

A PROPOSAL FOR AN ANALOG CMOS
MEDIAN FILTER SYSTEM BASED
ON NEURAL NETWORK
ARCHITECTURAL
PRINCIPLES

By

JOHN P. WAGNON

Bachelor of Science

Oklahoma State University


Stillwater, Oklahoma

1988

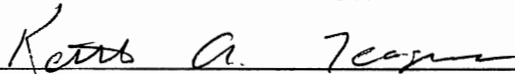
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 1992

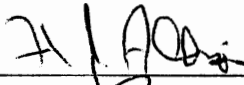
A PROPOSAL FOR AN ANALOG CMOS
MEDIAN FILTER SYSTEM BASED
ON NEURAL NETWORK
ARCHITECTURAL
PRINCIPLES

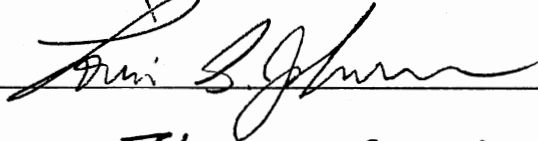
Thesis Approved:



Thesis Adviser









Dean of the Graduate College

PREFACE

This thesis summarizes the investigation of a proposed analog electronic CMOS system for performing median filtering. A description of the problem and rationale for investigating neural networks are given followed by a review of recent efforts toward solving the median filtering problem in hardware. A review of the major developments in hardware neural networks is also presented followed by the system proposal. A comparator design intended to function as a major building block is presented and analyzed. A description of efforts to accurately model the comparator follows. A Spice macro model simulation was assembled as well as a dedicated Runge–Kutta system level simulation. The two models were used to evaluate the system's performance when asked to perform median filtering on a number of different types of input data sets. Methods for predicting the behavior of the system are proposed and compared to simulation results. Finally, conclusions and suggestions for future investigations are offered based on the reported simulation results.

A large amount of time was spent on putting the necessary software in place to do the work that this thesis summarizes. Difficulties with incompatible spice models, curve fitters, pre–production software versions, and communication links between computers abounded. In spite of all these obstacles, some meaningful data was finally generated allowing the conclusion of this effort.

Many people have helped and supported me in my efforts to complete this investigation. I would like to thank in particular my thesis adviser, Dr. Chriswell Hutchens, for his great tolerance of the problems my personal life has injected

into this work, for his light-handed guidance of my efforts, and for his constant willingness to treat me as an equal throughout my college education. I would also like to thank Dr. Keith Teague for allowing me to work on the research contract with Texas Instruments Defense Systems and Electronics Group. Our brief work together spawned the central concept of this thesis. Dr. H. J. Allison supported me throughout much of this effort through my work for him as a Lab Assistant, both financially and scholastically, as much of my knowledge of analog circuit design and frequency response estimation stems from his instruction and my experience grading his students efforts.

Texas Instruments Semiconductor Group also supported this effort through their agreement to hire me prior to finishing my research (I may have had to leave school otherwise) and their insistence upon its completion and willingness to allow me to use their resources to accomplish it. My supervisors at Texas Instruments, Sham Banerji and Kim Asal, did more than I could hope for to help me.

My family, both immediate and in-law, have provided continual moral support, advice, and encouragement. Thanks are owed to all, especially my mother and father, Rob, Susan, Andrew, Jesse, and Jonathan Shell, my brother, Bill, Paul, Ramona, and Darren Brune, Steve Ford and family, and to Louise Beese and Elmer and Gertrude Brune for giving me the encouragement that only grandparents can.

Finally, great thanks are due to my wife, Andrea Helen, for long support and tolerance of the projects, including this thesis, that have made our lives difficult at times. With the completion of this thesis, we are at last closing a long and difficult, but rewarding, chapter of our lives, our formal educations. I am confident that many enjoyable years of continual learning and growth are waiting before us to experience together.

TABLE OF CONTENTS

Chapter	Page
I.	MEDIAN FILTERING AND ELECTRONIC NEURAL NETWORKS 1
	The Median Filtering Problem 1
	Hardware Implementations of Median Filtering 4
	Hardware Implementations of Neural Networks 5
	Neural Network Basics 6
	Analog Neural Networks 9
	Analog Weight Storage 11
	Digital Weight Storage 18
	Digital Neural Networks 26
	Miscellaneous Neural Designs 31
	Example Applications of Neural Systems 37
	Proposal for Neural Implementation of Median Filtering 42
	The Basic Architecture and its Operation 42
	Design Proposal for System Evaluation 44
II.	A PROPOSED COMPARATOR DESIGN 47
	Introduction 47
	General Design Requirements 47
	Meeting the Design Requirements 48
	Circuit Analysis 51
	Operating Point and Hand Calculation of Gain and
	Bandwidth 52
	Zero Generation 60
	Conclusion 65
III.	MODELING THE COMPARATOR DESIGN 66
	Introduction 66
	Justifying the Modeling Effort 66
	The Proposed Models 67
	The System Level Simulation 68
	The Runge–Kutta Numerical Method 68
	Application to the Neural Median Filter System 70
	The Spice Macro Model Simulation 73
	The Macro Model Components 73
	The DC Output Characteristic Model 73
	The Left Half Plane Pole Model 73
	The Left and Right Half Plane Zero Models 74
	Synthesizing an Example AC Response Curve 77

Chapter	Page
The 'Precise' Macro Model	78
The Simplified Macro Model	81
IV. SYSTEM BEHAVIOR AND SIMULATION RESULTS	86
Introduction	86
The Expected System Behavior	86
The Expected Convergence Time	86
The Expected Accuracy	89
The Algorithmic or System Error	90
Output Resistance Error	93
Stability	95
Process Variation	96
Total Error Effects	96
Runge-Kutta Simulation Results	97
Macro Model Simulation Results	103
V. CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK	108
Introduction	108
Summary and Conclusions	108
Suggestions for Future Investigation	109
REFERENCES	113
APPENDIXES.	116
APPENDIX A – EXAMPLE SPICE INPUT DECK FOR THE PROPOSED COMPARATOR DESIGN	117
APPENDIX B – EXAMPLE SPICE OUTPUT DECK FOR THE PROPOSED COMPARATOR DESIGN	121
APPENDIX C – EXAMPLE SPICE INPUT DECK FOR THE 'PRECISE' MACRO MODEL	135
APPENDIX D – EXAMPLE SPICE INPUT DECK FOR THE SIMPLIFIED MACRO MODEL	139
APPENDIX E – SPICE LIBRARY DECK FOR THE PIECE-WISE LINEAR DC OUTPUT CURRENT MODEL	142
APPENDIX F – EXAMPLE SPICE OUTPUT DECK FOR THE SIMPLIFIED MACRO MODEL	154
APPENDIX G – SOURCE CODE LISTING OF C PROGRAM FOR RUNGE-KUTTA SIMULATION	173

LIST OF TABLES

Table	Page
I. Specification for the Final Proposed Comparator Design	53
II. Pole Locations by Spice and by Hand	60
III. Spice Generated Zero Locations	62
IV. Expected Convergence Times for Different Sizes Input Windows	89
V. Nominal Specification for Runge–Kutta Modelled Comparator	99
VI. Comparison of Expected and Actual Runge–Kutta Results	102
VII. Summary of Spice Macro–Model/System Level Simulations	105

LIST OF FIGURES

Figure	Page
1. Sample Data Stream Containing a Discontinuity and an Outlier	2
2. Sample Data Stream After Median Filtering	3
3. Basic Neural System	6
4. Modified Neural System	7
5. (a) Sigmoidal Transfer Function (b) Piecewise Linear Transfer Function . .	7
6. Modified Neural System with Integrated Output	8
7. Hopfield Electronic Neuron	9
8. CMOS Invertor and Differential Amplifier	10
9. Basic Thin-Film Resistive Network	12
10. Basic Gate Capacitance Analog Memory and MOSFET Weight	13
11. Nonlinearity Cancellation	14
12. Differential Transconductance Amplifier	15
13. Programmable Analog Memory Cell	16
14. Floating Gate Analog Memory	17
15. Operational Transconductance Amplifier (OTA)	18
16. OTA Current-Voltage Characteristics	19
17. Multiple Input OTA	20
18. Holmdel Group Single Bit Programmable Weight	21
19. John's Hopkins Single Bit Programmable Weight	22
20. Multiplying D/A Converter (MDAC)	24

Figure	Page
21. CCD Multiplexed Net	25
22. Switched Capacitor Neuron	26
23. Stochastic Network Architecture	28
24. Stochastic Multiplication Example	29
25. Response of Stochastic Network	29
26. Neuron Type Processor Pulse Response (a) Spatial. (b) Temporal	30
27. Neuron Type Junction	31
28. Single Input Neural Type Junction Simulation Results	32
29. Four Input Boltzmann Neuron	35
30. Boltzmann Machine Simulation Results	37
31. (a) Thresholding, (b) Intensity Level Slicing, (c) Intensity Level Slicing w/Background, (d) Thresholding w/Background	38
32. (a) Three Laplacian Operators, (b) Neural Laplacian	40
33. Three Example Intensity Mapping Functions	41
34. Example Back Propagation Architecture for Intensity Mapping	41
35. Neural Median Filter Architecture	43
36. Sample Neural Median Filter Convergence Curve	45
37. Complementary, Common-Source, Push-Pull Output Stage	49
38. Complementary, Parallel, Differential Amplifier	50
39. Cascoded Current Sources	51
40. Final Annotated Circuit Design	52
41. Spice Calculated Operating Point of Current Reference	54
42. Spice Calculated Operating Point of Gain Stages	55
43. Equivalent Small Signal Resistances of Current References	56
44. Equivalent Small Signal Resistances of Gain Stage	57
45. Manual Gain Calculation Results	58

Figure	Page
46. Manual Frequency Response Calculation Results	59
47. Spice Generated AC Response Curve	61
48. Single Pole Blocks Placed in Parallel	63
49. The Non-Linear Transfer Function Approximations for the Runge-Kutta System Level Simulation	70
50. Neural Median Filter Architecture	71
51. The Runge-Kutta Simulation Algorithm for the Proposed System	72
52. The DC Output Characteristic Model	74
53. Comparison of the DC Transfer Function of the Design and the Model Driving (a) a Nominal Load, and (b) a Larger (Less Re- sistive) Load.	75
54. The Left-Half Plane Pole Model	75
55. AC Response Curve of the Pole Model	76
56. Schematic of the Zero Model	77
57. AC Response Curves of the Zero Models	78
58. Example Pole-Zero System	78
59. Asymptotic AC Response of the Example System	79
60. AC Response of the Example System	80
61. Schematic of the Precise Macro Model	81
62. Comparison of the Circuit and Macro Model's AC Response	82
63. Comparison of the AC Responses Under Other Load Conditions	83
64. Schematic of the Simplified Macro Model	84
65. Comparison of the Circuit and Simplified Model's AC Responses	84
66. Comparison of the Circuit and Simplified Model's Unity Gain AC Responses.	85
67. Results of Transient Analysis of Simplified Macro Model	85
68. Compact Schematic Description of the Neural Median Filter	94

Figure	Page
69. Illustration of $f_i()$	95
70. Illustration of $\Sigma f_i()$	96
71. Effect of Adding Output Resistance to $I_{OUT} = \Sigma f_i()$	97
72. (a) Large Output Resistance's Effect on 'Median' Portion of the Curve, and (b) Small Output Resistance's Effect on the 'Median' Portion of the Curve.	98
73. Example 9 Sample Input Window	100
74. Example Output Voltage Convergence Curve	101
75. Sample Input Windows	104
76. (a) Folded Cascode, (b) Cascoded Output Stage	110

CHAPTER I

MEDIAN FILTERING AND ELECTRONIC NEURAL NETWORKS

The Median Filtering Problem

In the last ten to twelve years, a class of circuits and systems referred to as neural networks have received a large amount of scrutiny. This class of circuits and systems covers a huge variety of ideas and functions but they all have a few characteristics in common. They are all composed of many processing elements performing a very simple operation in parallel on some subset of the inputs to the system. The majority of the problems they are applied to can be described as pattern recognition or signal processing problems that require some amount of parallelism and non-linearity.

This thesis will present a proposal for an electronic system that can be classified as a neural network. This electronic system will perform the function of median filtering on an input window of sampled analog voltages.

Median filtering is a non-linear filtering technique to smooth sampled sequences of data. Its primary function is the removal of impulsive noise, also called salt and pepper noise or outliers, from sampled sequences of data without distorting sharp edges or discontinuities as a linear low-pass filter would, blurring the data terribly.

Median filtering, when performed sequentially, requires that a window of the input data stream be taken about every sample. The sample values in that window are then sorted into ascending order and the median value is picked to re-

place the original pixel. Consider Figure 1, below. The graph represents a stream of sampled data containing a discontinuity and an outlier.

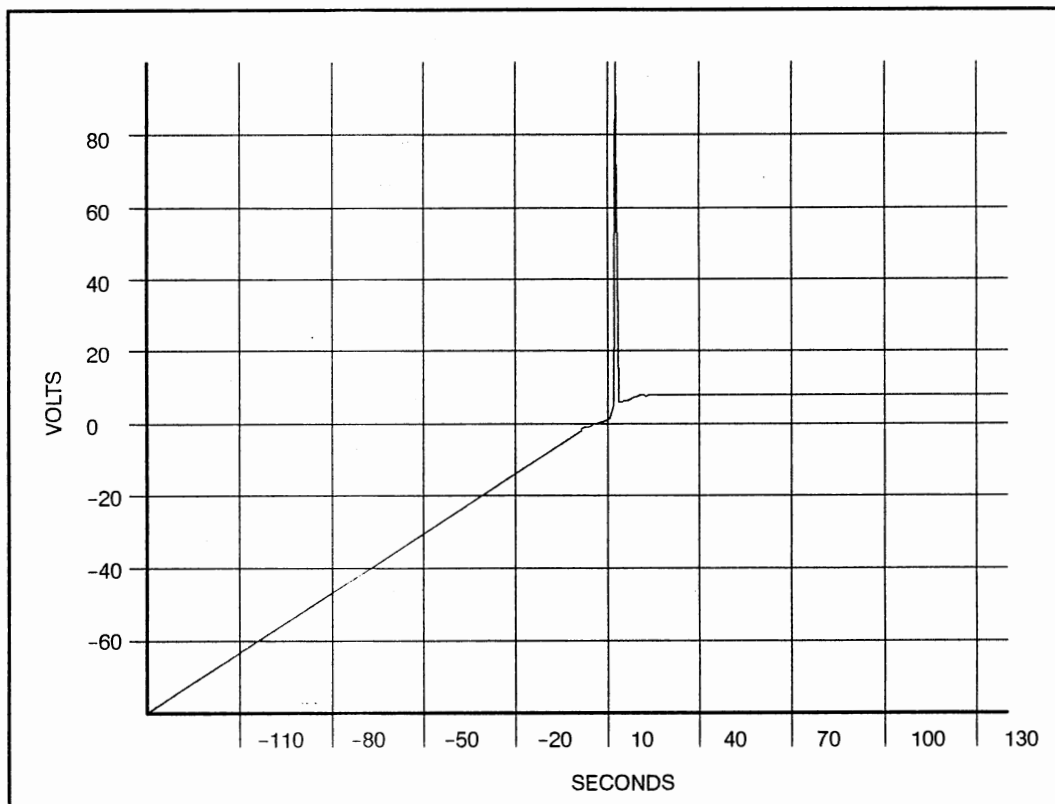


Figure 1. Sample Data Stream Containing a Discontinuity and an Outlier

Figure 2, below, represents the sample data stream after it has been median filtered. The stream is 280 samples long and the median filter window used was 5 samples. One can see that the data stream is not visibly altered.

If one assumes that the input data is a digital image of 512 X 512 pixels, using a 5 X 5 window implies that to filter a single image would require close to 6.5 million comparisons. A crude implementation of this algorithm would require at least 160 milliseconds on a state of the art digital signal processing chip

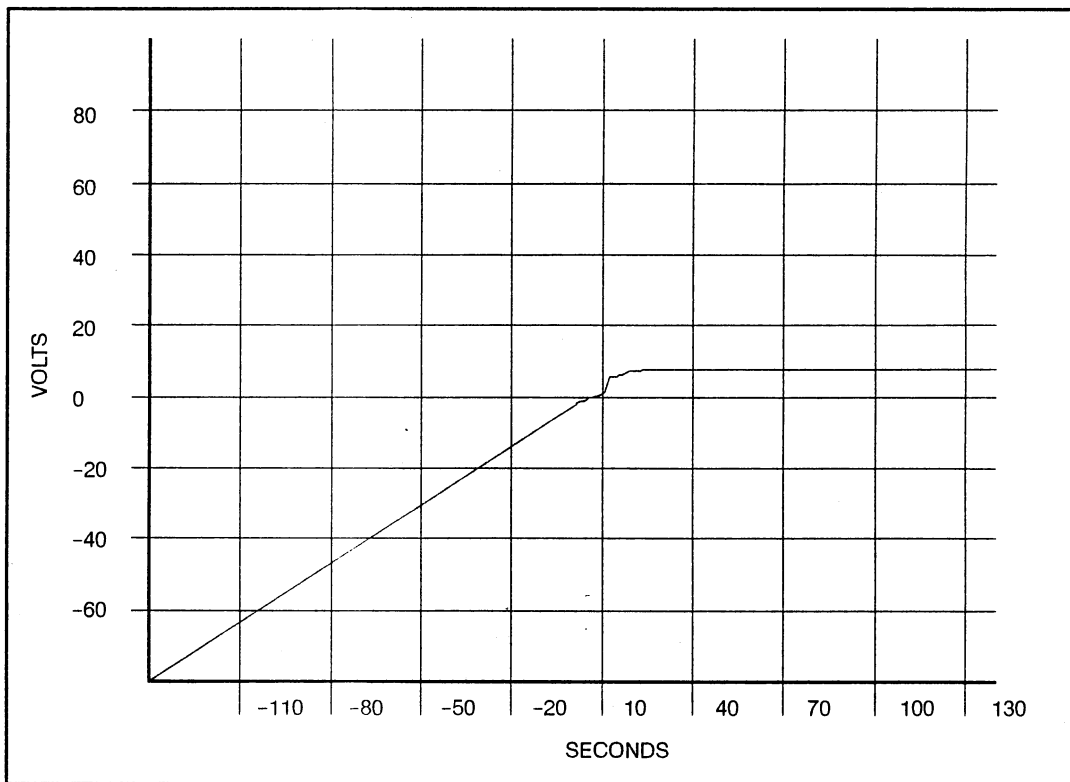


Figure 2 Sample Data Stream After Median Filtering

(assuming 25 nanosecond cycle time, an aggressive specification for contemporary DSP chips). This is 625 nanoseconds per window. Optimizations of this algorithm can result in processing times that are proportional to the square root of the number of pixels in the filtering window rather than number of pixels in the window. One such algorithm results in a 20% decrease in computation time for our example 5 X 5 window. The advantage increases, of course, with window size: 51% improvement for 7 X 7, 65% improvement for 9 X 9, etc.

The limiting factors on computation time for a serial median filter are, of course, serialism and clock speed. As technology improves, digital signal processors (and computers in general) get faster and faster. To improve beyond that however, requires that one find ways to implement filtering algorithms in parallel.

As has been stated before, neural networks are often investigated in connection with signal processing applications because their inherent parallelism will presumably provide a speed up over conventional, serial, signal processing algorithms. The work that this thesis summarizes was the investigation of a dedicated, hardware solution to the parallel median filter problem using a system inspired by the neural network architectural paradigm.

Hardware Implementations of Median Filtering

Before one delves too deeply into a problem, it is best to find out what has been tried before so that any new solutions can be judged by comparison. Toward that end a literature search was performed. Some of that effort resulted in the serial median filter speed estimations given above. The remainder of the effort involved searching for hardware implementations of median filtering algorithms and systems. Only three were found.

Chang long Lee and Chein-Wei Jen of the National Chiao Tung University in Taiwan proposed a novel design of a majority bit circuit. The circuit gave a binary output signal which corresponded to the value that the majority of the input bits were equal to. They then proposed a novel algorithm for calculation of the median of a window of N -bit binary numbers by applying majority selection recursively to the bits of the input window samples. They then proposed a VLSI architecture based on this algorithm which they estimate will run at 20 MHz in 2um CMOS. The number of cycles required to generate a median is proportional to the number of bits in the input words. So for 16 bit quantized samples, their proposed system requires 16, 50 nanosecond cycles corresponding to a calculation rate of 1.2 Mega-medians per second.

Arce et al. of the University of Delaware, also proposed a VLSI architecture based on recursive multi-level median filters, in which the final median is the me-

dian of the medians of several subsets of the input window. His proposed design in 3u Mosis CMOS takes up approximately half of the largest Mosis frame and is expected to be able to process 256x256 images with a 5x5 window at a 'real time' scan rate of 60Hz. Alternatively, the same design in 2u CMOS should be able to process 512x512 images at the 30Hz rate. These speeds correspond to 3.9 and 7.8 Mega-medians per second respectively.

