= -1 1

-0.900 -0.825 -0.750 -0.700 -0.660 -0.640 -0.600 -0.550 -0.475 -0.400
-0.700 -0.625 -0.550 -0.500 -0.460 -0.440 -0.400 -0.350 -0.275 -0.200
-0.500 -0.425 -0.350 -0.300 -0.280 -0.240 -0.200 -0.150 -0.075 0.000
-0.300 -0.225 -0.150 -0.100 -0.060 -0.040 0.000 0.050 0.125 0.200
-0.100 -0.025 0.050 0.100 0.140 0.160 0.200 0.250 0.325 0.400
 0.100 0.175 0.250 0.300 0.340 0.360 0.400 0.450 0.525 0.600
 0.300 0.375 0.450 0.500 0.540 0.580 0.600 0.850 0.725 0.800
 0.500 0.575 0.650 0.700 0.740 0.780 0.800 0.850 0.900 0.900
 0.700 0.775 0.850 0.900 0.900 0.900 0.900 0.900 0.900 0.900
 0.900 0.900 0.900 0.900 0.900 0.900 0.900 0.900 0.900 0.900
-0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900
-0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.850 -0.775 -0.700
-0.900 -0.900 -0.850 -0.800 -0.760 -0.740 -0.700 -0.850 -0.575 -0.500
-0.800 -0.725 -0.650 -0.600 -0.560 -0.540 -0.500 -0.450 -0.375 -0.300
-0.600 -0.525 -0.450 -0.400 -0.360 -0.340 -0.300 -0.250 -0.175 -0.100
-0.400 -0.325 -0.250 -0.200 -0.160 -0.140 -0.100 -0.050 0.025 0.100
-0.200 -0.125 -0.050 0.000 0.040 0.080 0.100 0.150 0.225 0.300
0. 0.075 0.150 0.200 0.240 0.260 0.300 0.350 0.425 0.500
0.200 0.275 0.350 0.400 0.440 0.460 0.500 0.550 0.625 0.700
0.400 0.475 0.550 0.600 0.640 0.680 0.700 0.750 0.825 0.900

done
INPUT FILE: TABLE 2

* INTEGRATOR #2
* Integrator gain : 0.500000
* Switch voltage : 0.250000

begin
x1 = -0.70 -0.50 -0.30 -0.10 0.10 0.30 0.50 0.70
x2 = -1.50 -1.30 -0.80 -0.40 -0.10 0.10 0.40 0.80 1.30 1.50
x3 = -1.0 1.0

-1.5000 -1.5000 -1.5000 -1.4250 -1.3250 -1.2250 -1.1250 -1.0250
-1.5000 -1.4250 -1.3250 -1.2250 -1.1250 -1.0250 -0.9250 -0.8250
-1.0250 -0.9250 -0.8250 -0.7250 -0.6250 -0.5250 -0.4250 -0.3250
-0.8250 -0.7250 -0.6250 -0.5250 -0.4250 -0.3250 -0.2250 -0.1250
-0.3250 -0.2250 -0.1250 -0.0250 0.0750 0.1750 0.2750 0.3750
-0.1250 -0.0250 0.0750 0.1750 0.2750 0.3750 0.4750 0.5750
0.1750 0.2750 0.3750 0.4750 0.5750 0.6750 0.7750 0.8750
0.5750 0.6750 0.7750 0.8750 0.9750 1.0750 1.1750 1.2750
1.0750 1.1750 1.2750 1.3750 1.4750 1.5000 1.5000 1.5000
1.2750 1.3750 1.4750 1.5000 1.5000 1.5000 1.5000 1.5000
-1.5000 -1.5000 -1.5000 -1.4750 -1.3750 -1.2750 -1.1750 -1.0750
-1.5000 -1.5000 -1.5000 -1.4750 -1.3750 -1.2750 -1.1750 -1.0750
-1.2750 -1.1750 -1.0750 -0.9750 -0.8750 -0.7750 -0.6750 -0.5750
-0.8750 -0.7750 -0.6750 -0.5750 -0.4750 -0.3750 -0.2750 -0.1750
-0.5750 -0.4750 -0.3750 -0.2750 -0.1750 -0.0750 0.0250 0.1250
-0.3750 -0.2750 -0.1750 -0.0750 0.0250 0.1250 0.2250 0.3250
-0.0750 0.0250 0.1250 0.2250 0.3250 0.4250 0.5250 0.6250
0.3250 0.4250 0.5250 0.6250 0.7250 0.8250 0.9250 1.0250
0.8250 0.9250 1.0250 1.1250 1.2250 1.3250 1.4250 1.5000
1.0250 1.1250 1.2250 1.3250 1.4250 1.5000 1.5000 1.5000
done

INPUT FILE: VB.CAS

(This file describes the coefficients for the cascade filter)
(This file was produced by a program external to ZSIM)

5
1.000000 0.0000000E+00
-1.347701 1.000000
-1.253062 1.000000
-0.9458774 0.9999999
8.8302962E-02 0.9999999
-1.530978 0.9652057
-1.513509 0.8840800
-1.517837 0.7729724
-1.531974 0.6509438
-0.7899454 0.0000000E+00
32000.00
1909.065000000000
EXAMPLE

OUTPUT FILE: example_output

"example_output"

" Mon Oct 5 08:55:05 1987"

ZSIM version ZSIM0a2
Nonlinear Z Domain Simulator
Brauns, Steer, Ardalan, ECE Dept
North Carolina State University

* histogram

* * EXAMPLE SIMULATION
* * Perform one table method simulation and one difference equation
* * simulation. Tables are derived by difference equation.
* *
* * set up environment: sampling frequency and the number of clock cycles
* environ freq=1.024 MHZ
* environ cycles=32768
* * since the topology is not completely automatic, specify that we are
* * simulating a second order circuit
* circuit=2
* * describe generator: the level is with respect to 2.5 V
* * we could have used AMP = 0.079
* gen 1 1 type=sine freq=1000 amp=-30 db wrt 2.5
* * describe the first switched capacitor integrator with tables
* scint 1 type=switch 1 4 2
* table scint 1 table1
T * INTEGRATOR #1
T * Integrator gain : 0.10000
T * Switch voltage : 2.50000
T begin
T x1= -2.50 -1.75 -1.00 -0.50 -0.10 0.10 0.50 1.00 1.75 2.50
T x2= -0.90 -0.70 -0.50 -0.30 -0.10 0.10 0.30 0.50 0.70 0.90
T x3= -1 1
T -0.900 -0.825 -0.750 -0.700 -0.660 -0.640 -0.600 -0.550 -0.475 -0.400
T -0.700 -0.825 -0.550 -0.500 -0.460 -0.440 -0.400 -0.350 -0.275 -0.200
T -0.500 -0.425 -0.350 -0.300 -0.260 -0.240 -0.200 -0.150 -0.075 0.000
T -0.300 -0.225 -0.150 -0.100 -0.060 -0.040 0.000 0.050 0.125 0.200
T -0.100 -0.025 0.050 0.100 0.140 0.160 0.200 0.250 0.325 0.400
T 0.100 0.175 0.250 0.300 0.340 0.360 0.400 0.450 0.525 0.600
T 0.300 0.375 0.450 0.500 0.540 0.560 0.600 0.650 0.725 0.800
T 0.500 0.575 0.650 0.700 0.740 0.760 0.800 0.850 0.900 0.900
T 0.700 0.775 0.850 0.900 0.900 0.900 0.900 0.900 0.900 0.900
**describe the second switched capacitor integrator with tables**