Finally, Karaman et al. of Turkey's Bilkent University, have proposed a pipelined and systolic design that is essentially a series of pass-and-swap comparison units. The network effectively performs the brute-force median filtering algorithm of sorting and picking the middle value. Once the pipeline is full, the system can evaluate medians at the clock rate. Therefore, according to their simulations, at a 50MHz clock rate, their system can generate up to 50 Megamedians per second for a fixed window size and word length. They also propose an extensible design that can generate $30/L$ Megamedians per second at 40MHz where L is length of the input word.

Hardware Implementations of Neural Networks

Now that the existing approaches to median filtering have been summarized and evaluated, the following text will review what types of neural systems have been implemented in hardware before the proposed neural median filter system is presented.

In order to understand what types of neural network systems have been implemented in hardware, it is necessary to understand what the basic building blocks of neural networks are and how they are commonly used. After a brief review of these things, the following section will summarize the major attempts that have been made at realizing neural systems in hardware to take advantage

of the parallelism that serial simulations of neural networks lose and to evaluate how these systems might be used to solve problems.

Neural Network Basics

One way to classify neural systems is to consider whether their computation is static or dynamic. Figure 3, below, illustrates the simpler static case: a weighted sum of an input vector followed by a threshold unit, also called a processing element or neuron. Figure 4 modifies the system by inserting an integrator between the sum and threshold, making the computation dynamic. This addition of dynamics to the system can increase its computational power, in some cases, at the expense of response time. Further possible modifications include changing the hard threshold to a sigmoidal or piece-wise linear transfer function as shown in Figure 5, (a) and (b), and moving the integrator to the output as shown in Figure 6.

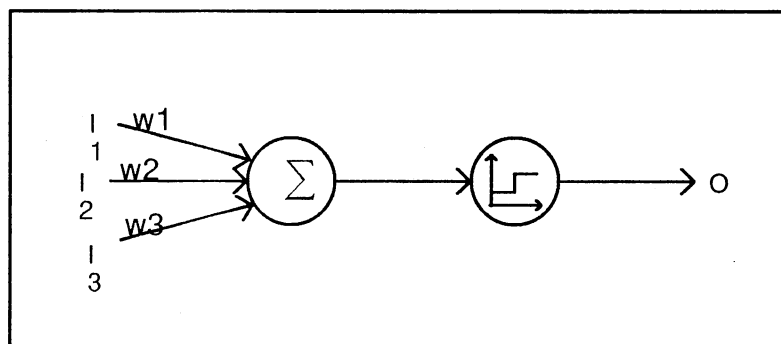


Figure 3 Basic Neural System

These systems, which can be said to emulate biological neurons, may be combined in a bank of parallel units supplying an output vector which can then be cascaded to further banks of neurons, or fed back to its own input vector, or both. Systems that use these types of architectures are called neural networks.

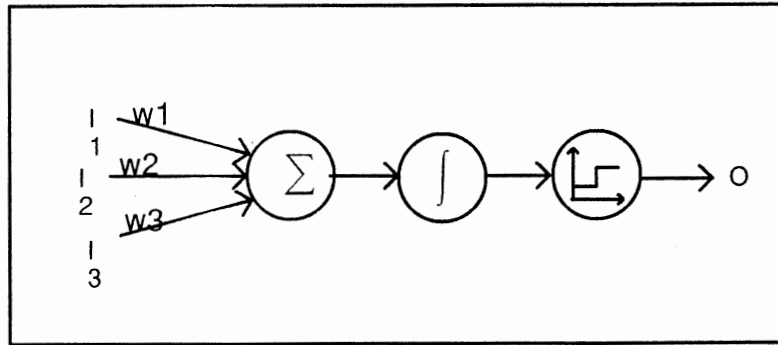


Figure 4 Modified Neural System

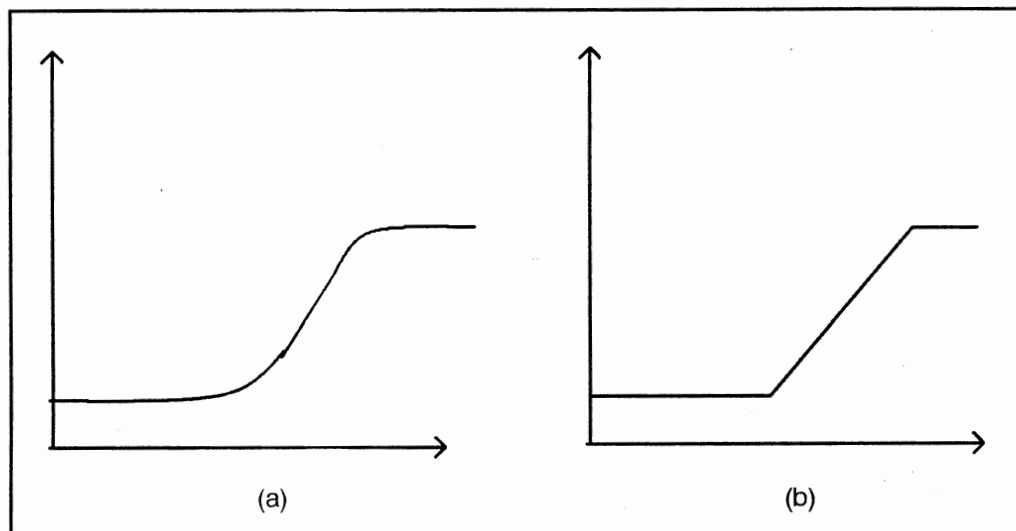


Figure 5 (a) Sigmoidal Transfer Function (b) Piecewise Linear Transfer Function

A neural network in which every input vector element is connected to every threshold unit by a weight and/or in which every threshold unit is connected to every other threshold unit is called a fully interconnected network. Other, more sparsely connected networks, are called locally interconnected networks. Many signal processing tasks, particularly image processing, lead to neural network implementations which are highly parallel but, unfortunately, many are also highly interconnected.

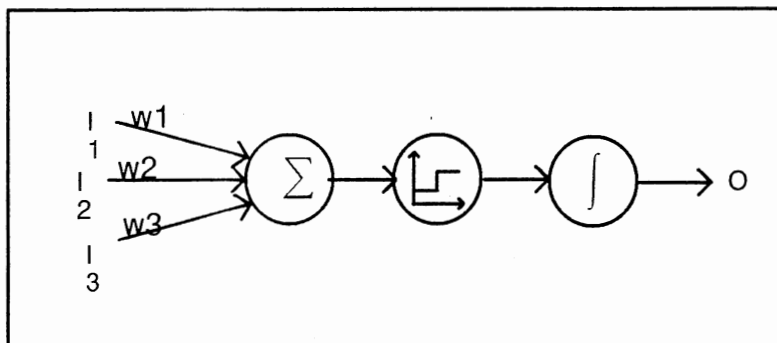


Figure 6 Modified Neural System with Integrated Output

In 1984, Dr. John Hopfield of CalTech published a paper suggesting an implementation of neural systems using common electronic components. His basic electronic realization is shown in schematic form in Figure 7, below. The state variables of the system are the input and output voltages, the weights are approximated by resistors (conductances) which inject a current proportional to an input voltage multiplied by a weighting factor (the magnitude of the corresponding conductance) into a capacitor (summer) on the input of an amplifier exhibiting a roughly sigmoidal response (the processing element). In this electronic approximation of an artificial neural system, the amplifier responds to the integral of the weighted sum of inputs rather than just to the weighted sum, although one could also view the capacitor–amplifier combination as taking the sign of the weighted sum of inputs, that is, the charging current, implementing the threshold rule.

This analog electronic neural model directly inspired most of the current work on realizing electronic neural nets in VLSI. The following paragraphs will first describe efforts toward realizing neural nets using analog components, followed by the progress made toward realizing a fully digital electronic neural net. Finally, those efforts which do not easily fit into the preceding classifications will be discussed.

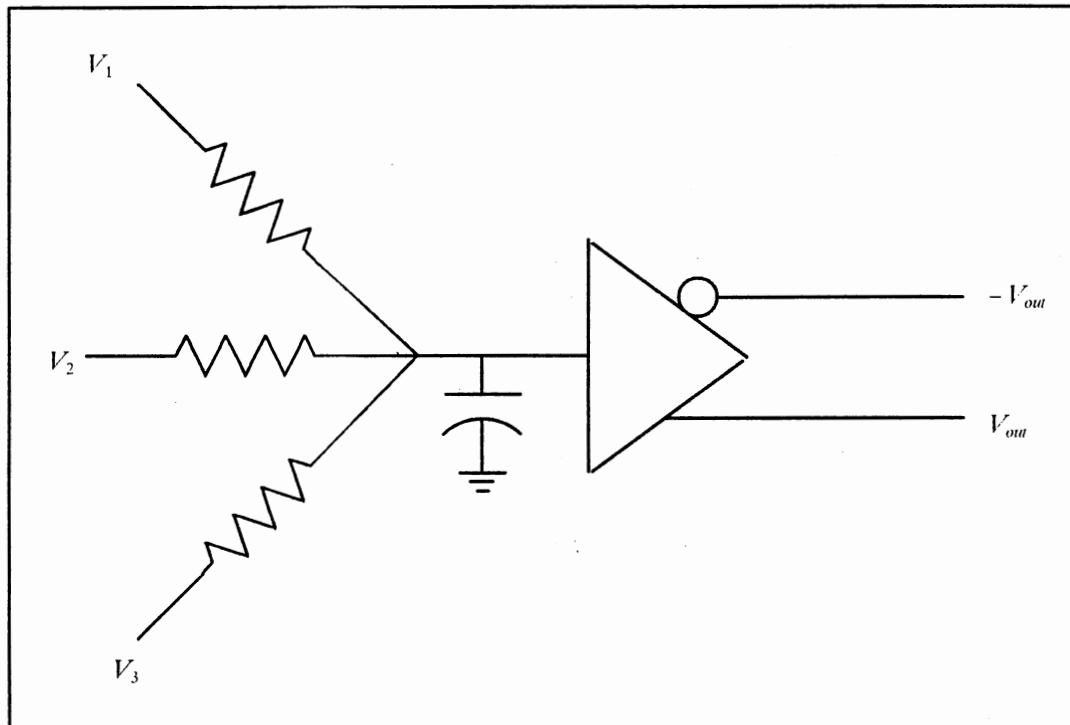


Figure 7 Hopfield Electronic Neuron

Analog Neural Networks

Realizations of analog electronic neural networks differ from one another in two major respects: how the nonlinearity is generated, and how the weights are realized and stored in long term memory. Other more subtle differences can arise like whether the input to the amplifier is the weighted sum of the inputs, does the state of the network evolve in continuous time or is the neural computation synchronously clocked, and is the network truly parallel or are the computations partially or completely multiplexed between a small number of neural processors. Hopefully, the organization of the designs discussed below with their

differences and contrasts will aid the reader's comprehension rather than losing his attention in a sea of detail.

The central element of the electronic neural net, the threshold-like sigmoidal nonlinearity, is the element that has received the least attention. Almost every analog hardware neural net use either a pair of invertors or a difference amplifier. Figure 8 shows these two possibilities in the CMOS paradigm that virtually every neural system designer uses.

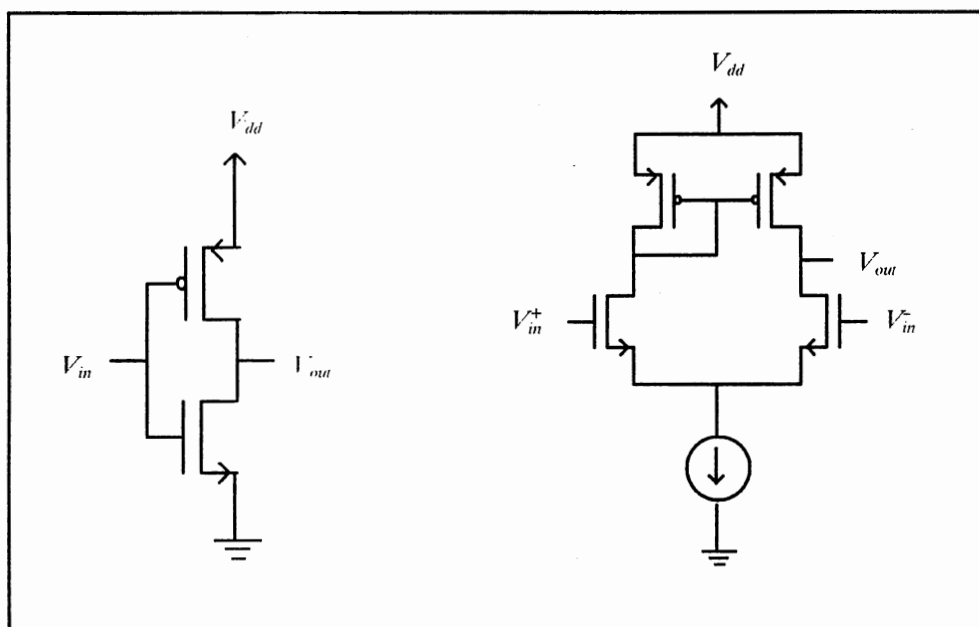


Figure 8 CMOS Invertor and Differential Amplifier

While the double invertor uses only four transistors and is very straightforward to design, the differential amplifier exhibits better tracking between the inverting and non-inverting outputs while in the semi-linear region and the availability of differential inputs allow the use of differential weights to help eliminate errors from device parameter variation.

The realization of the multiplicative weights, on the other hand, varies greatly from design to design, both in their actual construction and in the storage of their values, fixed value or programmable using analog or digital memory.

Analog Weight Storage. The most natural progression from Hopfield's initial model to an actual implementation is to use a resistive matrix for the weights. Howard et al. were among the first to do this at AT&T Bell Labs' facility in Holmdel, New Jersey in 1986. Using electron beam lithography, they fabricated a 22 X 22 array of amorphous silicon, thin film resistors on a chip along with 22 CMOS invertors (see Figure 9). Each resistor had a value around 300kOhms and was patterned into the neural circuit at the time of fabrication.

With this chip, the Holmdel group successfully demonstrated the use of a Hopfield type network as an associative memory with response times of one to ten microseconds. A year later, using an improved amorphous silicon technique, the same group designed and fabricated a chip having a 256 X 256 resistive array to be used in the same manner. The thin film resistors were valued around 2MegaOhms and resulted in response times around 700 nanoseconds. It should be said that these chips were made for experimental purposes and could really not be put to any practical use because the stable states of the network were permanently programmed into the network at the time of fabrication in a fairly expensive and irreversible step.

An obvious improvement to the Holmdel group's efforts would be to make the weights programmable and variable. In November, 1987, Y. Tsvividis and S. Satyanarayana of Columbia University in New York, New York, suggested using a MOSFET as a weighting conductance whose magnitude may be controlled by modulating the gate voltage. Figure 10 illustrates this concept. This analog control voltage may be stored on a capacitor.

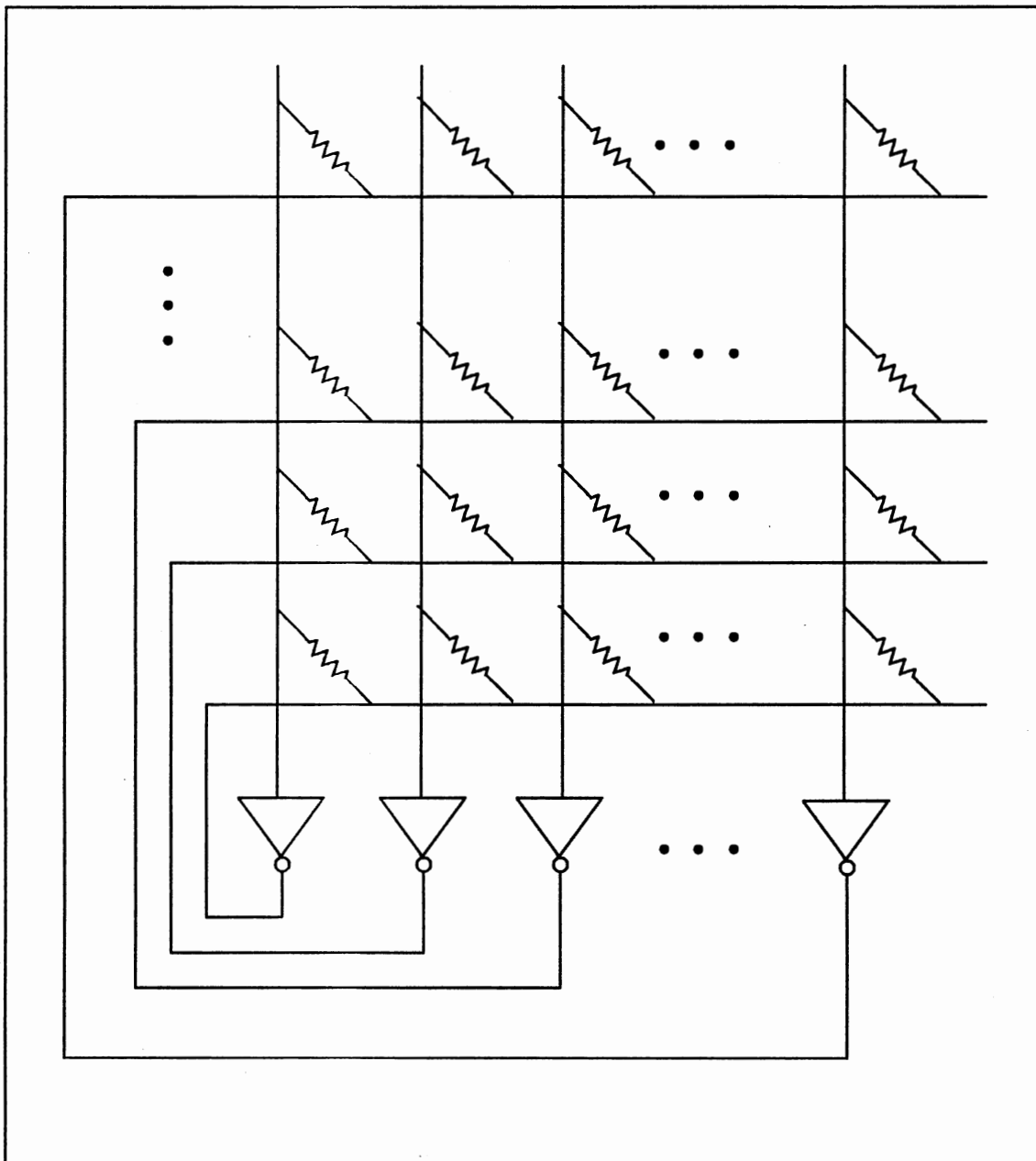


Figure 9 Basic Thin-Film Resistive Network

They also suggested cancelling the inherent nonlinearity of the transistor by using complementary input voltages through matching weighting transistors or by passing the same voltage through complementary weighting transistors, n-channel and p-channel. See Figure 11, below.

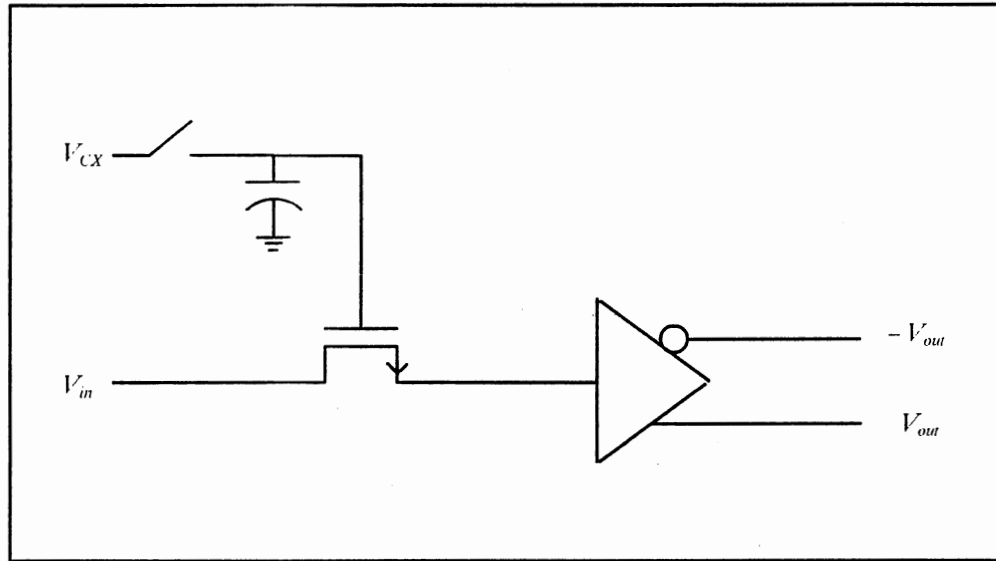


Figure 10 Basic Gate Capacitance Analog Memory and MOSFET Weight

Another weighting scheme is shown in Figure 12. By varying the bias voltage, the transconductance of the differential pair changes, in effect weighting the differential input voltages. No mention was made of measures to decrease the leakage currents that will drain the capacitance or the circuitry necessary to refresh the analog voltage periodically.

Many groups have been attending that problem. In March, 1989, R. Hochet of the Swiss Federal Institute of Technology has suggested using a clocked current source both to charge a capacitor to store a weighting magnitude and to refresh the charge on that capacitor as it decays in response to leakage current. The ratio between the current source's control clock and the main control clock sets the accuracy or resolution of the voltage stored on the capacitor. Hochet's example used a 100kHz clock on the charging current source and achieved five bits of accuracy.

In May of 1989, Swartz et al., of the Holmdel AT&T Group, submitted a design claiming 10 bits of accuracy, two billion weight changes per second, and

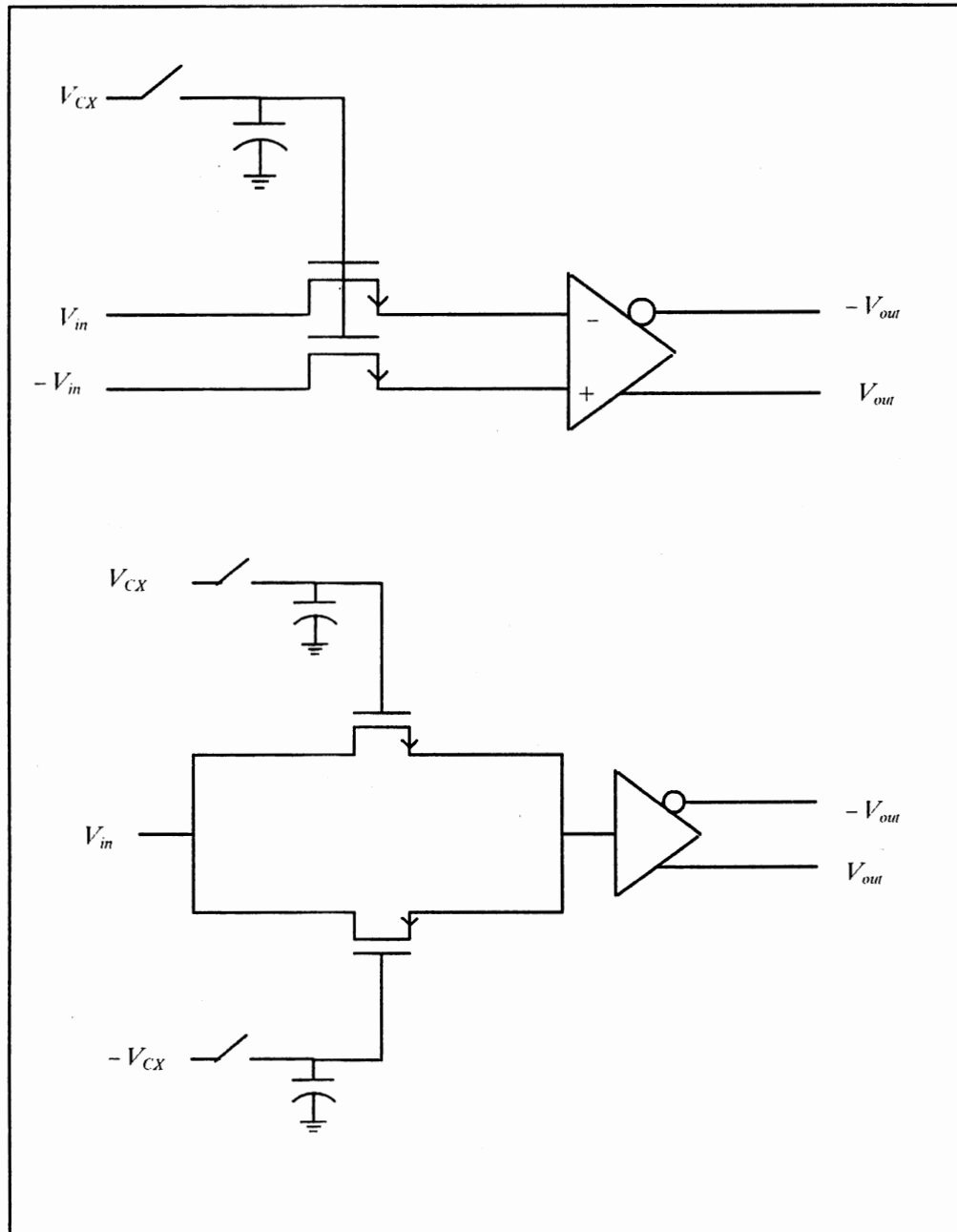


Figure 11 Nonlinearity Cancellation

storage times between refreshes up to 1 second at room temperature. The storage time increases logarithmically as temperature decreases, exhibiting a maximum reported storage time of 200 seconds at 265 K. These figures are achieved by purposefully exploiting charge injection in the MOS devices intended

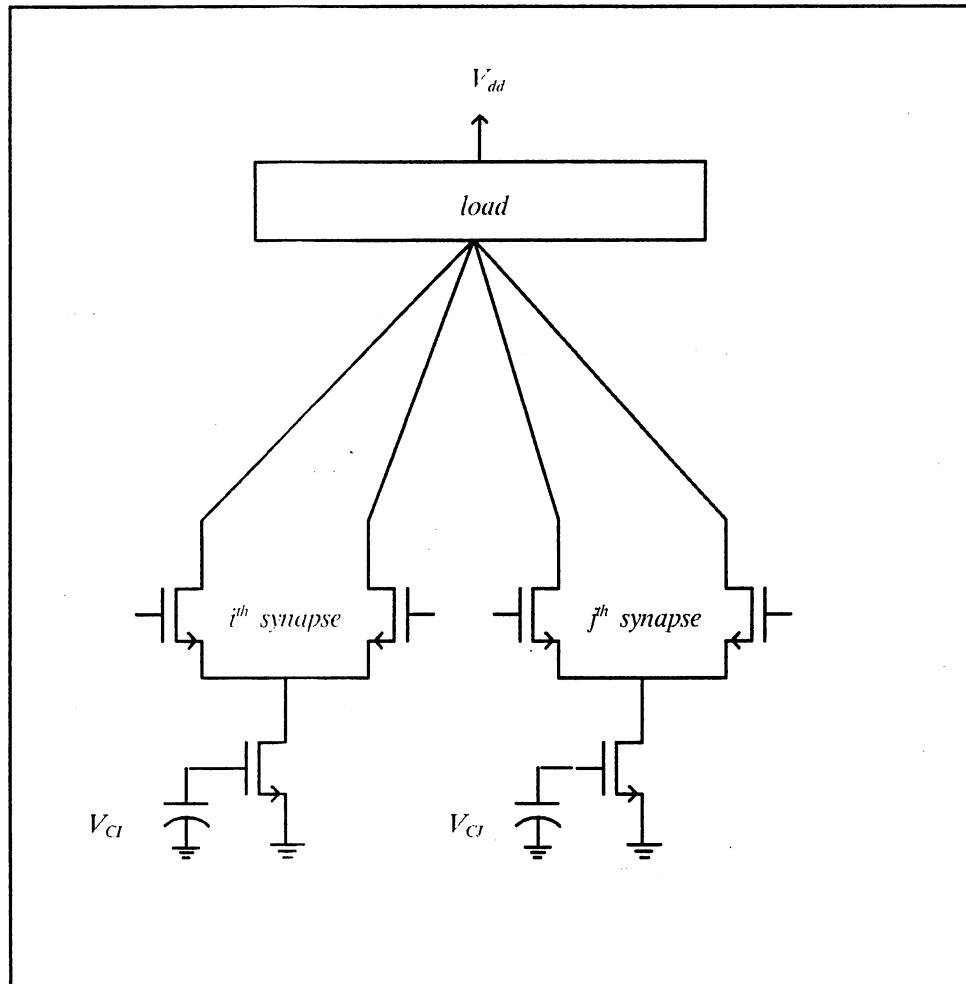


Figure 12 Differential Transconductance Amplifier

to charge the storage capacitor and by optimizing the layout to minimize parasitic leakage currents. This optimization includes minimizing contact area on storage nodes to reduce recombination currents, using long, narrow access transistors to reduce subthreshold leakage to the level of junction leakage, and isolating storage nodes from large capacitances driven at high frequency to avoid hot carrier effects.

The design consists of long channel injection transistors connected to differential storage nodes by clocked pass transistors (shown in Figure 13 below) and has two major flaws.

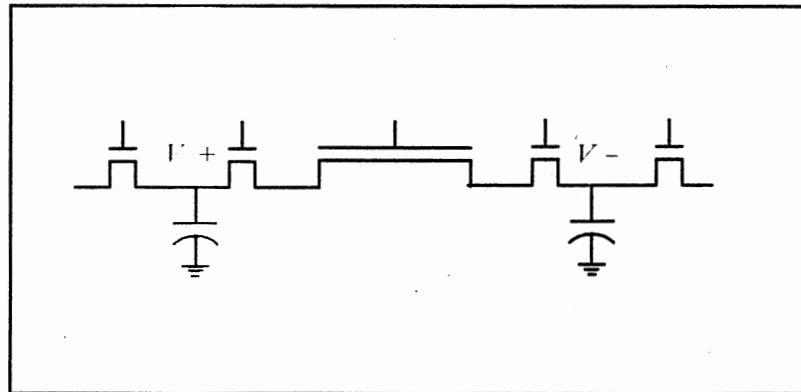


Figure 13. Programmable Analog Memory Cell

Firstly, it requires enough area to store each weight that the weight density is limited to about 1000 on a standard sized chip (for around 30 fully interconnected neurons), and secondly, that the charge increments injected onto the storage capacitor are not constant in size, depending on the already stored voltage, meaning that on a given update cycle, not all weights will be updated by the same amount or even by easily predictable amounts.

Also deserving reporting is the suggestion in May, 1988, by Shimabukuro, Reedy, and Garcia of the Naval Ocean Systems Center in San Diego California, that floating gate MOS circuitry be used to store analog current values. The design, shown basically in Figure 14, uses the high energy electrons resulting from large electric field regions near the drain of the controlling transistor on the left.

These large electric fields occur during avalanche multiplication induced by a critical negative voltage (-12 V) applied to the drain. The bottom pair of transistors mirror the effect using positive critical voltages and high energy holes. Predicted storage times at room temperature exceed ten years. Some disadvantages of this approach are the requirement of high voltage control lines, the time required for the control pulses (the authors used fifteen microsecond pulses

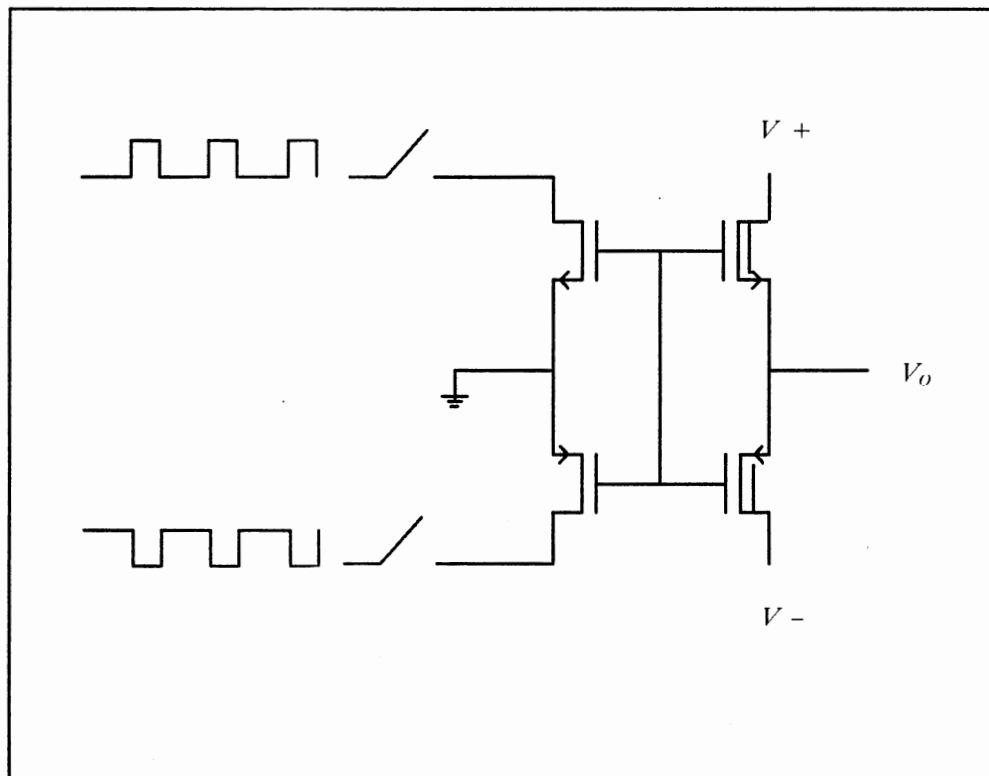


Figure 14. Floating Gate Analog Memory

— no justification or explanation for this value was given), the p-channel devices require more pulses than the n-channel to cover a similar dynamic range, and finally, that the current sunk by the floating gate device is nonlinear in the gate voltage and therefore in the number of applied pulses. The authors made no mention of the device size requirements.

Lastly, in May, 1989, Reed and Geiger of Texas A & M University reported a design for an operational transconductance amplifier (OTA) nicely suited for use as a weighting element because of its considerable linearity. The two transistor circuit, shown in Figure 15, sinks an output current directly proportional to the input voltage minus a small offset. Figure 16 exemplifies the expected current-voltage transfer function.

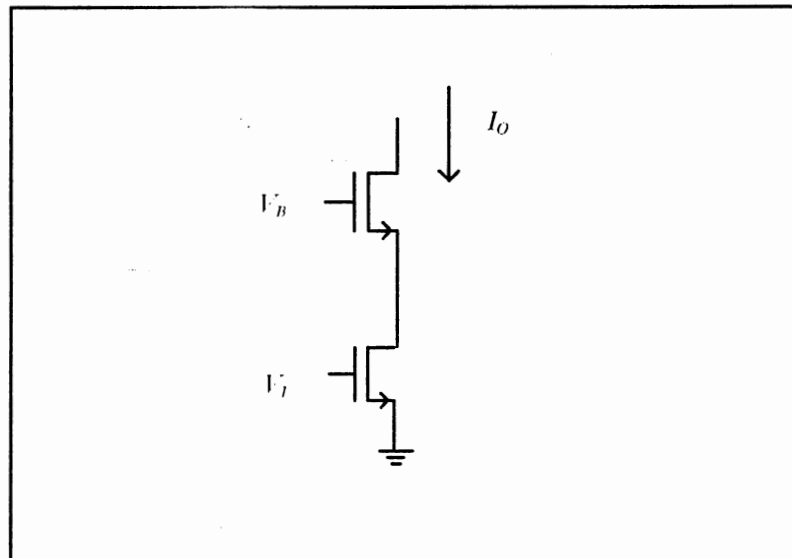


Figure 15. Operational Transconductance Amplifier (OTA)

The transconductance of the amplifier is also linear in the applied bias voltage making it ideally suited for those amplifiers using analog storage of the weight magnitudes because the nonlinearity of the weights doesn't have to be considered when calculating the magnitude of the weight storage charge. The output currents can be converted to excitatory or charging currents by passing them through a current mirror (see Figure 17). This design was included at this point in the report because, of all the weighting schemes requiring analog bias voltages, this one seemed the most useful and the most reasonable.

Digital Weight Storage. An option to storing the weights as an analog voltage to be carefully held as charge on a capacitor and refreshed at intervals would be to store the weights digitally and insert them into the analog network with digital to analog converters. The simplest realization of this is single bit storage, that is, either a weight is present or it is not. Its magnitude is either

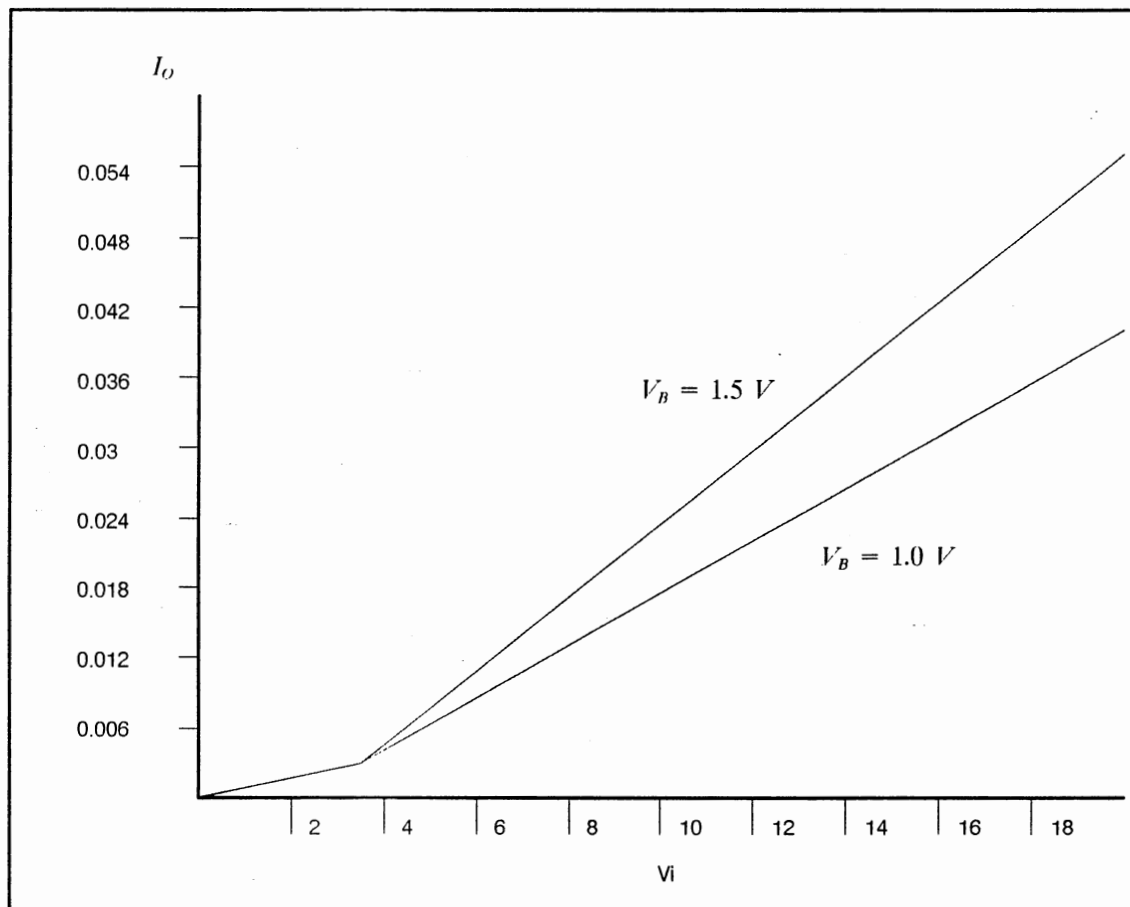


Figure 16. OTA Current-Voltage Characteristics

some nominal value or zero. Jackel, Graf, and Howard of the Holmdel AT&T group accomplished this in a 54 neuron chip in mid 1986, as did Silvotti, Emerling, and Mead of CalTech with their 17 neuron design. Moopenn, Lambe, and Thakoor, of the Jet Propulsion Laboratory at CalTech, published their 32 neuron design in March 1987. Finally, in May, 1989, Kwabena A. Boahen et al. of Johns Hopkins University reported a 46 neuron bi-directional associative memory design and Michel Verleysen et al. of the Univerisite Catholique de Louvain, Belgium, published their 128 neuron design.