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>(-0.70)</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>(-1.50)</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>(-1.0)</td>
</tr>
</tbody>
</table>

**describe the quantizer as ideal; output is +1 or -1**

<table>
<thead>
<tr>
<th>Quant 1</th>
<th>3</th>
<th>4</th>
<th>Threshold = 0</th>
<th>Type = nonhys</th>
</tr>
</thead>
</table>

**T done**
* * initialize node values - not completely necessary by this will
* * eliminate some warning messages
* init node 1 0
* init node 2 0
* init node 3 0
* init node 4 1
* * now start the simulation
* simulate

 INTEGRATOR OUTPUT

Integrator # 1
Input max value = 7.90589e-02  Input min value = -7.90589e-02
Output max value = 0.425817   Output min value = -0.427949

Integrator # 2
Input max value = 0.425817   Input min value = -0.427949
Output max value = 0.334090   Output min value = -0.337891

 HISTOGRAM FOR TABLE 1

 x1 values: 10
 x2 values: 10
 x3 values: 2

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 916 916 0 0 0 0
0 0 0 0 9721 9721 0 0 0 0
0 0 0 0 15164 15164 0 0 0 0
0 0 0 0 6663 6663 0 0 0 0
0 0 0 0 304 304 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 294 294 0 0 0 0
0 0 0 0 7563 7563 0 0 0 0
0 0 0 0 14927 14927 0 0 0 0
0 0 0 0 8821 8821 0 0 0 0
0 0 0 0 1163 1163 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
HISTOGRAM FOR TABLE 2

x1 values: 8
x2 values: 10
x3 values: 2

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 297 8158 8703 2858 16 0
0 0 297 7552 14924 8832 1163 0
0 0 0 1394 6221 5974 1147 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 891 6257 6345 979 0 0 0
0 916 9718 15175 6666 293 0 0
0 25 3461 8830 5687 293 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

* * * *(-----------------------------------------------)
* * VOICEBAND PERFORMANCE  parabolic decimation weighting
* *     sdr calculated using 256 baseband bins
* *     sdr calculated with equalization
* *     sdr ignores first 128 transient bins
* * input: -30 dB wrt 2.5 V
* * decimate 4 5 type=gendec number=32768 window=para taps=128
*     intdec=32 bbanddec=4 file=vb.cas filter=cascade
Decimation Parameters From File: " VB.CAS "
32768 points decimated to 256
* sdr 5 6 type=calsdr npoints=256 nskip=128 nfft=128 fftwin=unif
*     decwin=para fb=4000 fs=8000 taps=128 signal=1000
baseband frequency: 4000.00 Hz.; sampling frequency: 8000.00 Hz.
128 total fft bins
64 bins in baseband; signal on bin 17
3 harmonics in baseband
FFT RESULTS

Signal Strength : 1.9999918172406  Total Distortion : 7.3436389151260d-06
Total Noise     : 6.0304125187815d-06  Total Harmonics : 1.3132267903341d-06

Signal-to-Distortion (dB): 54.3512
Signal-to-Noise     (dB): 55.2068
Signal-to-Harmonic (dB): 61.8269

* * print out first 100 bits of bitstream
* dump node4 4 100
  DUMP file: "NODE4"
  no. values output = 100 node number = 4

* * print out the baseband signal
* dump node5 5 256
  DUMP file: "NODE5"
  no. values output = 256 node number = 5

* * print out the spectrum for the second set of 128 baseband points
* dump node6 6 128
  DUMP file: "NODE6"
  no. values output = 128 node number = 6

* * now simulate with difference equations for input
* * signal level of -20 dB wrt 2.5 V
* gen 1 1 type=sine freq=1000 amp=-20 dB wrt 2.5
  replacing old generator ?
* eq gain1=0.1 gain2=0.5 sat=1.5 delta1=2.5 delta2=0.25
* init node 1 0
* init node 2 0
* init node 3 0
* init node 4 1
* simulate

INTEGRATOR OUTPUT

Integrator # 1
Input max value = 0.250000  Input min value = -0.250000
Output max value = 0.461255  Output min value = -0.464121

Integrator # 2
Input max value = 0.461255  Input min value = -0.464121
Output max value = 0.352487  Output min value = -0.354863

********************************************************************
decimate 4 5 type=gendec number=32768 window=para taps=128
intdec=32 bbanddec=4 file=vb.cas filter=cascade
Decimation Parameters From File: "VB.CAS"
32768 points decimated to 256
sdr 5 6 type=calsdr npoints=256 nskip=128 nfft=128 fftwin=unif
decwin=para fb=4000 fs=8000 taps=128 signal=1000
baseband frequency: 4000.00 Hz.; sampling frequency: 8000.00 Hz.
128 total fft bins
64 bins in baseband; signal on bin 17
3 harmonics in baseband

FFT RESULTS

Signal Strength : 1.9999991705350 Total Distortion : 8.5826086590927d-07
Total Noise : 8.2024190071623d-07 Total Harmonics : 3.8018992284193d-08

Signal-to-Distortion (dB): 63.6741
Signal-to-Noise (dB): 63.8709
Signal-to-Harmonic (dB): 77.2103

* stop
APPENDIX B

SPICE Simulation Example
Class AB CMOS Operational Amplifier
This is a sample SPICE output file which simulates the first integrator for one clock period. This run has initial conditions

\[ x(k) = -1.0 \text{ V} \]
\[ y(k-1) = 0.1 \text{ V} \]
\[ p(k) = -2.5 \text{ V} \]