The major difference between these designs is the actual realization of the weights and how the digital memory activates the weights. For instance, the

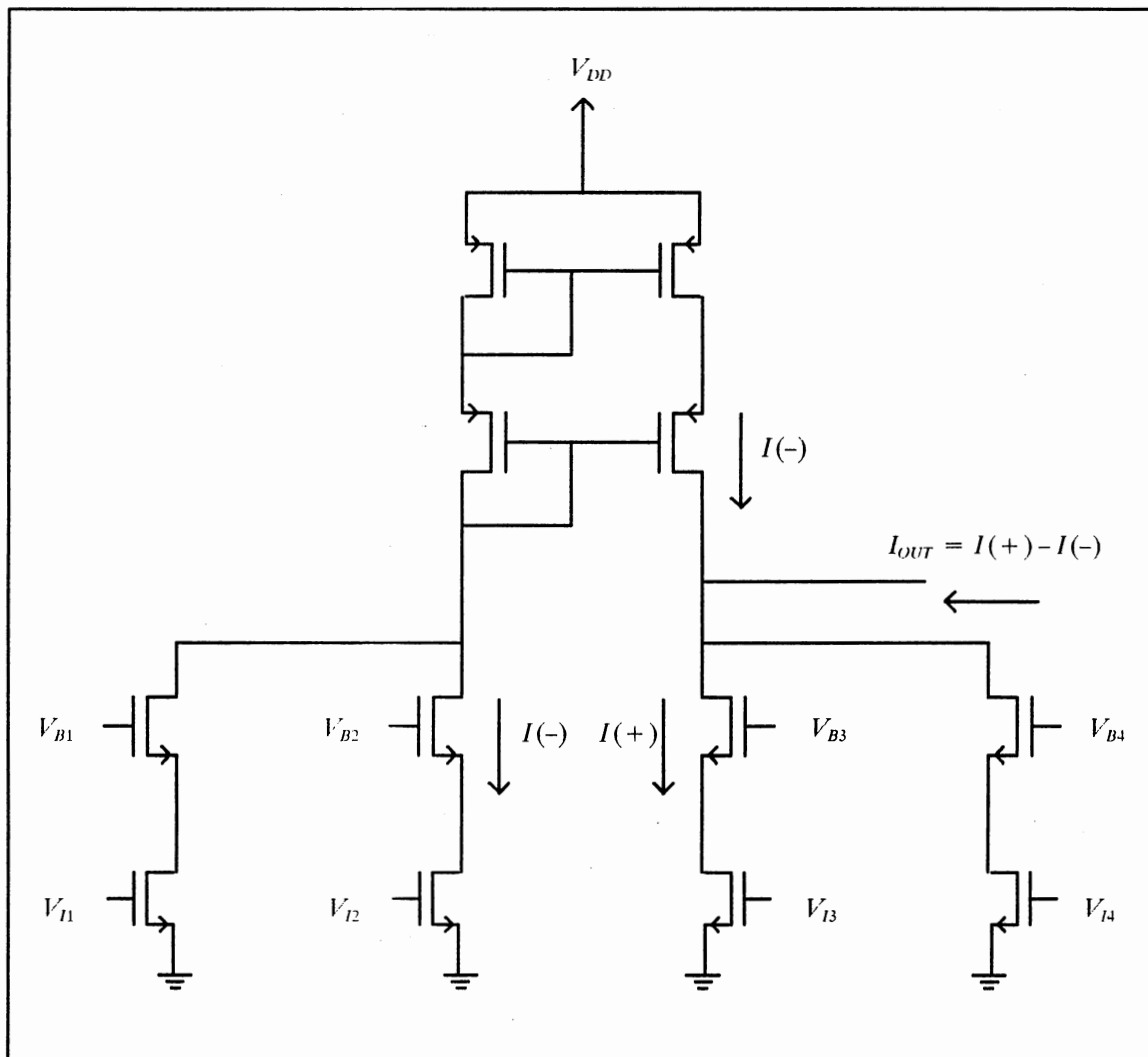


Figure 17. Multiple Input OTA

Holmdel group's design uses a RAM cell to drive a switch in series with an active resistance to either power rail. A second series switch connects the RAM selected active resistance to the input of a neuron (amplifier). The second switch is driven by the output of another neuron. In this way, a high voltage level on the output of one neuron injects an excitatory or inhibitory current onto the input of another (see Figure 18).

Mead and Silvotti's CalTech group, on the other hand, go the more conservative, Hopfield inspired, route of actually switching a conductance into a series

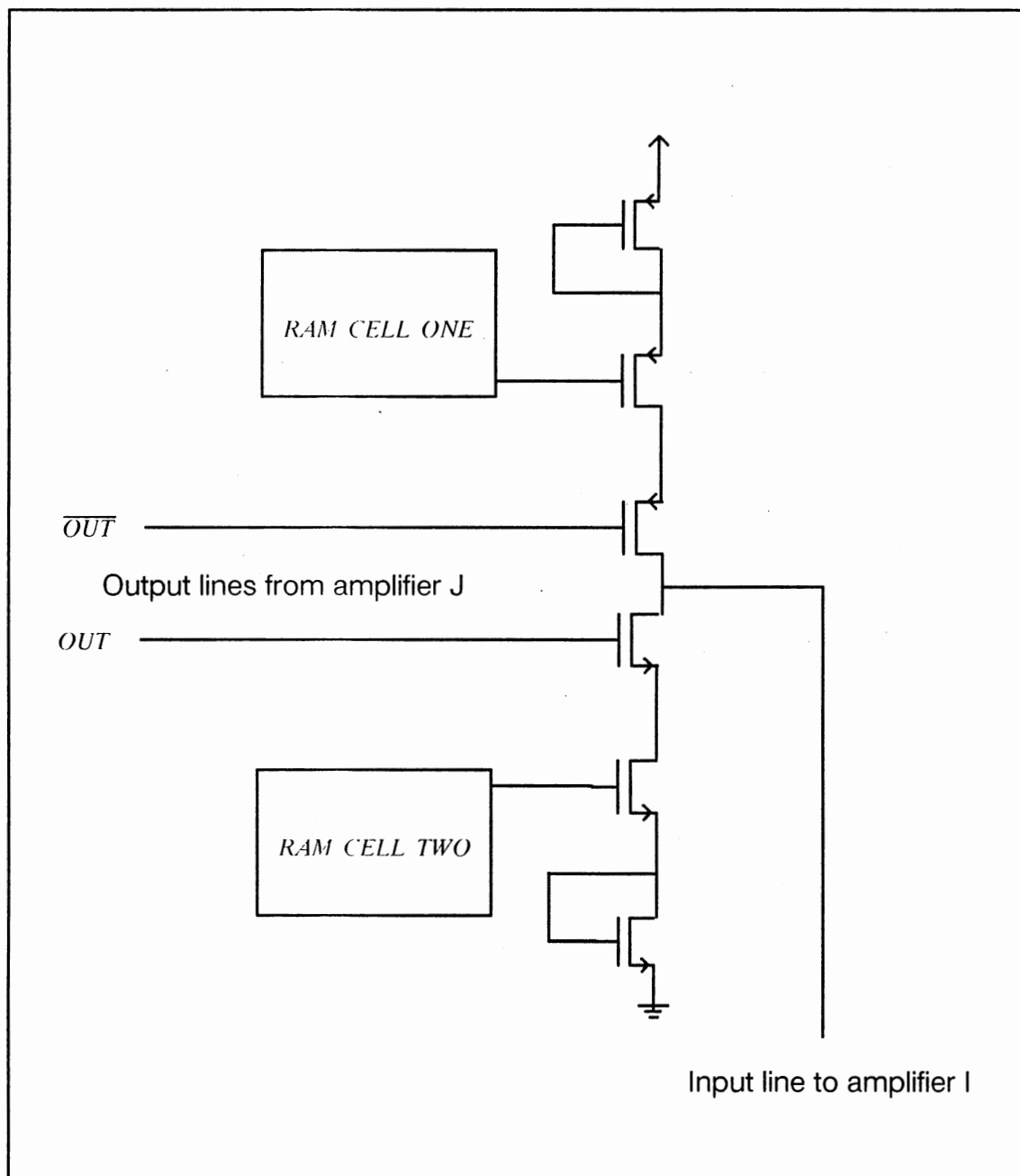


Figure 18. Holmdel Group Single Bit Programmable Weight

connection between two neurons. Their lower density stems from the fact that they used NMOS rather than CMOS as the Holmdel group did, and also that their synapse cell used about twice as many transistors as Jackel's design.

The Johns Hopkins design is also similar to the Holmdel effort except that all currents are excitatory. The single bit programming sets whether a weight's output current responds to the inverting or non-inverting output of a neuron. The basic synaptic cell is shown in Figure 19.

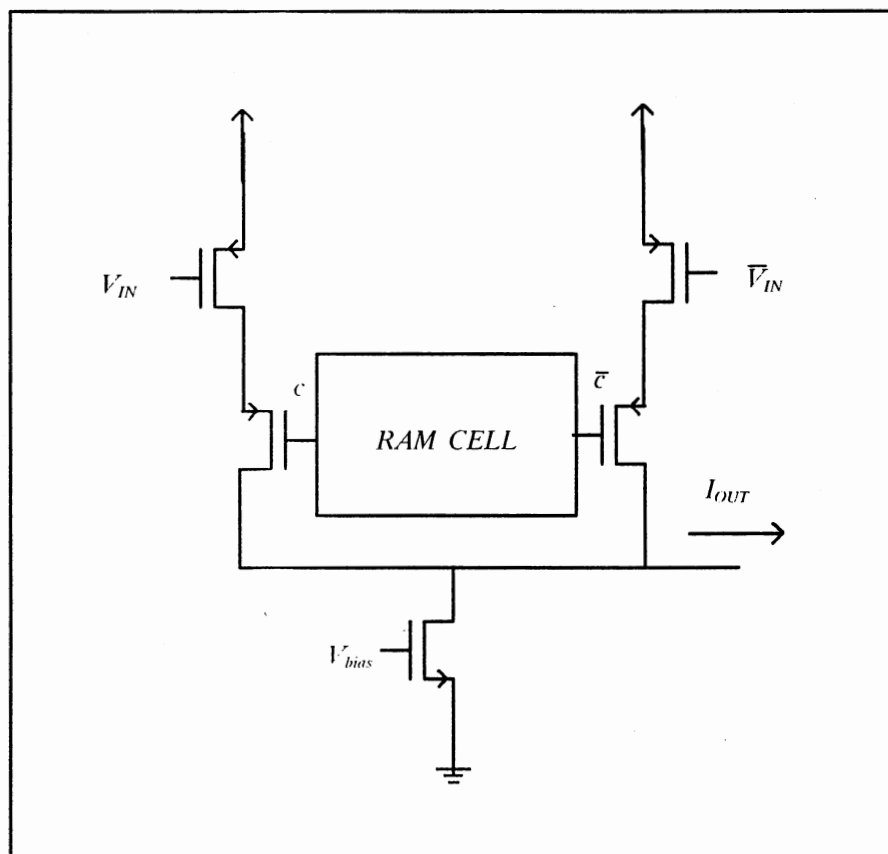


Figure 19. John's Hopkins Single Bit Programmable Weight

Upon examination, it can be easily seen that for a stored high bit, the output current responds to the non-inverting neuron output voltage, while for a stored low bit, it responds to the inverting neuron output voltage. A bias current is included on the bottom to allow a degree of programmability to the net in terms of average output current levels, and therefore, speed of response and power consumption.

Verleysen's design is very similar, using one RAM cell to determine if a connection exists and a second cell to determine if it responds to a high output voltage or a low output voltage.

Greater dynamic range could be added to these digitally programmable nets by adding more bits of memory per weight driving binarily weighted current sources. This was realized in silicon in May 1987 by Raffel et al. of MIT's Lincoln Labs. Each weight has 5 associated bits of storage, four magnitude bits and one sign bit. Each magnitude bit drives a corresponding number of parallel conductances that in turn connect from one neuron's output to another neuron's input. The sign bit selects whether the currents produced are sourcing or sinking (excitatory or inhibitory). The basic multiplying digital-analog converter cell is shown in Figure 20. A test chip with 1024 programmable MDAC weights (corresponding to 32 neurons) has been fabricated and tested successfully. An alternative realization is to drive binarily weighted current sources.

The limiting factor on all of these designs has been the huge amount of chip space the connection matrix occupies making the amplifier area almost negligible. By sacrificing total parallelism, and therefore response time, the size of a single chip network can be drastically increased through the multiplexing of weights.

At the beginning of a cycle, the output voltages of all neurons are sampled. Then the weights to each neuron are loaded and the weighted sum of outputs is evaluated before moving to the next neuron. After all the inputs to the neurons are evaluated, the outputs are discarded and the next set of output voltages are sampled. Moopenn et al. of JPL implemented this strategy in 1987. By clocking the multiplexing cycles much faster than the time constant of the neurons, the continuous time Hopfield network was approximated. Conversely, by clocking

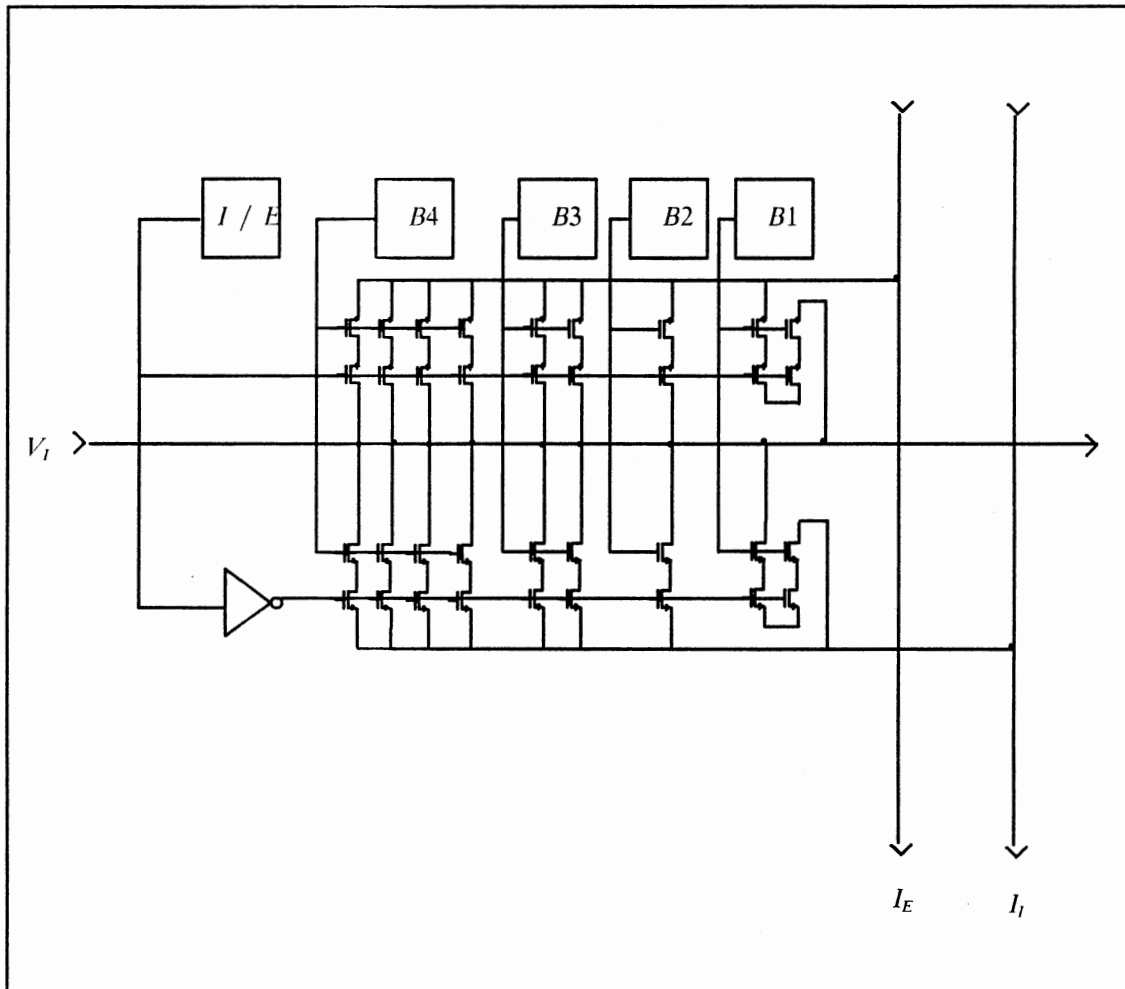


Figure 20. Multiplying D/A Converter (MDAC)

the cycles much slower than an input time constant, the synchronous Hopfield network was realized.

Another striking example of this strategy was described by Agranat and Yariv of CalTech in May 1987. Using charge coupled device technology, they designed a network with predicted densities up to 2000 neurons on a chip and response times in the neighborhood of 100 microseconds. Figure 21 illustrates the system in system level block diagram form.

The array of circularly connected registers, SYNPs, hold a vector of analog voltages which are accumulated in ACCUM as they are enabled by the appropri-

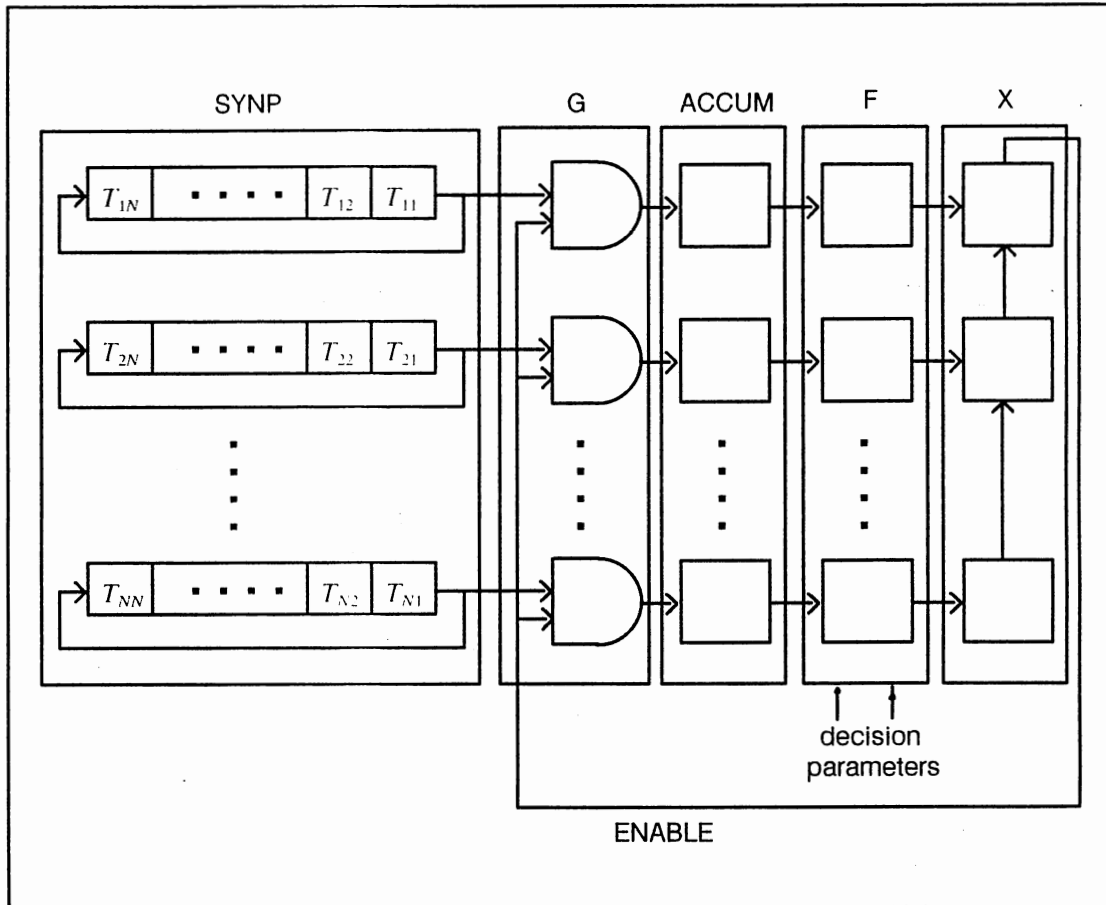


Figure 21. CCD Multiplexed Net

ate output X being in a high state. During one update cycle of the network, the circular output register, X, synchronously shifts through all N output values as the circular weight registers do the same, thereby accumulating the weighted sum of outputs before passing the value to F for the evaluation of the new X vector. No actual implementation based on this scheme has been reported.

Finally, a slightly different weighting scheme was proposed in August 1987 by Y. P. Tsvidis and D. Anastassiou of Columbia University in New York. Using switched capacitor techniques, both positive and negative weights may be realized depending on the clocking scheme used on each capacitor (see Figure 22). The weights may be stored either by using RAM to select binarily weighted capacitor, by loading the RAM weight value as the initial value for a count-down

counter controlling how many times a single capacitive weight is clocked, or by using the RAM to program a clock divider whereby capacitors clocked at different rates can approximate conductance values. No implementation has been reported using this scheme.