and corresponds to table point T(3,5,1)

```
******100687 ******* SPICE 2G.5 3/15/83 *******09:02:05*****

* SWITCH CAPACITOR INTEGRATOR #1 FOR DSM SIMULATION

**** INPUT LISTING  TEMPERATURE = 27.000 DEG C

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

* .WIDTH OUT=80
* * CLASS AB OPAMP FOR DSM SIMULATION
* * PLUS INPUT IS NODE 2, MINUS INPUT IS NODE 3, OUTPUT IS NODE 18
* *
* VDD 1 0 5V
* *
* M1 23 3 9 1 PCH L=2.0U W=34U AD=136P AS=136P
* M2 24 2 10 1 PCH L=2.0U W=34U AD=136P AS=136P
* M3 7 7 5 1 PCH L=2.0U W=34U AD=136P AS=136P
* M4 8 8 6 1 PCH L=2.0U W=34U AD=136P AS=136P
* M5 21 7 44 1 PCH L=2.0U W=68U AD=272P AS=272P
* M6 12 8 4 1 PCH L=2.0U W=68U AD=136P AS=136P
* *
* M7 9 9 5 0 NCH L=2.0U W=34U AD=136P AS=136P
* M8 10 10 6 0 NCH L=2.0U W=34U AD=136P AS=136P
* M9 22 9 4 0 NCH L=2.0U W=68U AD=136P AS=136P
* M10 11 10 44 0 NCH L=2.0U W=68U AD=136P AS=136P
* *
* M11 12 12 0 0 NCH L=2.4U W=80U AD=320P AS=320P
* M12 11 11 1 PCH L=2.4U W=80U AD=136P AS=136P
* *
* M13 7 23 0 0 NCH L=2.8U W=60U AD=240P AS=240P
* M14 8 24 0 0 NCH L=2.8U W=60U AD=136P AS=136P
* M15 9 13 1 1 PCH L=2.8U W=60U AD=136P AS=136P
* M16 10 13 1 1 PCH L=2.8U W=60U AD=136P AS=136P
* *
* M17 17 21 0 0 NCH L=2.4U W=40U AD=160P AS=160P
* M18 17 17 16 16 PCH L=2.8U W=42U AD=168P AS=168P
* M19 16 16 1 1 PCH L=2.4U W=40U AD=136P AS=136P
* *
* M20 14 14 0 0 NCH L=2.4U W=40U AD=136P AS=136P
* M21 15 15 14 0 NCH L=2.8U W=42U AD=136P AS=136P
* M22 15 22 1 1 PCH L=2.4U W=40U AD=136P AS=136P
* *
* M23 20 12 0 0 NCH L=2.4U W=80U AD=136P AS=136P
* M24 18 15 20 0 NCH L=2.8U W=84U AD=336P AS=336P
* M25 18 17 19 1 PCH L=2.8U W=84U AD=136P AS=136P
* M26 19 11 1 1 PCH L=2.4U W=80U AD=136P AS=136P
* *
* M27 13 13 1 1 PCH L=2.8U W=6U AD=24P AS=24P
* *
* M28 21 21 0 0 NCH L=2.4U W=80U AD=136P AS=136P
* M29 22 22 1 1 PCH L=2.4U W=80U AD=136P AS=136P
* M30 23 23 0 0 NCH L=2.8U W=60U AD=136P AS=136P
* M31 24 24 0 0 NCH L=2.8U W=60U AD=136P AS=136P
* *
* C4 4 0 0.03P
```
C5 5 0 0.03P
C6 6 0 0.03P
C7 7 0 0.03P
C8 8 0 0.03P
C9 9 0 0.03P
C10 10 0 0.03P
C11 11 0 0.03P
C12 12 0 0.03P
C13 13 0 0.03P
C14 14 0 0.03P
C15 15 0 0.03P
C16 16 0 0.03P
C17 17 0 0.03P
C18 18 0 0.03P
C19 19 0 0.03P
C20 20 0 0.03P
C21 21 0 0.03P
C22 22 0 0.03P
C23 23 0 0.03P
C24 24 0 0.03P
C44 44 0 0.03P

R1 13 0 450K

\text{NODESET } V(3)=2.498 \text{ } V(4)=2.7019 \text{ } V(5)=2.7022 \text{ } V(6)=2.7036 \text{ } V(7)=1.1105
+V(8)=1.1121 \text{ } V(9)=3.9012 \text{ } V(10)=3.9027 \text{ } V(11)=3.7771 \text{ } V(12)=0.9769
+V(13)=3.6402 \text{ } V(14)=0.9855 \text{ } V(15)=2.1021 \text{ } V(16)=3.7577 \text{ } V(17)=2.4950
+V(18)=2.6000 \text{ } V(19)=3.9763 \text{ } V(20)=0.9942 \text{ } V(21)=0.9781 \text{ } V(22)=3.7793
+V(23)=0.9475 \text{ } V(24)=0.9474 \text{ } V(44)=2.7022

\text{SWITCHES}

MS1N 110 51 101 0 NCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P
MS1P 110 52 101 1 PCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P

MS2N 101 53 2 0 NCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P
MS2P 101 54 2 1 PCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P

MS3N 102 53 2 0 NCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P
MS3P 102 54 2 1 PCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P

MS4N 102 51 3 0 NCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P
MS4P 102 52 3 1 PCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P

MS5N 111 51 103 0 NCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P
MS5P 111 52 103 1 PCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P

MS6N 103 53 2 0 NCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P
MS6P 103 54 2 1 PCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P

MS7N 104 53 2 0 NCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P
MS7P 104 54 2 1 PCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P

MS8N 104 51 3 0 NCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P
MS8P 104 52 3 1 PCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P

MS9N 18 51 105 0 NCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P
MS9P 18 52 105 1 PCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P

MS10N 105 53 2 0 NCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P
MS10P 105 54 2 1 PCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P

MS11N 106 51 2 0 NCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P
MS11P 106 52 2 1 PCH \text{ } W=5U \text{ } L=2U \text{ } AD=20P \text{ } AS=20P

NODES
V(3)=2.498 \text{ } V(4)=2.7019 \text{ } V(5)=2.7022 \text{ } V(6)=2.7036 \text{ } V(7)=1.1105
+V(8)=1.1121 \text{ } V(9)=3.9012 \text{ } V(10)=3.9027 \text{ } V(11)=3.7771 \text{ } V(12)=0.9769
+V(13)=3.6402 \text{ } V(14)=0.9855 \text{ } V(15)=2.1021 \text{ } V(16)=3.7577 \text{ } V(17)=2.4950
+V(18)=2.6000 \text{ } V(19)=3.9763 \text{ } V(20)=0.9942 \text{ } V(21)=0.9781 \text{ } V(22)=3.7793
+V(23)=0.9475 \text{ } V(24)=0.9474 \text{ } V(44)=2.7022
* CAPACITORS
CF 3 18 3.0P
CL 18 0 3.0P
C1 101 102 0.30P
C1S 101 0 0.30P
C2 103 104 0.30P
C2S 103 0 0.30P
C3 105 106 1.00P
C3S 105 0 1.00P

* CLOCKS

VPHI 51 0 PULSE(5V 0V 4NS 4NS 4NS 102NS 200NS)
VPHI1B 52 0 PULSE(0V 5V 4NS 4NS 4NS 102NS 200NS)
VPHI2 53 0 PULSE(0V 5V 10NS 4NS 4NS 90NS 200NS)
VPHI2B 54 0 PULSE(5V 0V 10NS 4NS 4NS 90NS 200NS)

.OPTIONS CHGTO=1.0F RELTOL=0.0001
.OPTIONS NUMDGT=5 NOMOD LIMPTS=501 ITL1=200 ITL5=0

* FORCE INITIAL VALUE

LF 99 18 1MEG
VLAST 99 3 DC 0.1000V

* INPUTS

VAIN 110 2 DC -1.0000V
VFB 111 2 DC -2.5000V
VAGND 2 0 DC 2.5000V

.TRAN INS 204NS
.PRINT TRAN V(18,2) V(51) V(53) V(110,2) V(111,2)

* MODELS FOR MCNC 1.25 MICRON CMOS PROCESS

.MODEL NCH NMOS LEVEL=2 VTO=+0.71 GAMMA=0.29 CGSO=2.89E-10
+ CGDO=2.89E-10 CJ=3.27E-4 MJ=0.4 TOX=225E-10 NSUB=3.5E16
+ XJ=0.2E-5 LD=0 UO=411 UEXP=0 KF=8.5E-28

.MODEL PCH PMOS LEVEL=2 VTO=-0.76 GAMMA=0.6 CGSO=3.35E-10
+ CGDO=3.35E-10 CJ=4.75E-4 MJ=0.4 TOX=225E-10 NSUB=1.6E16
+ XJ=0.2E-5 LD=0 UO=139 UEXP=0 KF=5E-30

.END

******100687*******SPICE 2G.63/15/83**********09:02:05*****

* SWITCH CAPACITOR INTEGRATOR #1 FOR DSM SIMULATION

**** INITIAL TRANSIENT SOLUTION TEMPERATURE = 27.000 DEG C

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

NODE VOLTAGE NODE VOLTAGE NODE VOLTAGE NODE VOLTAGE

( 1) 5.0000 ( 2) 2.5000 ( 3) 2.4982 ( 4) 2.7019
( 5) 2.7022 ( 6) 2.7036 ( 7) 1.1105 ( 8) 1.1121
VOLTAGE SOURCE CURRENTS

NAME CURRENT

VDD -4.949D-04
VPHII 0.000D+00
VPHIIB 0.000D+00
VPHI2 0.000D+00
VPHI2B 0.000D+00
VLAST -3.058D-14
VAIN 1.832D-11
VFB 2.080D-11
VAGND 3.010D-11

TOTAL POWER DISSIPATION 2.470D-03 WATTS

* SWITCH CAPACITOR INTEGRATOR #1 FOR DSM SIMULATION

* OPERATING POINT INFORMATION TEMPERATURE = 27.000 DEG C

MOSFETS

M1 M2 M3 M4 M5 M6 M7
MODEL PCH PCH PCH PCH PCH PCH PCH
ID -4.00E-05 -4.00E-05 -4.02E-05 -4.02E-05 -8.10E-05 -8.03E-05 4.02E-05
VGS -1.403 -1.403 -1.592 -1.592 -1.592 -1.590 1.199
VDS -2.954 -2.955 -1.592 -1.591 -1.724 -1.725 1.199
VBS 1.099 1.097 2.298 2.296 2.298 2.298 -2.702

M8 M9 M10 M11 M12 M13 M14
MODEL NCH NCH NCH NCH PCH NCH NCH
### Table 1: Simulation Results

<table>
<thead>
<tr>
<th>ID</th>
<th>VGS</th>
<th>VDS</th>
<th>VBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.02E-05</td>
<td>8.03E-05</td>
<td>8.10E-05</td>
<td>8.03E-05</td>
</tr>
<tr>
<td>1.199</td>
<td>1.199</td>
<td>1.201</td>
<td>0.977</td>
</tr>
<tr>
<td>1.199</td>
<td>1.077</td>
<td>1.075</td>
<td>0.977</td>
</tr>
<tr>
<td>-2.704</td>
<td>-2.702</td>
<td>-2.702</td>
<td>0.000</td>
</tr>
</tbody>
</table>

**Model M15-M21**

<table>
<thead>
<tr>
<th>ID</th>
<th>VGS</th>
<th>VDS</th>
<th>VBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.03E-05</td>
<td>4.20E-05</td>
<td>-4.20E-05</td>
<td>-4.20E-05</td>
</tr>
<tr>
<td>-1.360</td>
<td>-1.360</td>
<td>0.977</td>
<td>-1.273</td>
</tr>
<tr>
<td>-1.099</td>
<td>-1.097</td>
<td>2.495</td>
<td>-1.233</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

**Model M22-M28**

<table>
<thead>
<tr>
<th>ID</th>
<th>VGS</th>
<th>VDS</th>
<th>VBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4.27E-05</td>
<td>8.03E-05</td>
<td>8.03E-05</td>
<td>-8.03E-05</td>
</tr>
<tr>
<td>-1.221</td>
<td>0.977</td>
<td>1.107</td>
<td>-1.484</td>
</tr>
<tr>
<td>-2.898</td>
<td>0.995</td>
<td>1.603</td>
<td>-1.381</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

**Model M29-M32**

<table>
<thead>
<tr>
<th>ID</th>
<th>VGS</th>
<th>VDS</th>
<th>VBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.03E-05</td>
<td>4.00E-05</td>
<td>4.00E-05</td>
<td>-1.46E-11</td>
</tr>
<tr>
<td>-1.221</td>
<td>0.947</td>
<td>0.947</td>
<td>3.500</td>
</tr>
<tr>
<td>-1.221</td>
<td>0.947</td>
<td>0.947</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>-1.500</td>
</tr>
</tbody>
</table>

**Model MS3N-MS6N**

<table>
<thead>
<tr>
<th>ID</th>
<th>VGS</th>
<th>VDS</th>
<th>VBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.500</td>
<td>2.500</td>
<td>2.500</td>
<td>-2.498</td>
</tr>
<tr>
<td>-0.002</td>
<td>-0.002</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>-2.500</td>
<td>2.500</td>
<td>-2.498</td>
<td>2.502</td>
</tr>
</tbody>
</table>

**Model MS6P-MS9P**

<table>
<thead>
<tr>
<th>ID</th>
<th>VGS</th>
<th>VDS</th>
<th>VBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.500</td>
<td>-2.500</td>
<td>2.500</td>
<td>2.500</td>
</tr>
<tr>
<td>-2.500</td>
<td>-0.002</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>2.500</td>
<td>-2.500</td>
<td>2.500</td>
<td>-2.498</td>
</tr>
</tbody>
</table>

**Model MS10N-MS12P**

<table>
<thead>
<tr>
<th>ID</th>
<th>VGS</th>
<th>VDS</th>
<th>VBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.500</td>
<td>2.500</td>
<td>2.500</td>
<td>-2.500</td>
</tr>
<tr>
<td>0.098</td>
<td>0.098</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>-2.500</td>
<td>2.500</td>
<td>-2.500</td>
<td>2.500</td>
</tr>
</tbody>
</table>

**Switch Capacitor Integrator #1 for DSM Simulation**

* transient analysis  temperature = 27.000 deg C

```plaintext
* Switch Capacitor Integrator  #1 for DSM Simulation

** Transient Analysis ** Temperature = 27.000 Deg C
```