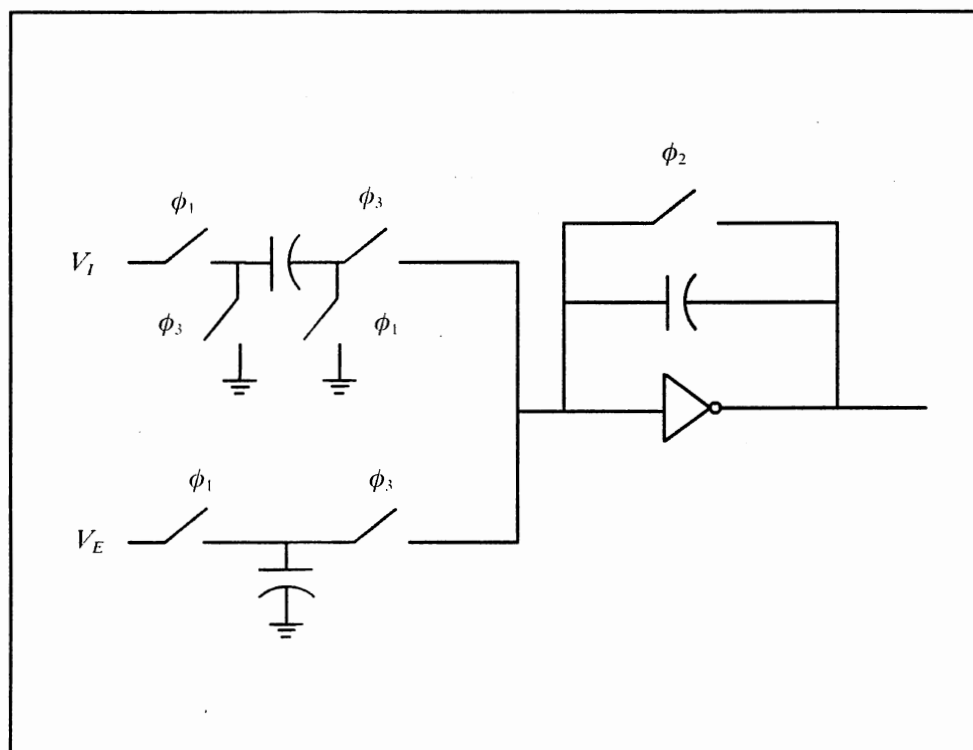


Figure 22. Switched Capacitor Neuron

Digital Neural Networks

The main disadvantage to using analog electronics to realize artificial neural networks in electronic form is that device parameter variation introduces some inherent error and unpredictability into the design. An all digital network would not suffer from the same limitations, trading speed, and possibly compactness, for accuracy, reliability, and ease of programmability. The following section of

this report describes three attempts to realize artificial neural networks using well understood and easily designed digital electronics.

A simple and easily understood example was presented by John Cleary of the University of Calgary in 1987. His VLSI implemented 'sigma-chip' used multiplexed inputs and output to realize 100 neurons with 1000 one bit input weights apiece on a single chip. Using multiplexors, 1000 bit shift registers, AND gates, counters, and a digital threshold (Boolean logic designed to give an output true for input words greater than a certain level), Cleary successfully designed and fabricated the chip out of two micrometer CMOS. The chip contains 450,000 transistors and can evaluate the state of the network every 2.5 milliseconds, the outputs of each neuron appearing serially at the output pin every 25 microseconds.

The problem with the above design is, of course, that the weights are only single bit, severely limiting the applicability of the network. Van Den Bout and Miller of North Carolina State University have solved this problem using a digital stochastic architecture. Similar to the multiplexed CCD analog described earlier, the network uses circular registers to multiplex the computation of the weighted sum of inputs to each neuron. The interesting part of the design is their scheme for using stochastic multiplication. As one can see in Figure 23, two random number generators are used. One comparator gives a high output pulse if one random number is greater than the weight currently being multiplied. The other comparator pulses if its random number is greater than the output state currently being multiplied. These two pulse trains are fed into AND gates which in turn drive the increment counter.

This control signal is a pulse train with the number of pulses proportional to the product of an output state and its associated weight (see Figure 24 for an example of stochastic multiplication). A second control line from the weight regis-

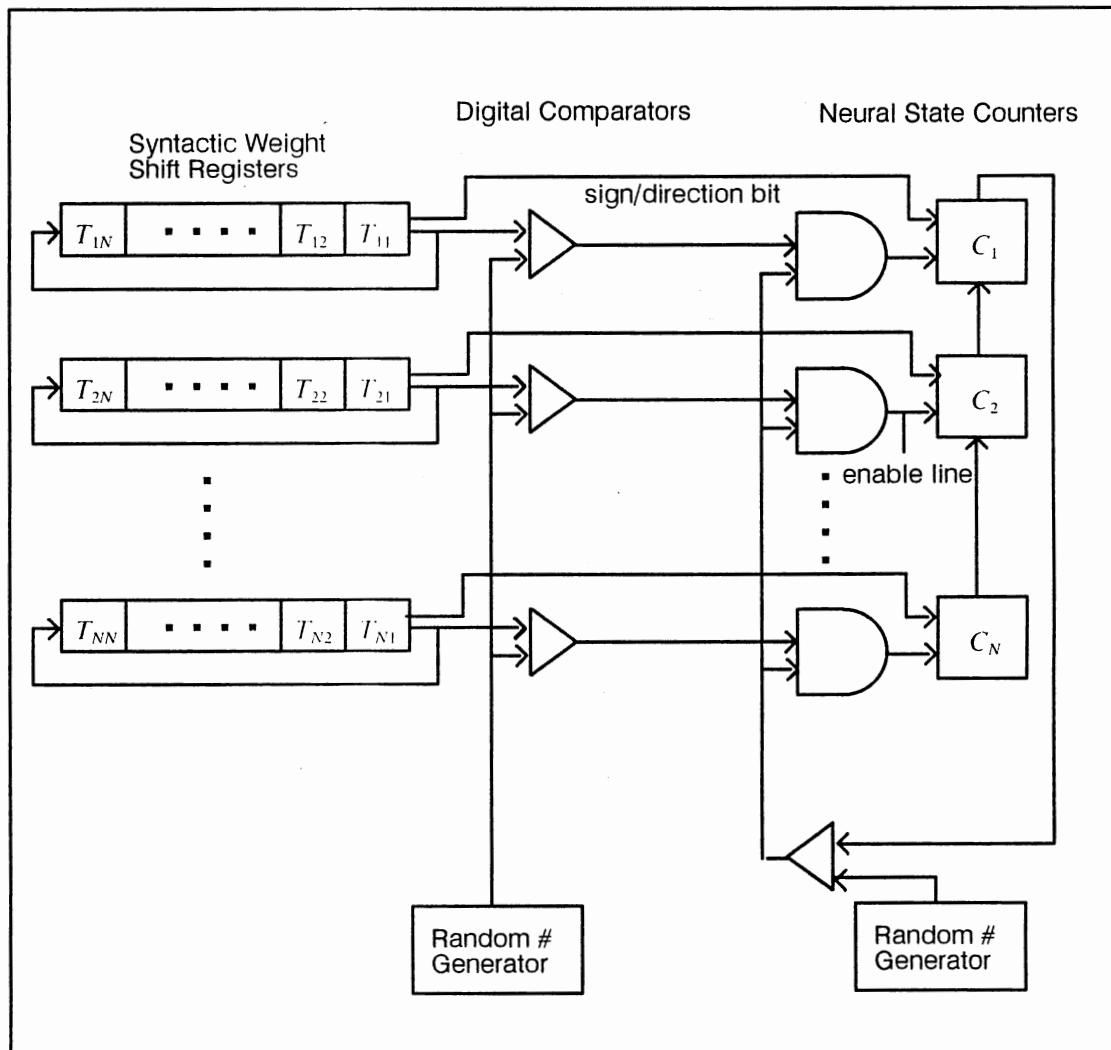


Figure 23. Stochastic Network Architecture

ter to the counter controls whether the increment line causes a count-up or count-down to occur, i.e. whether the weight is excitatory or inhibitory. The resulting synchronous operation is the discretized approximation of Hopfield's equations including neurons featuring graded response, saturation, and response to the integral of the weighted sum of the inputs. By adjusting the probability distribution function of the random number generator, high gain, low gain, and sigmoidal neurons can be realized. The graphs in Figure 25 illustrate the tracking between an analog network and the digital stochastic network.

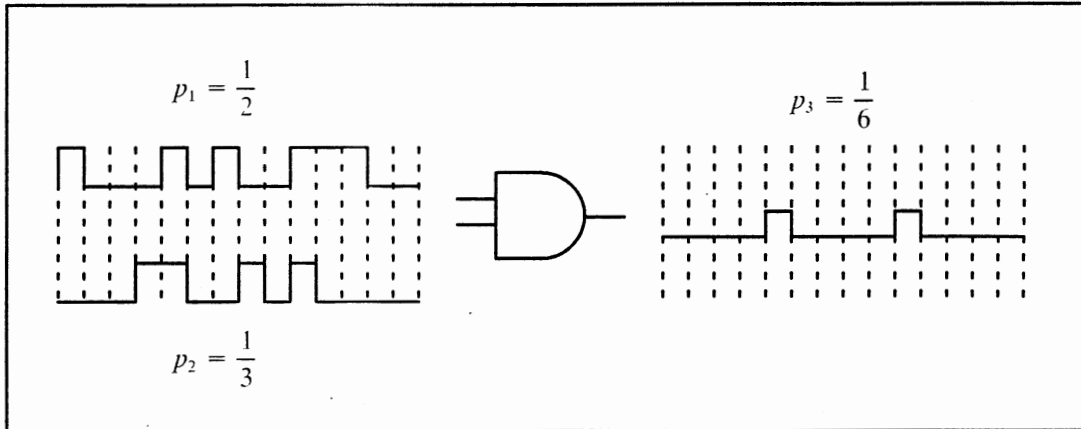


Figure 24. Stochastic Multiplication Example

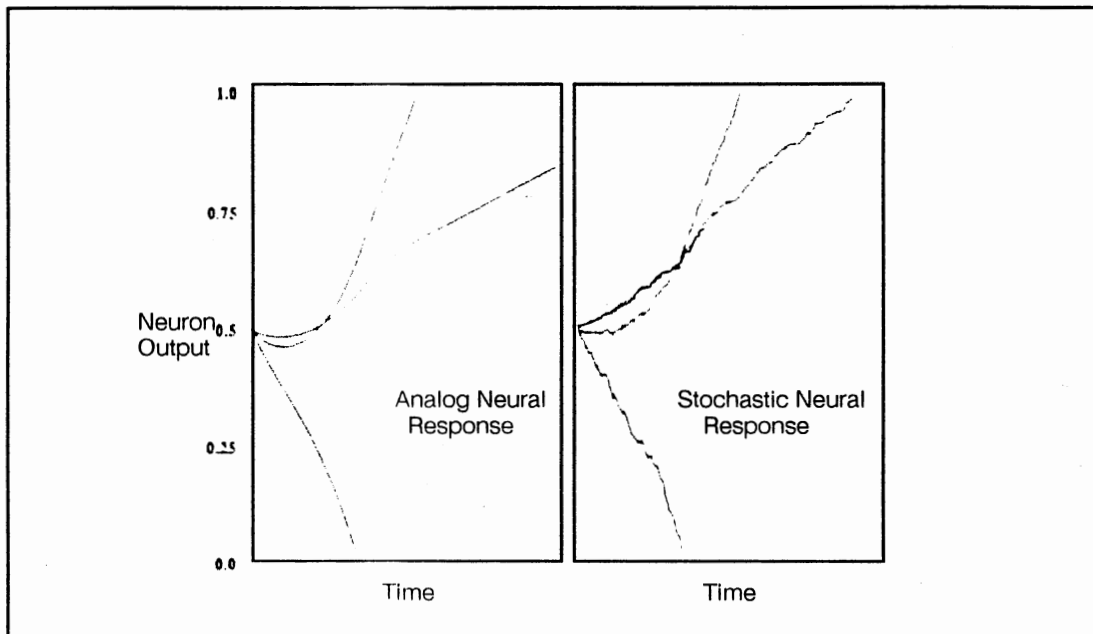


Figure 25. Response of Stochastic Network

Response times for the network are expected to be on the order of milliseconds. A prototype chip has been fabricated and tested and a more advanced version with off-chip weight storage and learning capability is currently being designed.

An architecture based on radically different principles was presented by Mahmoud K. Habib and H. Akel of Kuwait University in May 1989. Their digital neuron-type processor mimics to some extent the behavior of biological neurons. The state variables of the system are the frequencies of the input and output pulse trains and neural processors respond to the weighted temporal and spatial summation of the input pulses in an all or nothing fashion. The weighting takes place in the form of extending the length of the input pulse. Figure 26 illustrates the behavior of the neuron-type processor in response to spatial, (a), and temporal, (b), inputs.

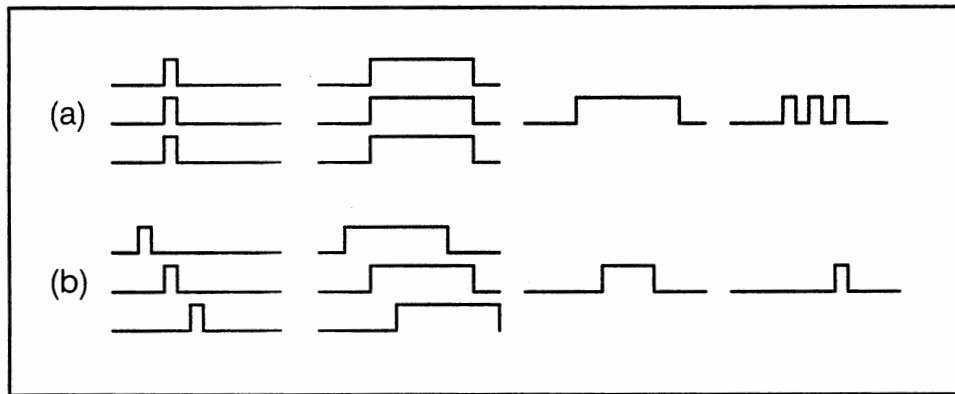


Figure 26. Neuron Type Processor Pulse Response.
(a) Spatial. (b) Temporal.

From left to right one can see the input pulse trains, the outputs of the weighting function, the sum of the weighted inputs and the output pulse train. While this design is interesting, it doesn't immediately lend itself to any of the common neural system designs and only one weight value is available. If the circuit could be altered to achieve a programmable weight, some interesting behavior could result.

Miscellaneous Neural Designs

The following section contains summaries of four designs which, for one reason or another, didn't really belong in either of the two preceding sections. They include an analog simulation of a biological neuron, a suggestion to implement automatic scaling of a neuron size and gain with network size, and two learning systems based on the error backpropagation learning rule and Hebbian simulated annealing.

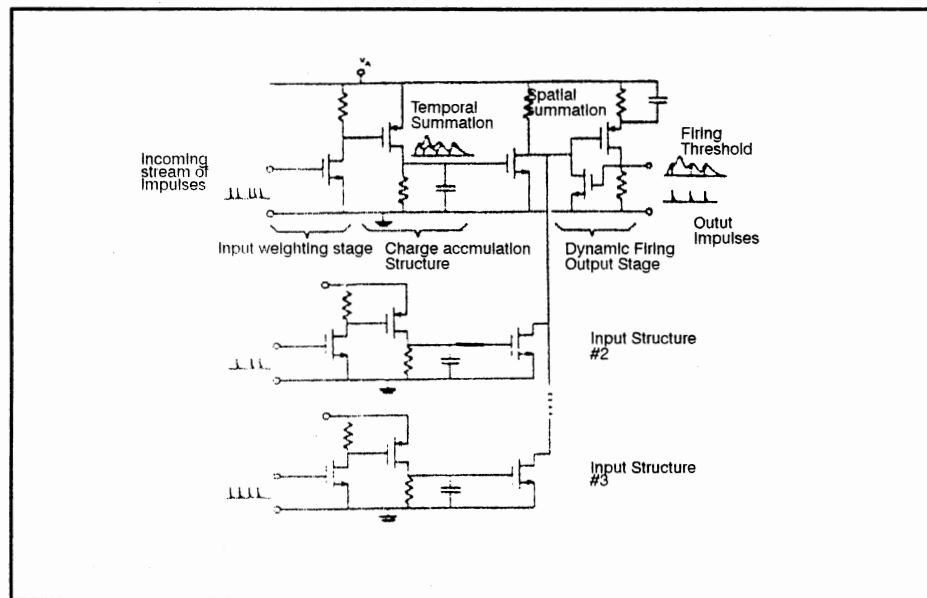


Figure 27. Neuron Type Junction

The circuit in Figure 27, below, was suggested in May 1987 as an analog electronic model for the behavior of an actual biological neuron by H. El-Leithy, R. W. Newcomb, and M. Zaghlour of George Washington University. It responds to spike trains, exhibits weighted temporal and spatial summation, and graded, sigmoidal type response. Like an actual neuron, the MOS Neural Type

Junction outputs a single spike when its input charge exceeds a certain threshold level. This input charge is accumulated by the addition of small weighted pulses from the input weighting stage. This results in a charge which is then added through the spatial summation stage to the other temporal charge sums in the form of current drawn through a summing resistor. The width of these small weighted pulses determines the minimum output firing frequency and the recovery time of the output oscillator determines the maximum firing frequency.

The authors simulated their circuit on Microcap and the results of that simulation (shown in Figure 28) indicate that the circuit does emulate the behavior of a biological neuron, at least for a single input.

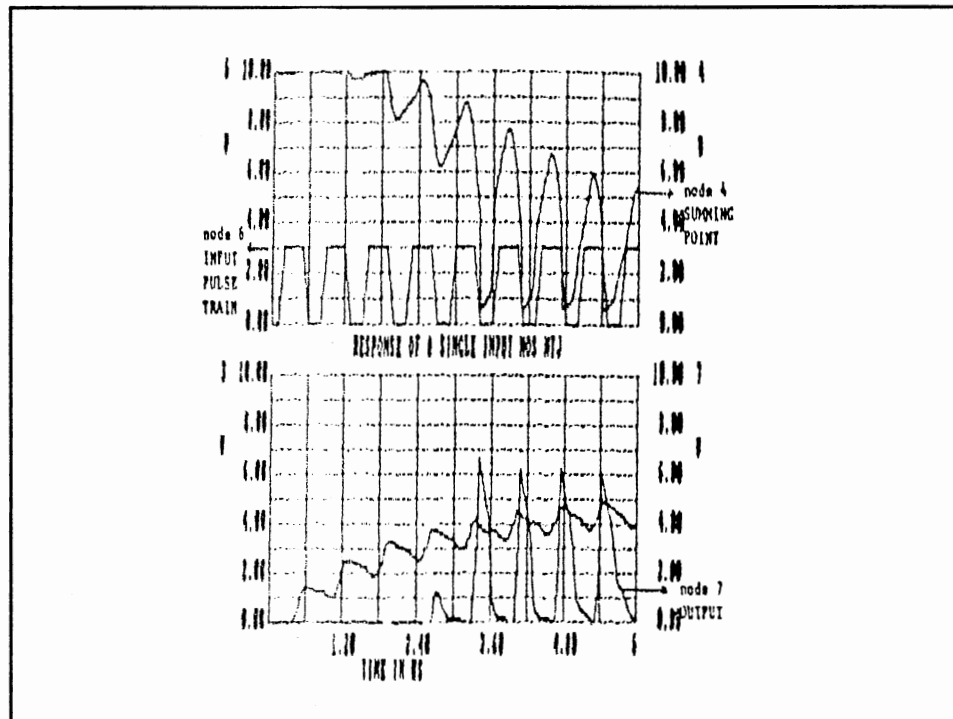


Figure 28. Single Input Neural Type Junction Simulation Results

The main problem with this circuit is that it complicates the problem. Yes, the state variable is the output firing frequency, just like in real neural networks, but that frequency has to be converted into a voltage and weighted through a conductance just like in normal analog networks. Also, the weighting is accomplished by modulating the substrate voltage, which on a chip is common to all transistors of the same channel type. This means only two magnitudes of weighting are available at a time and one is excitatory while the other is inhibitory. Finally, even if a way to isolate the substrates could be found, a capacitor to store the voltage would be subject to large discharge rates from the leakage current into the reversed biased diode junction from the substrate to the source.

Satyanarayana and Tsividis of Columbia University proposed an automatically scaling neuron-synapse pair called a distributed neuron. By including a small neuron with every synaptic weight, the network becomes very easily reconfigurable. All one needs to do is connect the weights into the desired topology and at every node, the weights automatically form a neuron with an appropriate gain and fan-out, assuming the demand on all the neurons in the network is going to be about the same. A chip based on these design principles is being fabricated using 0.9 umeter CMOS technology achieving 1024 reconfigurable neuron-synapses on a 7mm X 4mm chip. The synapses are four-quadrant multipliers with the weight stored on one input capacitance. The distributed neuron is a simple two transistor inverter load. The chip has been optimized to achieve seven bits of accuracy.

The Backward Error Propagation learning rule has become something of a watershed for adaptive learning networks in the seven years since its rediscovery by Rumelhart and McClelland in 1985. A hardware implementation of the back-propagation algorithm (as it is called) would be invaluable for the application of neural systems to real control or pattern recognition problems. In 1987, B. Fur-

man and A. A. Abidi of the University of California proposed a scheme for the practical hardware realization of the backpropagation algorithm with the following features: a single chip realizes a single layer of the feedforward network making multi-layer networks straightforward to construct, a four quadrant analog multiplier performs the multiplication of forward and backward propagation signals by analog, capacitively stored weights and by a continuously variable learning rate parameter, the sigmoidal transfer function is width programmable, the derivative of the sigmoid is realized by a separate circuit reducing the math involved in the algorithm, and finally, that forward and backward propagating signals be multiplexed on the same pin as voltages and currents to conserve chip i/o. The authors have some good ideas and provide some circuit diagrams for their proposal, but no actual fabrication plans were discussed since many of the component blocks of the system were yet to be designed or optimized. In the two years since the publication of their proposal, nothing more has been heard concerning their progress.