**Time** V(18,2) V(51) V(53) V(110,2) V(111,2)

<table>
<thead>
<tr>
<th>Time</th>
<th>V(18,2)</th>
<th>V(51)</th>
<th>V(53)</th>
<th>V(110,2)</th>
<th>V(111,2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000E+00</td>
<td>9.8162E-02</td>
<td>5.0000E+00</td>
<td>0.0000E+00</td>
<td>-1.0000E+00</td>
<td>-2.5000E+00</td>
</tr>
<tr>
<td>1.000E-09</td>
<td>9.8163E-02</td>
<td>5.0000E+00</td>
<td>0.0000E+00</td>
<td>-1.0000E+00</td>
<td>-2.5000E+00</td>
</tr>
<tr>
<td>2.000E-09</td>
<td>9.8163E-02</td>
<td>5.0000E+00</td>
<td>0.0000E+00</td>
<td>-1.0000E+00</td>
<td>-2.5000E+00</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>2.010E-07</td>
<td>4.4277E-01</td>
<td>5.0000E+00</td>
<td>0.0000E+00</td>
<td>-1.0000E+00</td>
<td>-2.5000E+00</td>
</tr>
<tr>
<td>2.020E-07</td>
<td>4.4302E-01</td>
<td>5.0000E+00</td>
<td>0.0000E+00</td>
<td>-1.0000E+00</td>
<td>-2.5000E+00</td>
</tr>
<tr>
<td>2.030E-07</td>
<td>4.4325E-01</td>
<td>5.0000E+00</td>
<td>0.0000E+00</td>
<td>-1.0000E+00</td>
<td>-2.5000E+00</td>
</tr>
<tr>
<td>2.040E-07</td>
<td>4.4348E-01</td>
<td>5.0000E+00</td>
<td>0.0000E+00</td>
<td>-1.0000E+00</td>
<td>-2.5000E+00</td>
</tr>
</tbody>
</table>

**JOB CONCLUDED**

**TOTAL JOB TIME** 104.03
APPENDIX C

Linear Interpolation Formula Derivation
INTERPOLATION FORMULA

Data Points:

- \( X_1 \) values: \( s_1, x_1, f_1 \)
- \( X_2 \) values: \( s_2, x_2, f_2 \)
- \( Y \) values: \( p_{t1}, p_{t2}, p_{t3}, p_{t4} \)

Determine percentages:

\[
\Delta A = \frac{x_1 - s_1}{f_1 - s_1} \quad \text{(D.1)}
\]
\[
\Delta B = \frac{x_2 - s_2}{f_2 - s_2} \quad \text{(D.2)}
\]

Find 2 new data points in \( X_2 \) direction:

\[
aa = \Delta B(p_{t3} - p_{t1}) + p_{t1} \quad \text{(D.3)}
\]
\[
bb = \Delta B(p_{t4} - p_{t2}) + p_{t2} \quad \text{(D.4)}
\]

Find data point in \( X_1 \) direction:

\[
Y = \Delta A(bb - aa) + aa \quad \text{(D.5)}
\]

Substitute D.3 and D.4 into D.5:

\[
Y = \Delta A \left\{ \left[ \Delta B(p_{t4} - p_{t2}) + p_{t2} \right] - \left[ \Delta B(p_{t3} - p_{t1}) + p_{t1} \right] \right\} + \Delta B(p_{t3} - p_{t1}) + p_{t1} \quad \text{(D.6)}
\]

Rearrange terms to obtain the interpolation formula along with D.1 and D.2:

\[
Y = p_{t1} + \Delta A(p_{t2} - p_{t1}) + \Delta B(p_{t3} - p_{t1}) + \Delta A \Delta B(p_{t4} + p_{t1} - p_{t3} - p_{t2}) \quad \text{(D.7)}
\]
APPENDIX D

Selected ZSIM Source Code
POLATE

PURPOSE:
This subroutine interpolates table data.

METHOD:
Newton interpolation routine.
See p.23 Carnahan, Luther and Wilkes

PARAMETERS:
dim : number of variables used in interpolation (max=3)
fbindx : functional block code
blk : block i of fbindx type
ptable: pointer to table values
tables : table of functional block values
cout : output of interpolation
x1 : signal number 1
x2 : signal number 2
x3 : signal number 3 (discrete)

subroutine POLATE(dim,fbindx,blk,ptable,tables,out,x1,x2,x3)
integer xl,x2beg,x3beg,dim,knt,fbindx,beg,blk,s1
integer sta1,sta2,sta3,sta4,g1,g2,g3,g4
common /parm/icons/parmax,blk,mxblk,mxdim,mtel,table,table,table,table,table,table,table,table,table,table,table,table,table
real pt1,pt2,pt3,pt4,s1,s2,f1,f2,delaa,delatc
real x1,x2,x3,fnewt
real tables(mxtable,out)
real al(20),bl(20),table(20,20)
integer ptable(mxtion,mxblk,mxdim3)
character*20 str
logical kr,ftn,prt,ofile,rfile
common /freerd/ nwrite,ntype,aval,ival,str,ich,kr,fln,prt,ofile,rfile
common/ table/ exceed
integer exceed

ONE DIMENSIONAL INTERPOLATION

10 beg=ptable(fbindx,blk,2)
ideg=ptable(fbindx,blk,3)
x2beg=ptable(fbindx,blk,4)
if(ideg.ge.x2beg)ideg=x2beg+1
if(x1.lt.tables(beg) or x1.gt.tables(beg-1+x2beg))then
  write(ntype,"'" table values exceeded ")
  exceed=exceed+1
  if(exceed.gt.100)goto 9900
  goto 1000
endif

LINEAR INTERPOLATION

if(ideg.eq.1)then
if(x1.eq.tables(beg)) then
  stal = 1
  finl = 2
else
  do 30 i = 1, ptable(fbindx, blk, 4)
  if(x1.le.tables(beg+(i-1))) goto 40
  continue
30
  stal = i-1
  finl = i
  endif
  st1 = ptable(fbindx, blk, 4)
  if(tables(beg+finl-1)-tables(beg+stal-1).eq.0) then
    out = 0.
  else
    out = (x1-tables(beg+stal-1))
    out = out/(tables(beg+finl-1)-tables(beg+stal-1))
  endif
  out = out "+ (tables(beg+st1+finl-1)-tables(beg+stal+st1-1))
  out = out + tables(beg+st1+stal-1)
  return
else
  c
  POLYNOMIAL INTERPOLATION
  c
  if(ptable(fbindx, blk,4).lt.6) then
    mx = ptable(fbindx, blk,4)-1
  else
    mx = 5
  endif
  do 20 i = 1, ptable(fbindx, blk,4)
    a1(i) = tables(ptable(rbindx, blk,2)+(i-1))
    b1(i) = tables(ptable(fbidx, blk,2)+ptable(fbidx, blk,4)+(i-1))
  continue
20
  call DTABLE(a1,b1,table,ptable(fbidx, blk,4),mx,tru,b,knt)
  if(trubl.ne.O.O) goto 1000
  out = fnewt(a1,b1,table,ptable(fbidx, blk,4),mx,ideg,x1,tru,bl,knt)
  if(tru.bl.ne.O.O) goto 1000
  return
endif
  c
  MULTI-DIMENSIONAL INTERPOLATION
  c
  x1beg = ptable(fbidx, blk,2)
  x2beg = x1beg + ptable(fbidx, blk,4)
  x3beg = x2beg + ptable(fbidx, blk,5)
  c
  CHECK FOR TABLE OVERFLOW ERROR
  c
  if(x1.lt.(tables(x1beg)-0.0001).or.x1.gt.(tables(x2beg-1)+0.0001)) then
    write(ntype,"') ' table values exceeded '?
    exceed = exceed + 1
    if(exceed.gt.100) goto 9900
    goto 1000
  endif
  if(x2.lt.(tables(x2beg)-0.0001).or.x2.gt.(tables(x3beg-1)+0.0001)) then
    write(ntype,"') ' table values exceeded ?'
    exceed = exceed + 1
    if(exceed.gt.100) goto 9900
    goto 1000
  endif
  if(x1.lt.tables(x1beg)) x1 = tables(x1beg)
  if(x1.gt.tables(x2beg-1)) x1 = tables(x2beg-1)
  if(x2.lt.tables(x2beg)) x2 = tables(x2beg)
  if(x2.gt.tables(x3beg-1)) x2 = tables(x3beg-1)
FIND BEGINNING OF OUTPUT TABLE

\[
\text{beg} = \text{ptable}(\text{fbindx, blk, 2})
\]

\[
\text{do 55 i = 4, mxdim3}
\]

\[
\text{beg} = \text{beg} + \text{ptable}(\text{fbindx, blk, i})
\]

\[55\] continue

\[
\text{if(dim .eq. 3) then}
\]

\[
\text{do 58 i = 1, ptable(\text{fbindx, blk, 6})}
\]

\[
\text{if(x3 .eq. \text{tables(x3beg + (i-1))})}
\]

\[59\] goto

\[58\] continue

\[
\text{write(\text{ntype, ""}) ?? interpolation error -> non-discrete 3rd dim ??}
\]

\[57\] return

\[
\text{beg} = \text{beg} + \text{ptable(\text{rbindx, blk, 4})} \cdot \text{ptable(\text{fbindx, blk, 5})} \cdot (i-1)
\]

\[56\] endif

c

c FIND INTERPOLATION INPUT VALUES FOR X1
c

\[
\text{if(x1 .eq. \text{tables(\text{ptable(\text{fbindx, blk, 2})})})}
\]

\[
\text{sta1 = 1}
\]

\[
\text{fin1 = 2}
\]

\[
\text{else}
\]

\[
\text{do 90 i = 1, ptable(\text{fbindx, blk, 4})}
\]

\[
\text{if(x1 .le. \text{tables(\text{ptable(\text{fbindx, blk, 2})} + (i-1))})}
\]

\[90\] goto

\[100\] continue

\[
\text{sta1 = i-1}
\]

\[
\text{fin1 = i}
\]

\[90\] continue

\[
\text{endif}
\]

\[
\text{s1 = \text{tables(\text{ptable(\text{fbindx, blk, 2})} + \text{sta1}-1)}
\]

\[
\text{f1 = \text{tables(\text{ptable(\text{fbindx, blk, 2})} + \text{fin1}-1)}
\]

\[c\]

c FIND INTERPOLATION INPUT VALUES FOR X2
c

\[
\text{if(x2 .eq. \text{tables(\text{x2beg})})}
\]

\[
\text{sta2 = 1}
\]

\[
\text{fin2 = 2}
\]

\[
\text{else}
\]

\[
\text{do 60 i = 1, ptable(\text{fbindx, blk, 5})}
\]

\[
\text{if(x2 .le. \text{tables(\text{x2beg + (i-1)})})}
\]

\[70\] goto

\[60\] continue

\[
\text{sta2 = i-1}
\]

\[
\text{fin2 = i}
\]

\[60\] continue

\[
\text{endif}
\]

\[
\text{s2 = \text{tables(x2beg + sta2-1)}
\]

\[
\text{f2 = \text{tables(x2beg + fin2-1)}
\]

\[c\]

c FIND THE FOUR (4) INTERPOLATING POINTS IN THE OUTPUT TABLE
c

\[
\text{g1 = beg + sta1-1 + ptable(\text{fbindx, blk, 4})} \cdot (\text{sta2-1})
\]

\[
\text{g2 = beg + fin1-1 + ptable(\text{fbindx, blk, 4})} \cdot (\text{sta2-1})
\]

\[
\text{g3 = beg + sta1-1 + ptable(\text{fbindx, blk, 4})} \cdot (\text{fin2-1})
\]

\[
\text{g4 = beg + fin1-1 + ptable(\text{fbindx, blk, 4})} \cdot (\text{fin2-1})
\]

\[
\text{pt1 = \text{table}(g1)}
\]

\[
\text{pt2 = \text{table}(g2)}
\]

\[
\text{pt3 = \text{table}(g3)}
\]

\[
\text{pt4 = \text{table}(g4)}
\]

\[c\]

c FORMULA FOR INTERPOLATING
c

\[
\text{if(f1 .eq. 0) then}
\]

\[
\text{deltaa = 0}
\]

\[
\text{else}
\]

\[
\text{deltaa = (x1-s1)/(f1-s1)}
\]

\[c\]

c END
deltac = 0.
else
   deltax = (x2 - s2)/(f2 - s2)
endif

out = pt1 + deltaa*(pt2 - pt1) + deltax*(pt3 - pt1)
out = out + deltaa + deltax

CGTB_START
C This determines which table points are used.
   ideep = 1
   if(ptable(fbindx, blk, 6) .eq. 2)
      if(nint(x3).eq.1) ideep = 2
      else
         if(nint(x3).eq.0) ideep = 2
         if(nint(x3).eq.1) ideep = 3
      endif
   endif

   hist(blk, sta1, sta2, ideep) = 1 + hist(blk, sta1, sta2, ideep)
   hist(blk, sta1, fin2, ideep) = 1 + hist(blk, sta1, fin2, ideep)
   hist(blk, fin1, sta2, ideep) = 1 + hist(blk, fin1, sta2, ideep)
   hist(blk, fin1, fin2, ideep) = 1 + hist(blk, fin1, fin2, ideep)

CGTB_END

1000 return
9900 write(ntype, *) " POLATE - Too many function values exceeded ?"
write(nwrite, *) " POLATE - Too many function values exceeded ?"
write(nwrite, *) " POLATE x1=',x1,' x2=',x2,' x3=',x3
stop
end

SDRSUB

purpose: To calculate signal-to-total distortion ratios,
signal-to-white noise ratios, and signal-to-harmonic
distortion ratios for a fourier sequence of numbers

subroutine sdrsub(xx,equal,sdr,snr,sthd,n,spread,iw,fb,sampfq,
   + period,sigfq,prt,nwrite)
CGTB_START
   logical prt
   integer iw
   real fb, period, sigfq, sampfq
CGTB_END

   real xx(n), equal(n), sdr, snr, sthd
   double precision spec(32768), tosp
   integer n, nwrite, spread
   logical tot3
   common/ fouriel, snoise, sigtotno, totno2, totno3, snr1, snr2, snr3, fftyes

   determine baseband bins
CGTB_START
CGTB_START
   if(iw.eq.0) sampfq = 1.0/period
   iband = nint(n*fb/sampfq)
   iict = nint(n*sigfq/sampfq+1)
   if(iict.eq.1) then
      write(ntype,"') ' no sdr calculation completed for D.C. Signal'
      if(prt)
         write(nwrite,"') ' no sdr calculation completed for D.C. Signal'
      return
   endif
   write(nwrite,"') ' baseband frequency ',fb,' Hz.; sampling',
   ' frequency ',sampfq,' Hz.'
   write(nwrite,"') n, 'total ft bins'