The final design to be discussed in this literature review is perhaps the most impressive of all. Published in the Proceedings of the 1987 Stanford Conference on Advanced Research in VLSI, the VLSI learning system designed by Joshua Alspector and Robert B. Allen of BellCore incorporates many of the ideas discussed in this survey. The design is a trainable Boltzmann machine. The Boltzmann machine is much like a Hopfield network except that the output voltage is high based on a sigmoidal probability curve rather than deterministically depending on the input voltage.

This is realized in analog electronics by the circuit in Figure 29. As can be seen, the differential amplifier providing the sigmoidal nonlinearity has an extra pair of inputs. These inputs are driven by the complementary outputs of a variable gain noise amplifier. In this way, a noise voltage, or jitter, is added to the de-

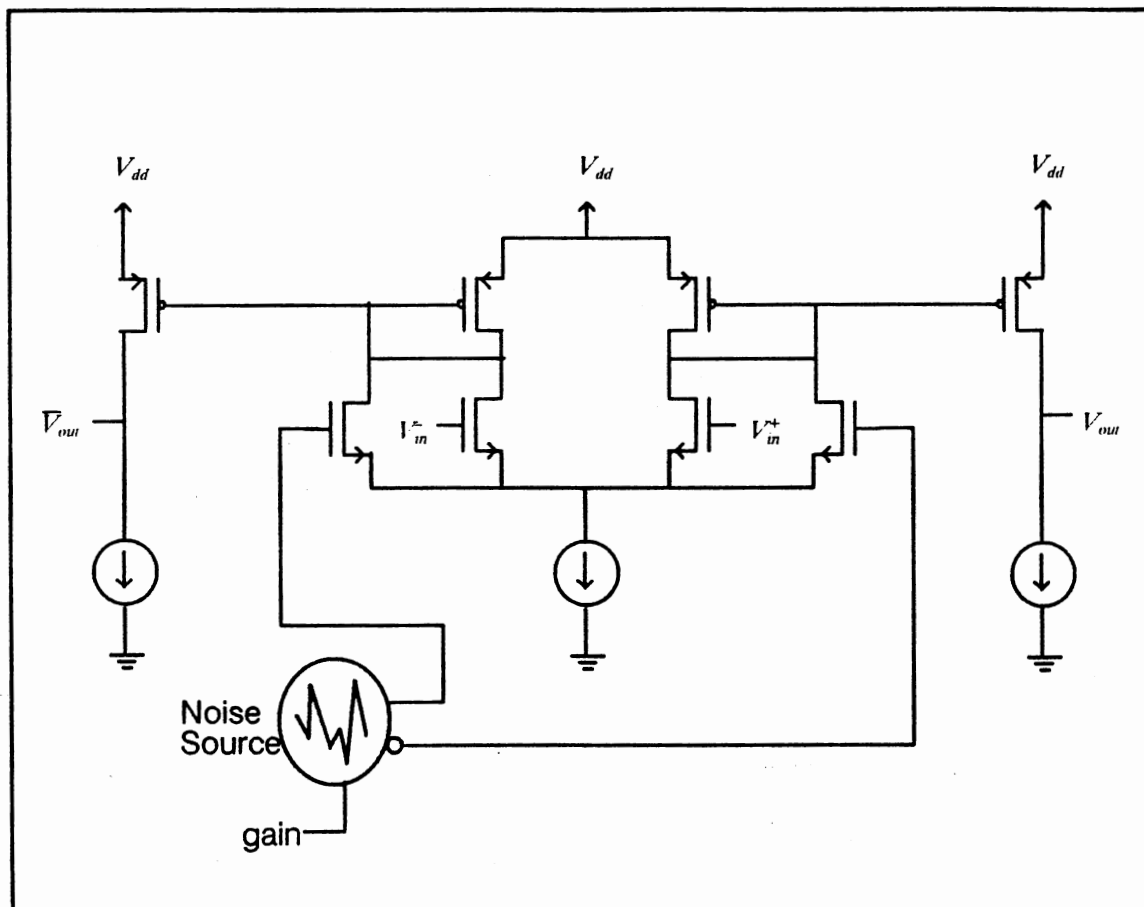


Figure 29. Four Input Boltzmann Neuron

terministic output voltage. It can also be viewed as a means to stochastically vary the bias current off the difference amp, making the transconductance of the amplifier behave noisily. The noise amplifier is variable gain for simulated annealing. By starting the network out with a very high level of noise injected into the system and gradually tapering it off, a stable state corresponding to a global energy minimum will be reached most of the time. The weights of the network are realized as five bit RAMS driving binarily weighted conductances joining one neuron's output to another neuron's input the original Hopfield inspired style. In addition, each weight has an associated weight processor which is used during training to increment or decrement its associated weights.

The training is conducted by alternately forcing the inputs and outputs of the network to the desired pattern associations and then letting the outputs of the network relax to what the weights dictate they should be. Simulated annealing is employed in both phases to avoid getting caught in local minima. The weight processors accumulate the states of the neurons a particular weight is connecting in both phases and increments or decrements the weight by an amount proportional to the difference between the number of times both neurons had the same state in the forced phase and the number of times they both had the same state in the relaxed phase. This simply implements the Hebbian learning rule stochastically.

The authors demonstrate the learning ability of the chip design in several simulations, the results of one of which is shown in Figure 30. By varying parameter of the network, the authors were able to determine the number of units and the learning rules best suited to certain standard pattern classification problems. The curve shown is for XOR, at traditionally difficult classification problem for a machine to learn.

This section has summarized the works of many different researchers in the field of electronic neural networks including all digital networks, all analog networks, and digital/analog hybrids. Some of the designs mentioned appear very promising and merit further development, while others appear to have very little practical use at all having been investigated for purely academic purposes or for experimental evaluation of early neural ideas. One thing is certain, hardware neural networks will be an essential development if artificial neural systems are to fulfill their early promise of usefulness and applicability.

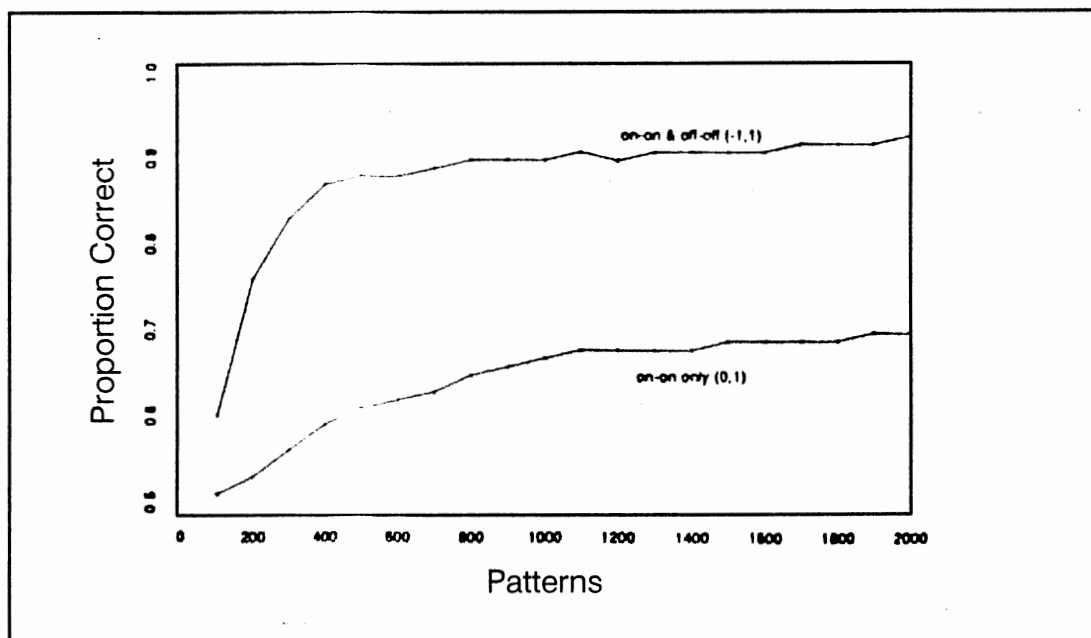


Figure 30. Boltzmann Machine Simulation Results

Example Applications of Neural Systems:

How that efforts toward realizing neural system building blocks in hardware have been reviewed, it would be useful to consider some example applications of neural systems. Since the problem that this thesis is addressing, median filtering, is commonly applied to images, the examples presented below will be concerned with image processing. They could be applied to any type of digital signal processing.

Some common image processing schemes may be decomposed into weighted sums and thresholds and are therefore ideally suited to a neural architecture. These include: thresholding; intensity level slicing; spatial averaging; high-, low-, and band-pass filter; gradient approximation; and linear transforma-

tion. Some less obvious applications of neural networks are intensity mapping, image recognition, and vector quantization.

Thresholding may be accomplished neurally simply by feeding an image into a bank of neurons equal in number to the the pixels in the input image with each input pixel only connected to the neuron "above" it in the network with unity weight. The neuron outputs form the thresholded input image. Figure 31, below, illustrates a number of thresholding, intensity selection functions and their neural implementations.

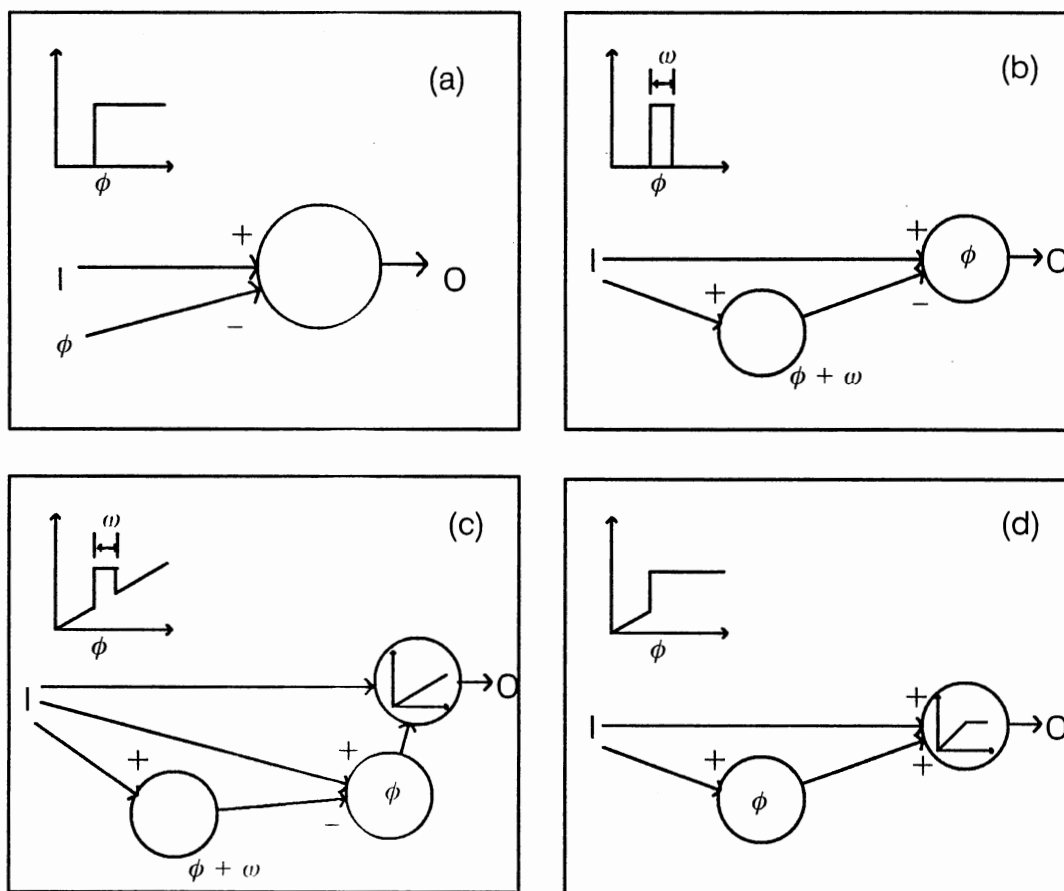


Figure 31. (a) Thresholding, (b) Intensity Level Slicing, (c) Intensity Level Slicing w/Background, (d) Thresholding w/Background

Some of the most common operations in image processing are spatial averaging, FIR filtering, and gradient approximation. All of these operations involve

replacing a pixel in the output image by a weighed sum of a neighborhood of pixels in the input image. An output plane of neurons with linear transfer functions and weights coming to each of them from a neighborhood of input pixels could perform any of these functions. For example, Figure 32a shows three discrete approximations of the Laplacian operator, useful in edge detection, while Figure 32b shows how one of those operators would be implemented with a neural network.

Any linear image transformation (i.e. Fourier) could be mapped onto a neural network in a similar manner except that the inputs would be fully interconnected with the output plane of neurons and each neuron's weight pattern would be unique, representing the basis function associated with the particular location in the output plane.

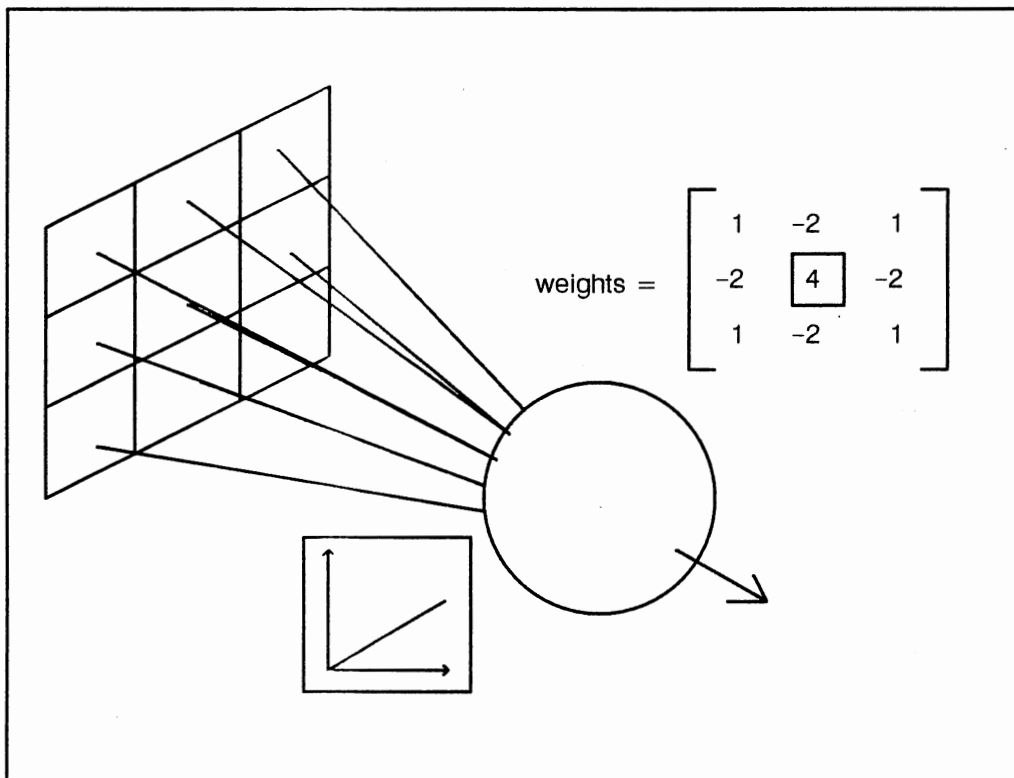
Intensity mapping is useful for compressing, expanding, or enhancing images. As its name implies, intensity mapping is simply replacing every pixel intensity in an image by another intensity dictated by a mapping function. Figure 33 shows several such functions. These single input, single output functions are also realizable using neural networks.

It has been shown that a Backward Error Propagation network is guaranteed to learn to approximate almost any function to any desired degree of accuracy as long as the network meets certain architectural requirements. Initial publications by Hecht-Nielsen suggested that a relatively small network with one hidden layer and a fixed number of processing units could learn to approximate practically any function to within any desired level of accuracy.

However, more recent results have shown that a Backward Error Propagation network can approximate any function to within a desired error only given that enough hidden units are supplied. The more accurate the approximation desired, the more hidden units are necessary.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 6 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

(a)



(b)

Figure 32 (a) Three Laplacian Operators, (b) Neural Laplacian

For simple intensity mapping functions, a small network might suffice (see Figure 34). This small network would then be trained using the generalized delta rule and examples of the required intensity mapping. Each pixel's network would be identical.

Although it is beyond the scope of this report to cover them in any detail, there are many ways to employ Back Propagation networks to recognize crude

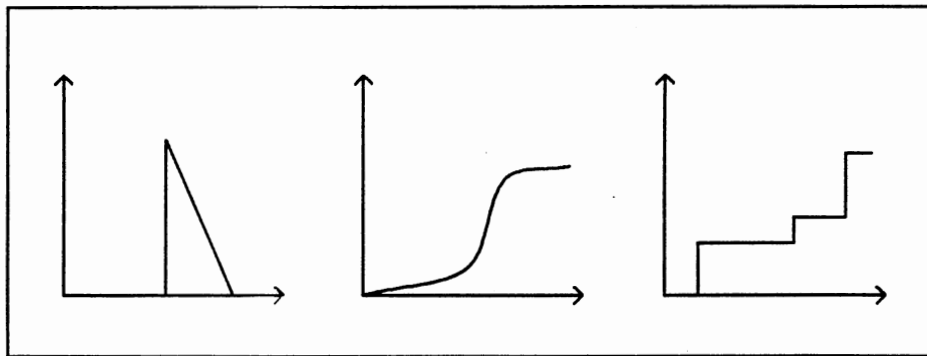


Figure 33. Three Example Intensity Mapping Functions

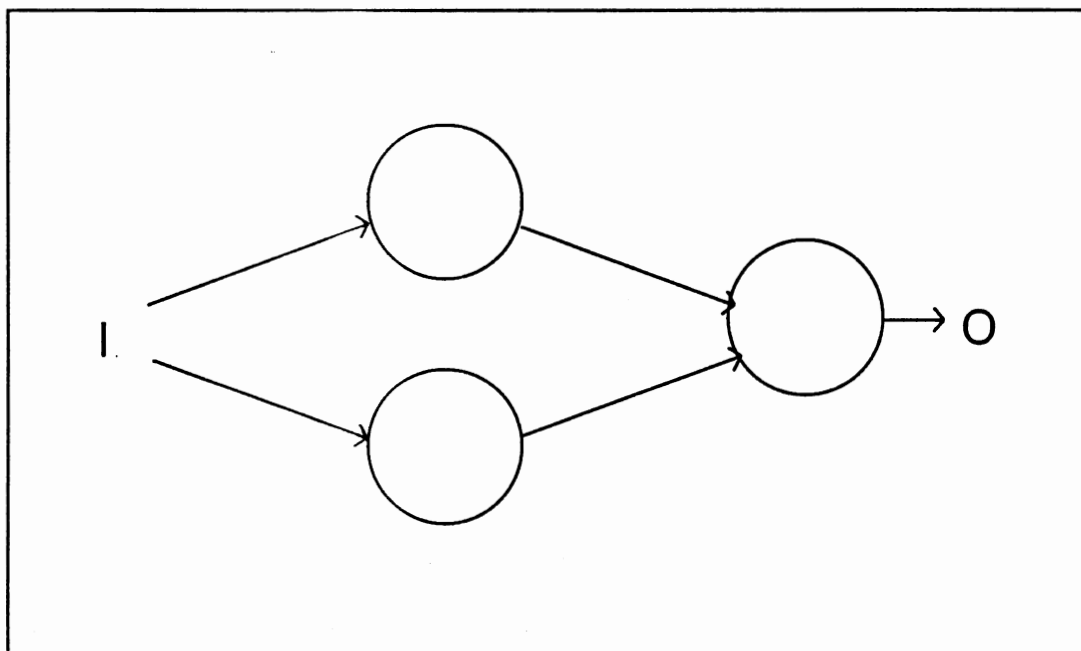


Figure 34. Example Back Propagation Architecture for Intensity Mapping

images. Their main advantage, their train-ability, allows a generic network architecture to apply to any number of image or pattern recognition problems. This trait also limits them, for unless the training set of images is very carefully assembled, an image not explicitly among the training images could provoke unpredictable responses.

On a higher level of complexity, Kohonen networks, named for their creator, are also useful for image processing in that they can be used as a vector quantizer. Operating under very subtle rules, Kohonen networks can be trained to categorize input vectors while maintaining their topological relationships. In other words, two vectors that are similar in the input space will provoke responses from the net that are similar in the output space, an occurrence not at all guaranteed by other networks.

This section has summarized several rudimentary ways that neural networks can be applied to image processing. The following section will focus on a specific proposal for applying neural system-like architecture to the parallel median filtering problem.

Proposal for Neural Implementation of Median Filtering

The Basic Architecture and its Operation

Figure 35, below, shows the proposed architecture for a neurally inspired median filter. A window of N input pixels, P_{i0} through $P_{i(N-1)}$ where N is an odd number, are fed into a bank of summers. The outputs of these summers are in turn fed into a bank of piecewise linear thresholding units. The outputs of these units are then summed and integrated resulting in an output value M_{out} . This value is initially an arbitrary estimate of the median of the input window. M_{out} is fed back to the inputs and subtracted from them through the initial bank of summers.

The output of this network will converge to the median value of the input window regardless of its initial value. Equation I.1 describes the dynamic behavior of M_{out} .

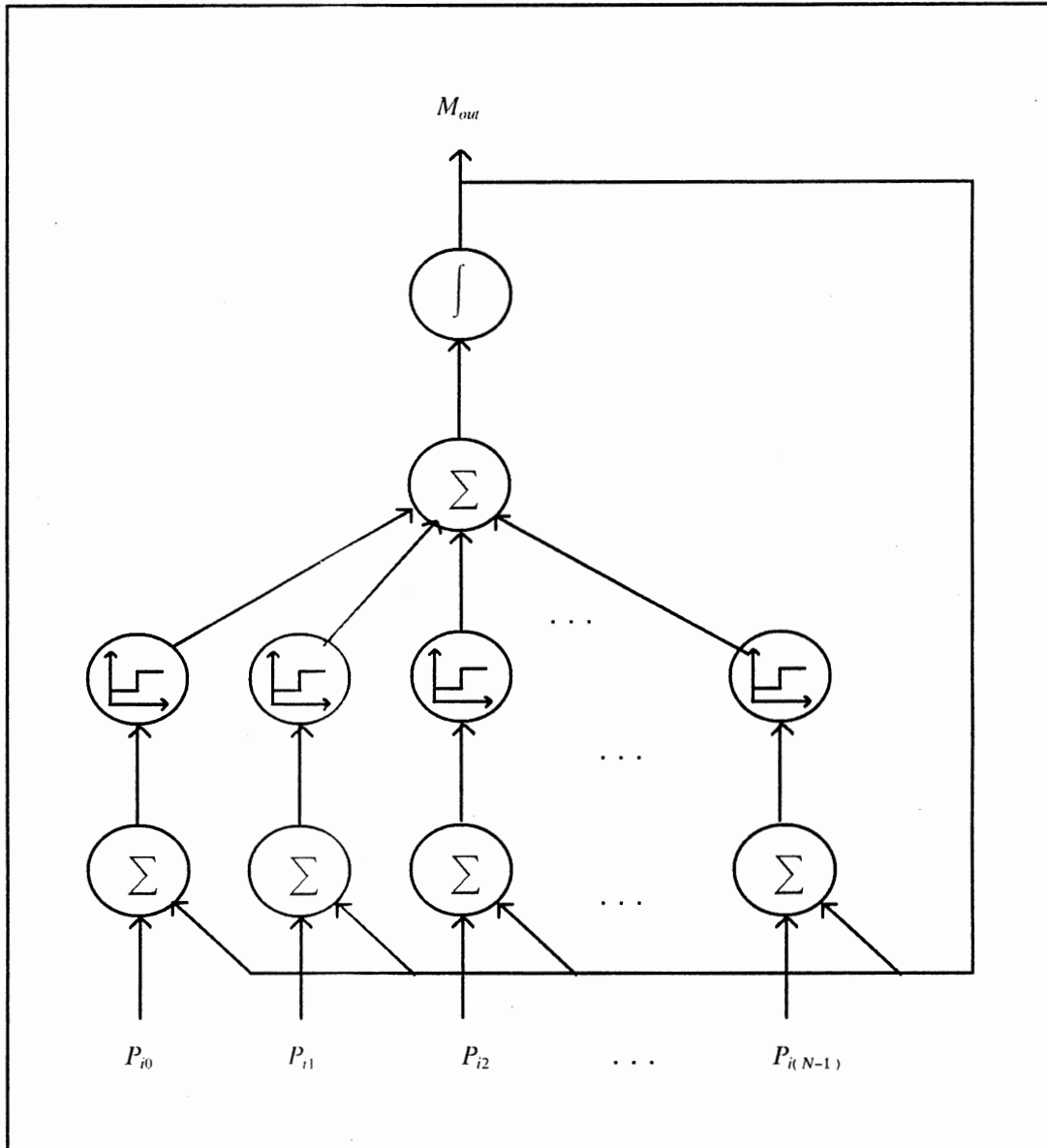


Figure 35. Neural Median Filter Architecture

Equation 1.1

$$\frac{dM_{out}}{dt} = \sum_{n=0}^{N-1} \text{sign}(P_{in} - M_{out}).$$

The output of the network is initialized to an arbitrary value which is compared to all the values in the input window through the bank of summers and piecewise linear units. The resulting sum sets the derivative of M_{out} to be negative if the majority of the input pixel values are less than the initial output value

and positive if the majority of the input sample values are greater than the initial output value. In this way, the network's estimate of the median is continually corrected until there are as many input values greater than the output as there are less than the output. At this time, the network reduces to the simple negative feedback system shown in Figure 40 whose behavior is described by Equation 1.2.

Equation 1.2
$$\frac{dM_{out}}{dt} = A_V(P_{in} - M_{out}).$$

Quite obviously, the output comes to a rest when it equals the input value. This value will be one value in the input window with as many sample values greater than itself as there are sample values less than itself, the median. Figure 36, below, shows a sample convergence curve. The C program used to simulate the network and generate the curves below is appended to this thesis. In most cases, the output of the network converged to the correct value of the median. The errors and their causes will be examined in greater detail in Chapter IV of this thesis.

Design Proposal for System Evaluation

This system, while not explicitly a neural network, is clearly inspired by neural network principles. The system is a conglomerate of simple processing elements, weighted sums, thresholds, and integrators, and it uses system dynamics to gain computational power and converge to the solution.

As a neurally inspired system, it is very simple to make equations between system building blocks and electronic components and building blocks. The bank of summers and threshold units can be easily realized by an analog voltage comparator, basically a differential amplifier which has already been shown to be a popular choice for neural system implementations. Since only unity weights are

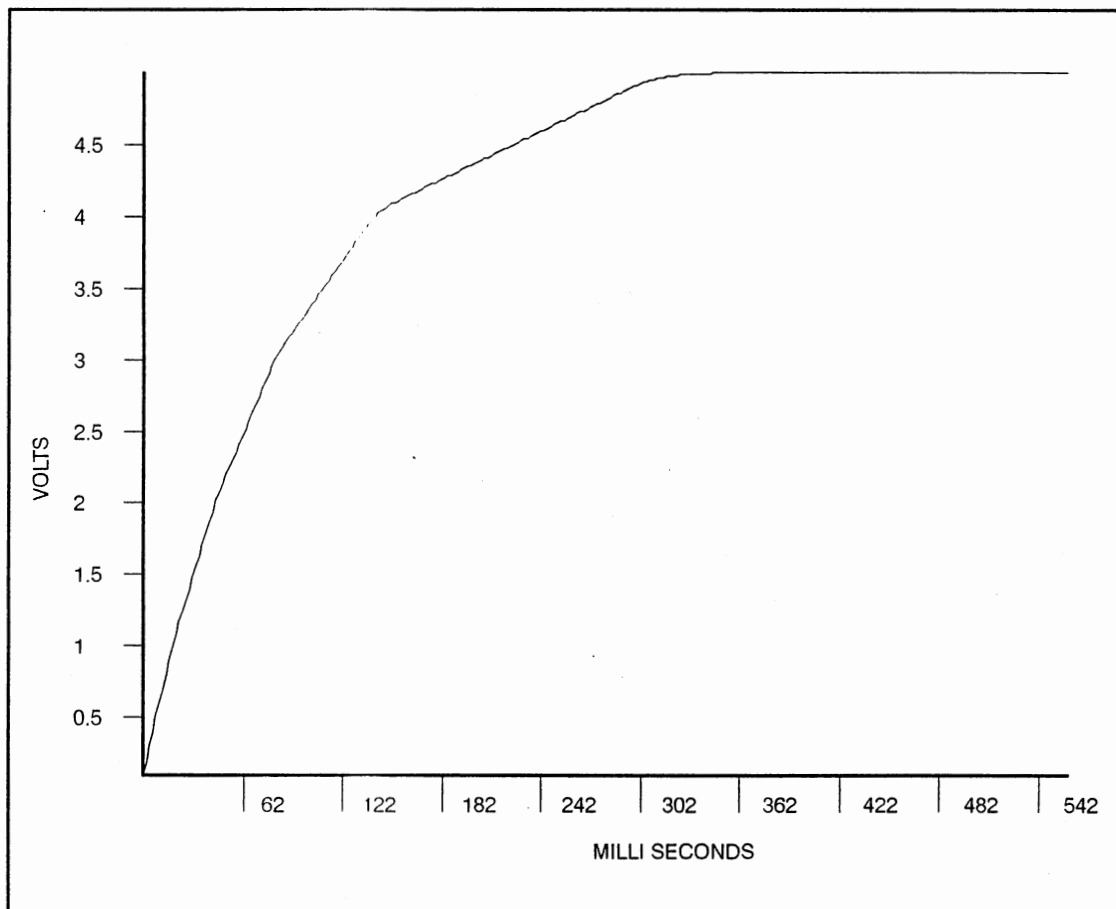


Figure 36. Sample Neural Median Filter Convergence Curve

needed, of positive and negative sign, and since each sum is only of two inputs, the differential inputs on the comparator will suffice for all weighting. The output summer and integrator can be realized by a capacitor if the output variable of the comparator is current rather than voltage.

It is easily seen, then, that the main obstacle to implementing this system is the design of a differential voltage comparator whose output variable is current rather than voltage. The remainder of this thesis is devoted to summarizing a proposed design to remove this obstacle and evaluating the proposed comparator's effectiveness in the context of the neural median filter system.

Chapter II, next, will describe the design in detail and the design's resulting performance. Chapter III will summarize efforts toward modeling the design's behavior for evaluation on a system level, after which, Chapter IV will summarize the results of the system level simulations. Chapter V will offer conclusions based on these results and suggestions for future work connected with investigations of this proposed neural median filter system.

CHAPTER II

A PROPOSED COMPARATOR DESIGN

Introduction

This chapter describes a proposed design for an analog CMOS comparator to fulfill the function of the comparison units described in Chapter I. After discussing the specifications the design intends to satisfy and the methods chosen to meet those specifications, the chapter will present the finished design with approximate hand calculations and their correlation with Spice simulation results.

General Design Requirements

In order for the neural median filter to function accurately and usefully, the component comparators should exhibit certain characteristics. They must converge under unity feedback conditions. They must have a high gain to minimize the decision region width. They must have a high output resistance to approximate pure integration closely. They must have high output current rails to converge to the final answer speedily. They must have a high common mode rejection ratio to minimize common mode error. They must have symmetric output characteristics so that charging and discharging currents will cancel each other out during the convergence. They must have a low offset voltage to minimize threshold error. They must be designed in an accessible and fully characterized process with common breakdown voltage characteristics to keep compatibility with existing systems.

Meeting the Design Requirements

Analog CMOS was chosen as the technology to make the resulting systems compatible with most digital signal processing architectures. The ORBIT process in particular was chosen for its availability to educational institutions.

The high output resistance and symmetric output characteristics suggest a complementary, common-source, Push-Pull output stage like the one shown in Figure 37 below. The devices must be wide for high current output and long for high drain resistance. Also, their W/L ratios must be proportioned appropriately to insure that the output charging current closely matches the output discharging current.

In order to boost the gain, two parallel input gain stages, one n and one p, were designed to drive the output stage. They are of course differential amplifiers as shown below in Figure 38. The device sizes were chosen to maximize gain and to tailor their DC output characteristics to gain maximum benefit from driving the output stage simultaneously. Their n to p ratios were kept constant to force the DC transfer functions to track closely with another although certain widths were adjusted to position the transfer functions on the horizontal input voltage axis, adjusting the offset voltage.

These differential amplifiers were biased with the cascoded current sources shown below in Figure 39. This type of current source was chosen for its high output resistance. This feature was needed to satisfy the requirement for a high common mode rejection ratio. The sizes were chosen to insure that the two amplifiers had closely matched bias currents preserving the effectiveness of ratioing the n and p sizes.

The near constant ratio of $W_n L_p / W_p L_n$ between the two halves of the circuit insures that their bias characteristics will be similar and predictable and that their small-signal characteristics will be almost identical. By attempting to cor-

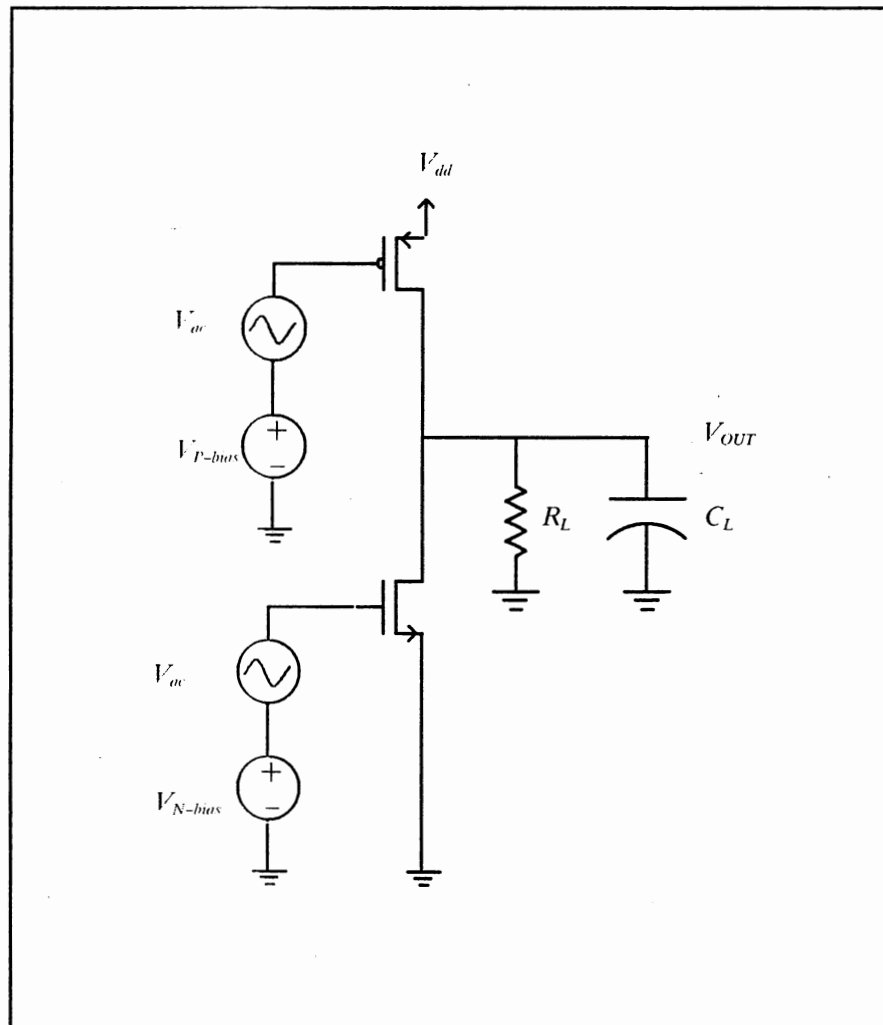


Figure 37. Complementary, Common-Source, Push-Pull Output Stage.

rect differences between n and p transistors through W/L ratioing, a symmetric output characteristic is much easier to achieve. The ratio between their lengths was chosen to achieve an identical output conductance. The ratio between their widths was chosen to achieve identical transconductances. An attempt to calculate these ratios by hand was made. However, the second order effects proved to be too complicated to predict other than by Spice simulation. The p to n length ration was found to be 1.5 and the p to n width ratio was found to be 3.75 with the exception of the output stage which has a width ratio of around 5.5. This

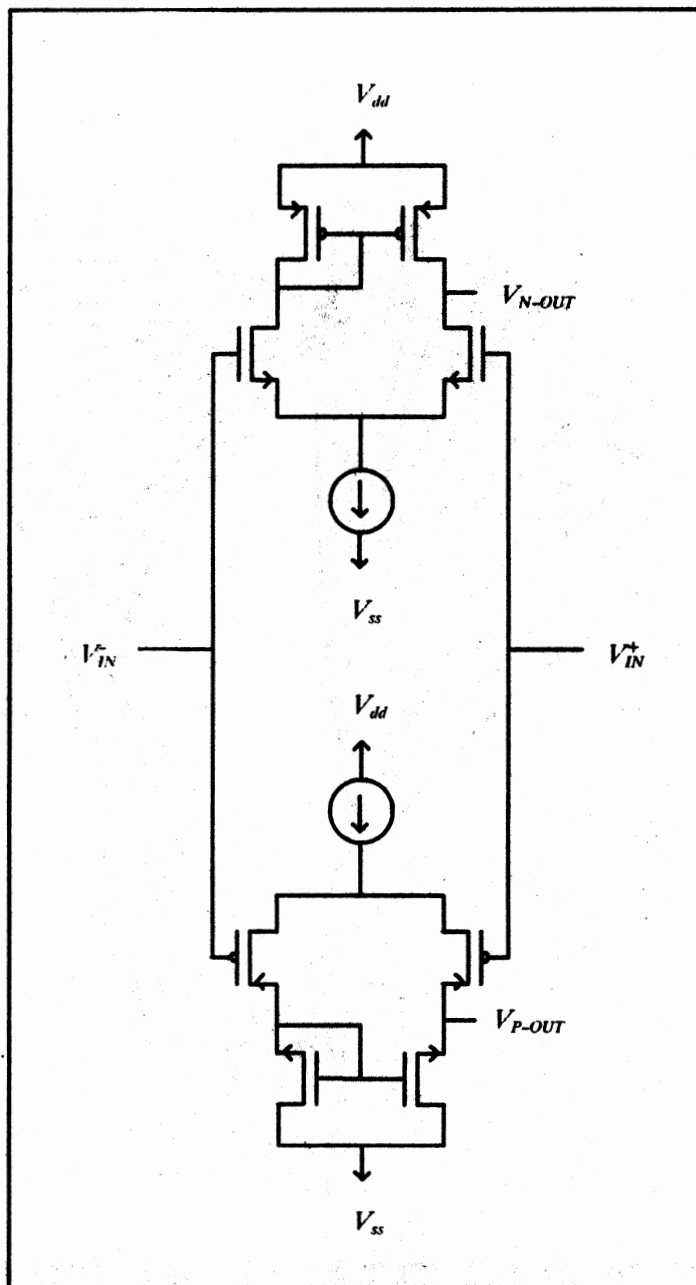


Figure 38. Complementary, Parallel, Differential Amplifiers

was necessitated by second order effects as the operating conditions of the output stage differed considerably from those of the differential amplifiers. The result of all this was that the beta ratios came out to be unity in most cases making the behavior of the circuit fairly symmetrical over the supply voltage range.

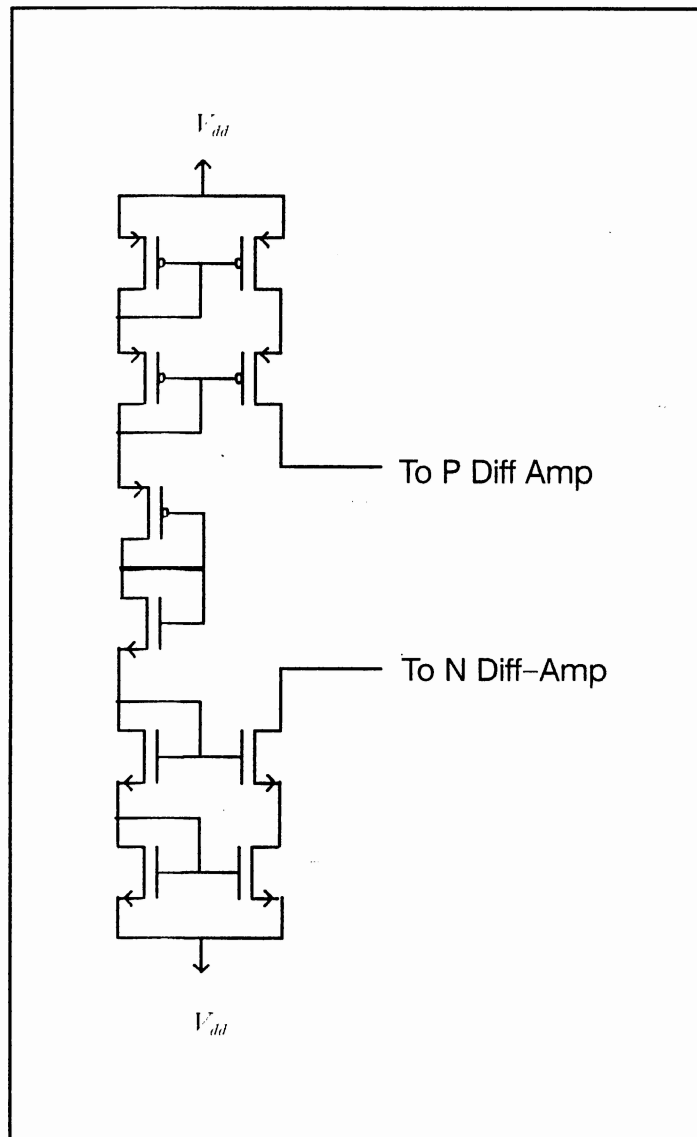


Figure 39. Cascoded Current Sources.