CGTB_START
CGTB_START
write(nwrite,*)iband,'bins in baseband; signal on bin',iict

get energy and equalize for $\sin(x)/x$ correction

totp=0.0

do 900 j=1,iband+1
   spec(j)=xx(2*j-1)*xx(2*j-1)+xx(2*j)*xx(2*j)
   spec(j)=spec(j)*equal(j)*equal(j)
   totp=totp+spec(j)
900 continue

CGTB_END

totno1=2*spec(1)
totno2=2*spec(1)
totno3=0.0

iharm=nint(iict/(iband/(iict-1)))*1.0

write(nwrite,*)iharm,'harmonics in baseband'
icount=1

totno1: total distortion
totno2: white noise
totno3: harmonic noise

iharm : number of harmonics

icount: present harmonic

do 449 j=2,iband+1
   if(j.ge.iict-spread and j.le.iict+spread)then
      sig=sig+2*spec(j)
   else
      totno1=totno1+2*spec(j)
      tot3=.false.
      itmp=icount*(iict-1)+iict
      if(itmp+spread+1.gt.iband+1)then
         tot3=.true.
      endif
      if(tot3)then
         totno3=totno3+2*spec(j)
      else
         totno2=totno2+2*spec(j)
      endif
   endif
449 continue

c take out white noise from signal

c ave=(spec(iict-spread-1)+spec(iict+spread+1))/2.0

c sig=sig-2*(2*spread+1)*ave

c totno1=totno1+2*[2*spread+1]*ave-2*[spec(1)+spec(2)]
totno2=totno2+2*[2*spread+1]*ave-2*[spec(1)+spec(2)]


c separate harmonic noise from white noise

do 67 ii=1,iharm
   itmp=ii*(iict-1)+iict
   if(itmp+spread+1.gt.iband+1)then
      ave=(spec(itmp-spread-1)+spec(itmp+spread-2))/2.0
   else
      ave=(spec(itmp-spread-1)+spec(itmp+spread+1))/2.0
   endif

CGTB_START
if(spec(itmp).gt.ave)then
   totno2=totno2+2*[2*spread+1]*ave
   totno3=totno3-2*[2*spread+1]*ave
else
   do 3 i=-1.0*spread,spread
      totno2=totno2+spec(itmp+i)
   totno3=totno3-spec(itmp+i)
3 continue
3 continue
endif
CGTB_END
67 continue
CGTB_START
if(totno3.lt.0.0)then
    totno3=0.0
    totno2=totno2-totno3
endif
CGTB_END

c compute snr ratios in dB

c if(totno1.lt.1.0e-15)then
    sdr=sig/1.0e-15
else
    sdr=sig/totno1
endif
if(sdr.lt.1.0e-10)then
    sdr=-100.0
else
    sdr=10.0*alog10(sdr)
endif
if(totno2.lt.1.0e-15)then
    snr=sig/1.0e-15
else
    snr=sig/totno2
endif
if(snr.lt.1.0e-10)then
    snr=-100.0
else
    snr=10.0*alog10(snr)
endif
if(totno3.gt.1.0e-15)then
    sthd=sig/1.0e-15
else
    sthd=sig/totno3
endif
if(sthd.lt.1.0e-10)then
    sthd=-100.0
else
    sthd=10.0*alog10(sthd)
endif
write(nwrite,*)
write(nwrite,312)
312 format( ' ',78('-') )
write(nwrite,*) ' FFT RESULTS'
write(nwrite,*)
write(nwrite,) ' Signal Strength : ',sig/totsp,
+ ' Total Distortion  : ',totno1/totsp
write(nwrite,*)
write(nwrite,*) ' Total Noise     : ',totno2/totsp,
+ ' Total Harmonics : ',totno3/totsp
write(nwrite,*)
write(nwrite,) ' Signal-to-Distortion (dB) : ',sdr
write(nwrite,) ' Signal-to-Noise    (dB) : ',snr
write(nwrite,) ' Signal-to-Harmonic (dB) : ',sthd
write(nwrite,312)
return
end

ZANA

PURPOSE:
This subroutine simulates the circuit.
PARAMETERS:
- x: storage array for circuit node values
- gen: storage array for generator
- scint: storage array for integrator
- quant: storage array for quantizer
- ptable: pointer array for table look-up
- tables: array containing functional block values
- ndpres: acknowledges presence of circuit nodes
- node: contains present node values
- nodep: contains previous node values

SPECIAL VARIABLES:
- stocnt: initialize to 1; pseudo-cycle value for node storage
- ix: initial seed for pseudo-random number (ix < 0)

subroutine ZAN(x,gen,scint,quant,ptable,tables,ndpres,
    + node,nodep)

integer cycle,nocyc,block,tsadd,mxnode,circuit,nodes,ix
integer mgen,pgen,mstocnt,mscint,mquant,pquant
common/misc/ix
common/blkmax/mgen,pgen,mscint,mscint,mquant,pquant
common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
common/funcblk/block,period,nocyc,tsadd,pi,freqpt,circuit
real tables(mxtele),x(mxicyc,mxnode),gen(mgen,pgen)
real scint(mscint,mscint),quant(mquant,pquant)
real node(mxnode),nodep(mxnode)
integer ptable(mxtype,mxblk,mxdim3),stocnt
common/sto/stocnt
logical ndpres(mxnode)
logical cpypar,cpynod,over,clear
common/commnd/cpypar,cpynod,over,clear
character*20 str
logical kf,fln,prt,oflle,rflle
common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fln,prt,oflle,rflle
integer topol
common/aimtyp/topol
call rread

if(str(1:4).eq.'HELP')then
  if(topol.eq.1)then
    write(ntype,50)
  else if(topol.eq.0)then
    write(ntype,55)
  if(prt)write(nwrite,50)
format(
  + '################ TABLE SIMULATION ################',/.
  + ' Usage:/',/.
  + ' SIM HELP:',/.
  + ' This Message:',/.
  + ' SIM ',/.
  + ' Simulate a circuit described by tables',/.
  + '#####################################################################'
else if(topol.eq.0)then
  write(ntype,55)
  if(prt)write(nwrite,55)
format(
  + '################ DIFFERENCE EQUATION SIMULATION ################',/.
  + ' Usage:/',/.
  + ' DES HELP:',/.
  + ' This Message:',/.
  + ' DES ',/.
  + ' Simulate a circuit described by',/.
  + ' difference equations',/.
  + '#####################################################################'
endif
return
endif

c DETERMINE NUMBER OF NODES IN THE CIRCUIT
c
if(circut.eq.0)then
nodes=2
else if(circut.eq.1)then
nodes=3
else if(circut.eq.2)then
nodes=4
endif
c
PERFORM THE SIMULATION
c
stocnt=1
ix=-7841
2 do 100 cycle=1,nocyc
c
SIMULATE FIRST ORDER DELTA-SIGMA
c
c
if(circut.eq.1)then
  call AGEN(1,node,cycle,gen,f)
if(topol.eq.0)then
  call DESIM(cycle,node,nodep,nodes)
elseif(topol.eq.1)then
  call ASCINT(1,cycle,node,nodep,scint,ptable,tables)
call AQUANT(1,node,nodep,cycle,quant)
endif

c
SIMULATE SECOND ORDER DELTA-SIGMA
c
c
else if(circut.eq.2)then
  call AGEN(1,Dode,cycle,gen,r)
if(topol.eq.0)then
  call DESIM(cycle,node,nodep,nodes)
elseif(topol.eq.1)then
  call ASCINT(1,cycle,node,nodep,scint,ptable,tables)
call ASCINT(2,cycle,node,nodep,scint,ptable,tables)
call AQUANT(1,node,nodep,cycle,quant)
endif

c
THIS IS TO TEST THE NOISE GENERATOR AND/OR TRI-LEVEL COMPARATOR
c
c
else if(circut.eq.0)then
  call AGEN(1,node,cycle,gen,f)
call AQUANT(1,node,nodep,cycle,quant)
endif
c
STORE NODE VALUES
c
if(cycle.gt.0)then
  do 99 ii=1,nodes
    x(cycle,ii)=node(ii)
  99 continue
  endif

ACKNOWLEDGE PRESENCE OF NODES
c
ndpres(1)=.true.
npres(2)=.true.
npres(3)=.true.
if(circut.eq.2)ndpres(4)=.true
if(circut.eq.0)ndpres(3)=.false

C SET EACH PREVIOUS NODE VALUE TO ZERO AFTER COMPLETION OF SIMULATION
do 300 i=1,nodes
  nodep(i)=0.0
300 continue
return
end

ZCMD

PURPOSE:
This subroutine controls the command sequence

COMMANDS:
simulate: perform the s-domain simulation
gen: define generator input
scint: define integrator input
quant: define quantizer input
table: define table to describe functional blocks
hist: set table histogram flag
print: set printing features
noise: define gaussian noise
circuit: define circuit type
env: environment; define simulation features
clear: erase all parameters to default
pause: stop to perform printing features
stop: end session
dump: dump contents of stored node values
init: initialize quantities
decimate: perform a decimation
sdr: do Signal-to-Distortion ratio calculation

PARAMETERS:
x: storage array for node values
gen: informational array of generators
scint: informational array of integrators
quant: informational array of quantizers
ptable: pointer to table values
table: vector of functional block values
genfig: acknowledges presence of generators
intfig: acknowledges presence of integrators
qntfig: acknowledges presence of quantizers
ptnode: should node value be stored
ndpres: acknowledges presence of node

SUBROUTINES USED:
input, fread, gentor, intgtr, compar, init, sdec
nodset, zana, envirn, noiset, tabldr, dsmsdr

subroutine ZCMD(x,gen,scint,quant,ptable,tables,genfig,intfig, +
qntfig,ptnode,ndpres,node,nodep)
logical kf, fn, pr, ofile, rfile
common /freerdb/nwrite, ntype, aval, ival, str, ich, kf, fn, pr, ofile, rfile
integer topol
common/simtyp/topol

CGTB_START
integer hist(5, 50, 50, 3)
logical phist
common/histgm/phist, hist

CGTB_END

C READ A LINE OF INPUT
C
10 call input(***, nerr, 3)
call fread
if(fn) goto 10

C PROCESS INPUT ACCORDING TO FIRST FIELD OF INPUT LINE
C
if(str(1:3).eq. 'SIM'.or. str(1:3).eq. 'DES') then
if (str(1:3).eq. 'DES') topol = 0
if (str(1:3).eq. 'SIM') topol = 1
call ZANA(x, gen, scint, quant, ptable, tables, ndpres, node, nodep, bits)
goto 1000
else if (str(1:4).eq. 'DUMP') then
idump = 9
call ZDUMP(x, mxycy, mxnode, idump)
else if (str(1:4).eq. 'HELP'.or. str(1:1).eq. '?') then
call ZHELP
else if (str(1:4).eq. 'INIT') then
call INIT(nodep)
else if (str(1:2).eq. 'EQ') then
call ZEQUAT
else if (str(1:3).eq. 'DEC') then
call ZDEC(x)
else if (str(1:3).eq. 'SDR') then
call ZSDR(x)
else if (str(1:3).eq. 'GEN') then
call GENTOR(gen, genflg)
else if (str(1:3).eq. 'SCI') then
call INTGTR(scint, intflg, ptable)
else if (str(1:3).eq. 'QUA') then
call COMPAR(quant, qntflg, ptable)
else if (str(1:5).eq. 'TABLE') then
call TABLRD(scint, quant, ptable, tables)

CGTB_START
else if (str(1:4).eq. 'HIST') then
phist = true.

CGTB_END
else if (str(1:5).eq. 'PRINT') then
call fread
else if (str(1:3).eq. 'NOD') then
call NODSET(ptnode)
else if (str(1:3).eq. 'PAR') then
copy par = true.
call fread
else if (str(1:3).eq. 'OFF') copy par = false.
else
write(ntype, *) 'argument expected for "print" '
endif
else if (str(1:5).eq. 'NOISE') then
call NOISE
else if (str(1:3).eq. 'ENV') then
call ENVIRN
else if (str(1:3).eq. 'CIR') then
call fread
if(str(1:4).eq.'HELP')then
  write(ntype,50)
  if(prt)write(nwrite,50)
endif

format(
+ '#################################################################### CIRCUIT ####################################################################',/,
+ ' Usage:',/,
+ ' CIRCUIT HELP',/,
+ ' : This Message',/,
+ ' CIRCUIT -= n ',/,
+ ' : Set order of modulator to "n"',/,
+ ' ###############################################################################################################################################')
else
  circuit=ival
endif
else if(str(1:5).eq.'CLEAR')then
  call fread
  if(str(1:4).eq.'HELP')then
    write(ntype,52)
    if(prt)write(nwrite,52)
  endif
format(
+ '################################# CLEAR ###########################################',/,
+ ' Usage:',/,
+ ' CLEAR HELP',/,
+ ' : This Message',/,
+ ' CLEAR ',/,
+ ' : Set everything to default value',/,
+ ' ###############################################################################################################################################')
else
  clear=.true.
  goto 1000
endif
else if(str(1:5).eq.'STOP')then
  call fread
  if(str(1:4).eq.'HELP')then
    write(ntype,54)
    if(prt)write(nwrite,54)
  endif
format(
+ '################################################################################ STOP ################################################################################',/,
+ ' Usage:',/,
+ ' STOP HELP',/,
+ ' : This Message',/,
+ ' STOP ',/,
+ ' : End ZSIM session',/,
+ ' ###############################################################################################################################################')
else
  over=.true.
  goto 1000
endif
else if(str(1:5).eq.'PAUSE')then
  call fread
  if(str(1:4).eq.'HELP')then
    write(ntype,56)
    if(prt)write(nwrite,56)
  endif
format(
+ '######################################################################### PAUSE ##########################################################################',/,
+ ' Usage:',/,
+ ' PAUSE HELP',/,
+ ' : This Message',/,
+ ' PAUSE ',/,
+ ' : Stop simulation to perform printing features',/,
+ ' ###############################################################################################################################################')
else
  goto 1000
endif
elseif(str(1:1).eq.'"')then
  jj=1
else
  write(ntype,2)str(1:10)
  if(prt)write(nwrite,2)str(1:10)
  format('? unrecognisable command?',a10)
  endif
  goto 10

1000 return
end