The final circuit design, annotated with widths and lengths and the compensatory Miller capacitors, is shown below in Figure 40.

Circuit Analysis

The entire spice analysis of the circuit is reported in Appendix A. The main characteristics of the design are detailed in Table I below.

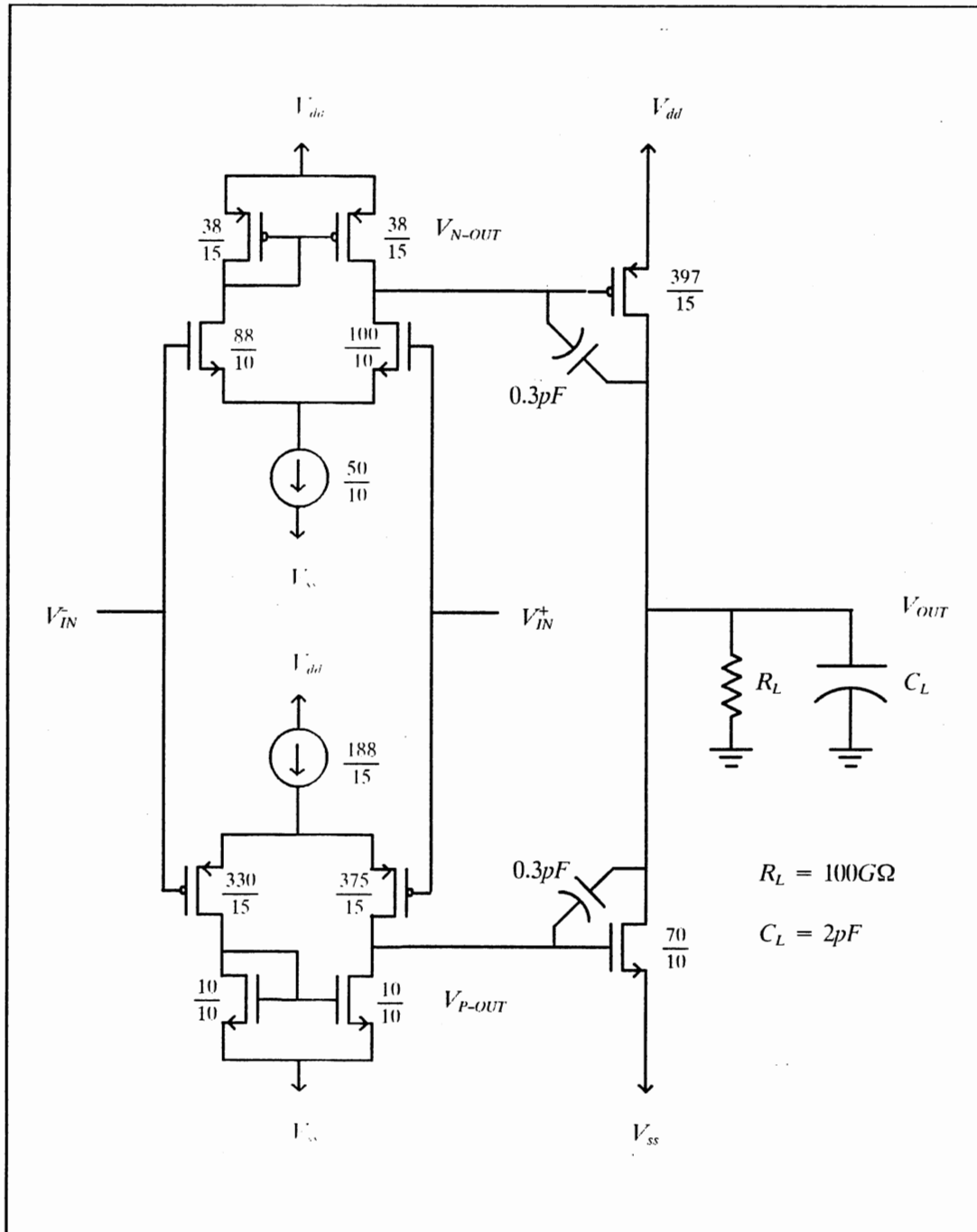


Figure 40. Final, Annotated Circuit Design.

Operating Point and Hand Calculation of Gain and Bandwidth

Figures 41 and 42 below show the Spice calculated operating point of the cascoded current sources and the differential amplifier and output stage, respec

TABLE I
SPECIFICATION FOR THE FINAL PROPOSED
COMPARATOR DESIGN

Characteristic	Value
N diff amp bias current	9.93uA
P diff amp bias current	10.20uA
Output stage bias current	2.11mA
Total Transconductance	594 mA/V
Output Resistance	28.9 kOhms
Voltage Gain	17,180 (84.7Db)
Output voltage rails	+/- 5V
Output current rails @ zero output volts	+/- 4.05 mA
Output current rails absolute	+ 4.16mA / - 4.2mA
Input capacitance	+ 26.2pF / - 3.02pF
Output capacitance	2.827pF
CMRR	1365 (62.7Db)
Rolloff frequency	1.644 kHz
GBW-product	28.24 MegaHz
Gain margin	10Db
Phase margin	30 degrees
Estimated Slew Rate	3.3nsec (81nsec by T)
Estimated Internal Delay	60usec
Input offset voltage	.425 milliVolts
Output voltage @ .425mV input	.103V

tively, in terms of their capacitances, transconductances, and output resistances. Figures 43 and 44 show the equivalent small-signal load resistances for each stage.

Figure 45 below shows the resulting gain for each node in the path from $V+$ to V_{out} .

Figure 46 below shows the effective load resistances and capacitance after Miller effects have been considered. The inputs to the output stage show two calculations for the pole locations, corresponding to the two possibilities that either of the two poles rolls the response off before the other one, resulting in a

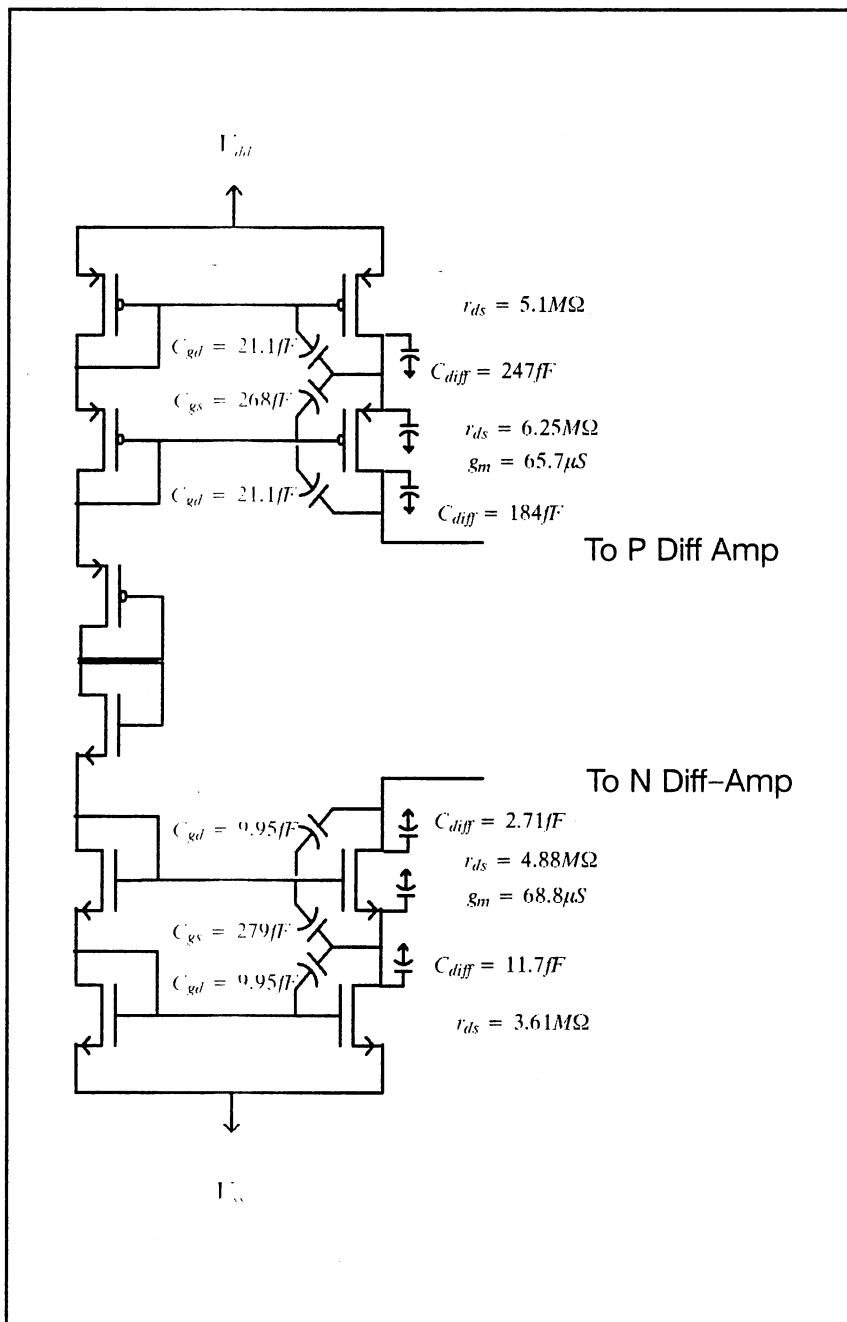


Figure 41. Spice Calculated Operating Point of Current Reference.

smaller Miller capacitance and therefore a different pole location. The correct poles for the two nodes are circled. These values become obvious as one applies logic to the four pole locations.

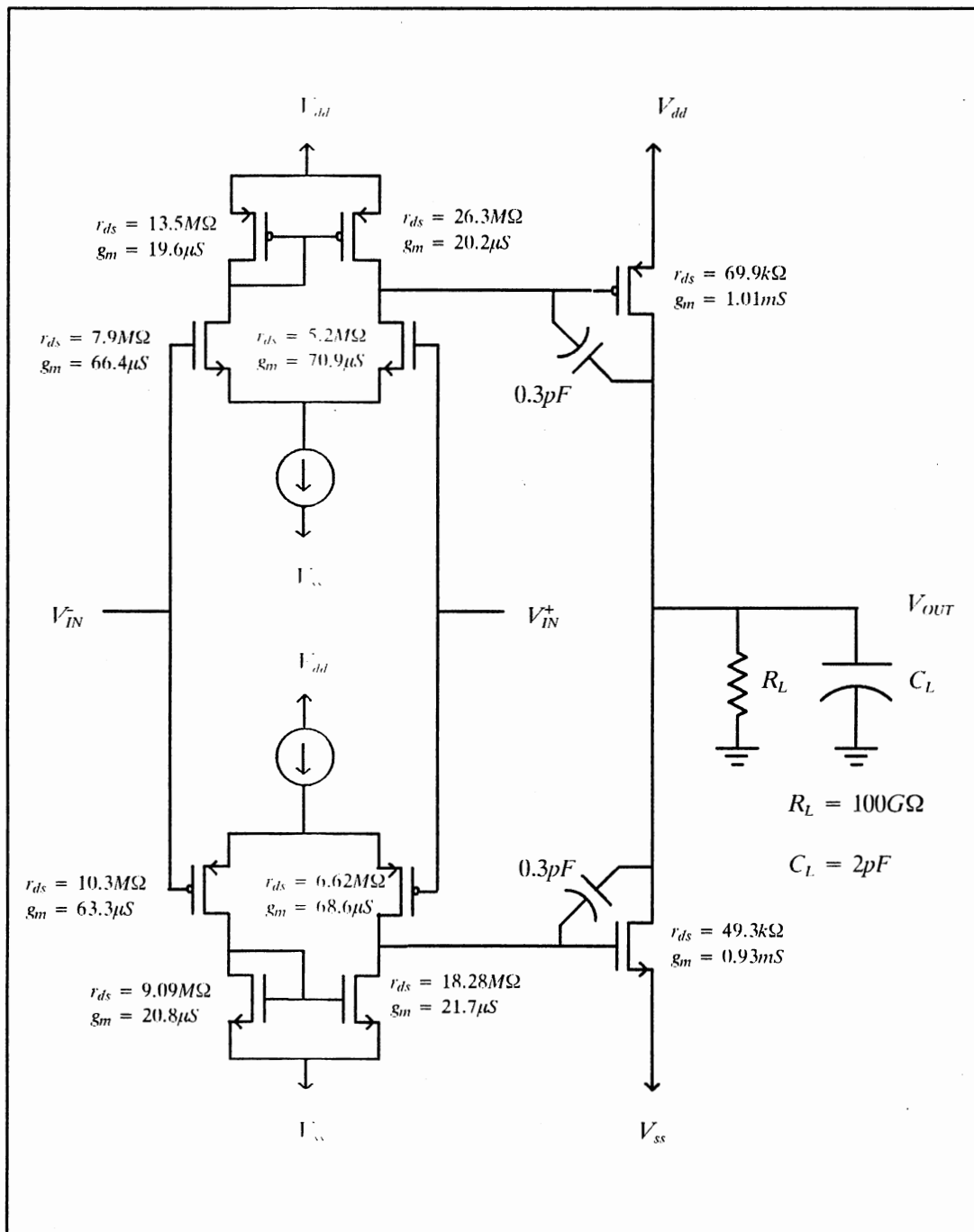


Figure 42. Spice Calculated Operating Point of Gain Stages.

Table II, below, lists the pole locations as calculated by Spice and the pole locations as calculated by hand from the Spice operating point information.

Note that the order of the poles is not guaranteed to correspond as many of the

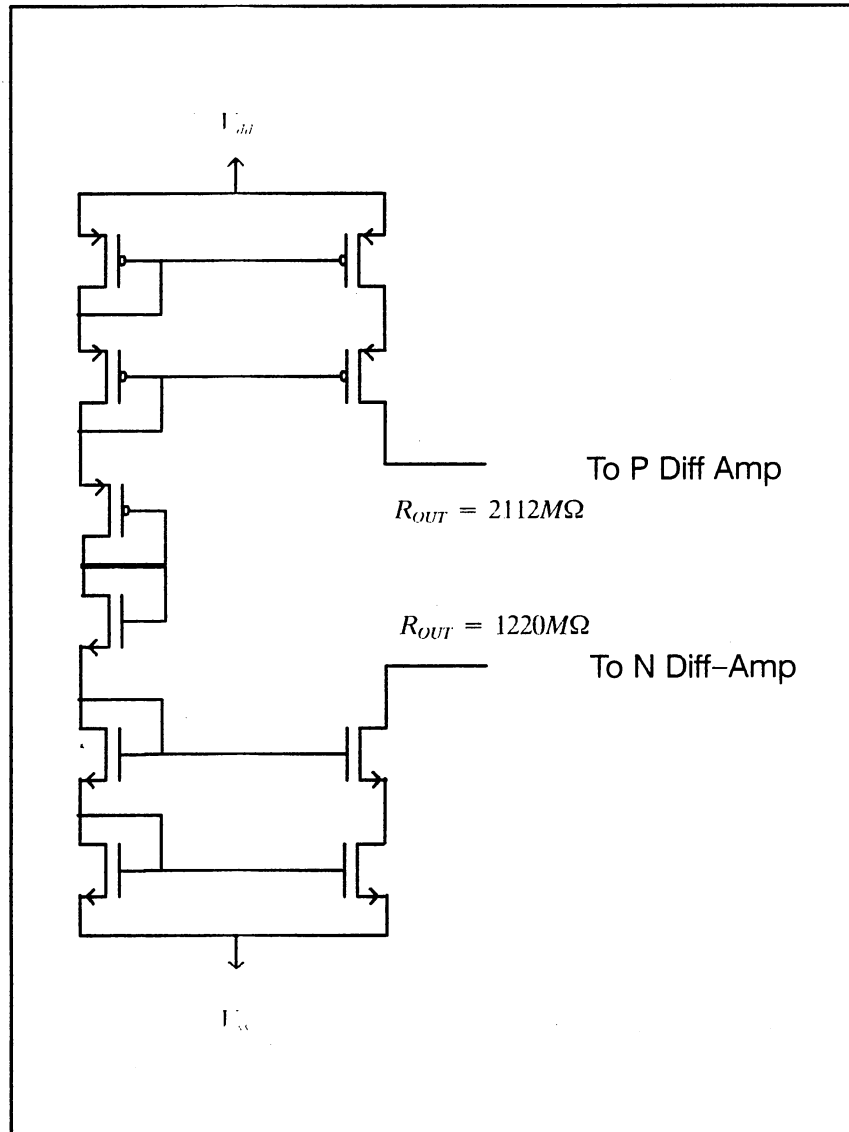


Figure 43. Equivalent Small Signal Resistances of Current References.

poles are clumped together in magnitude and small errors could reverse the order easily. The seemingly great differences between the two lists illustrates the difficulty inherent in calculating the AC response of a complicated active circuit by hand. Any time poles are close to one another, the basic assumptions of the hand analysis break down (that the gain is stable around the pole location, that the gain of one stage does not effect the gain of another stage so that one pole

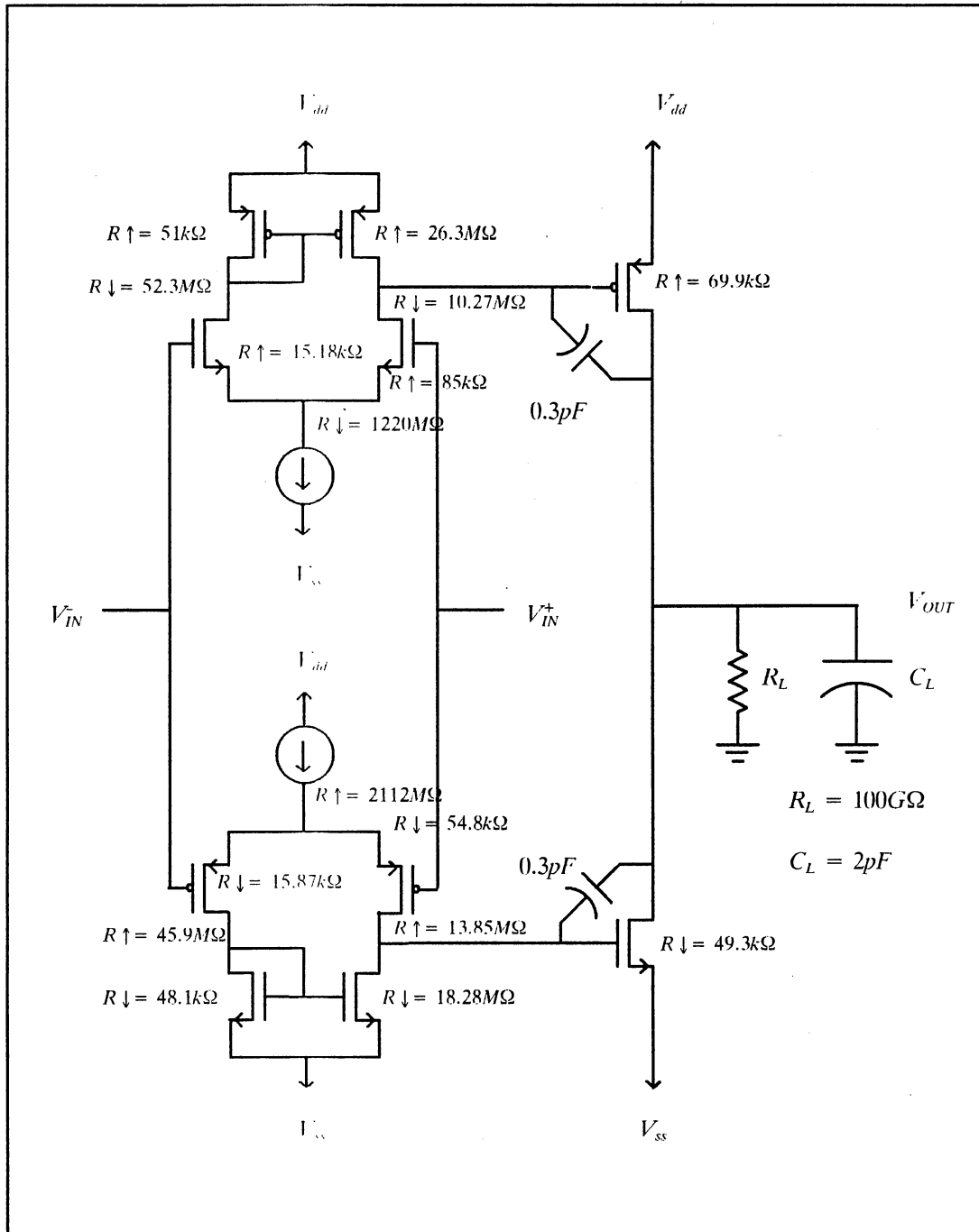


Figure 44. Equivalent Small Signal Resistances of Gain Stage.

location can not effect another pole location). In this circuit, almost every pole effects at least two other poles, making a hand calculation almost futile. A computer program could be written to perform the iterative calculation necessary, but

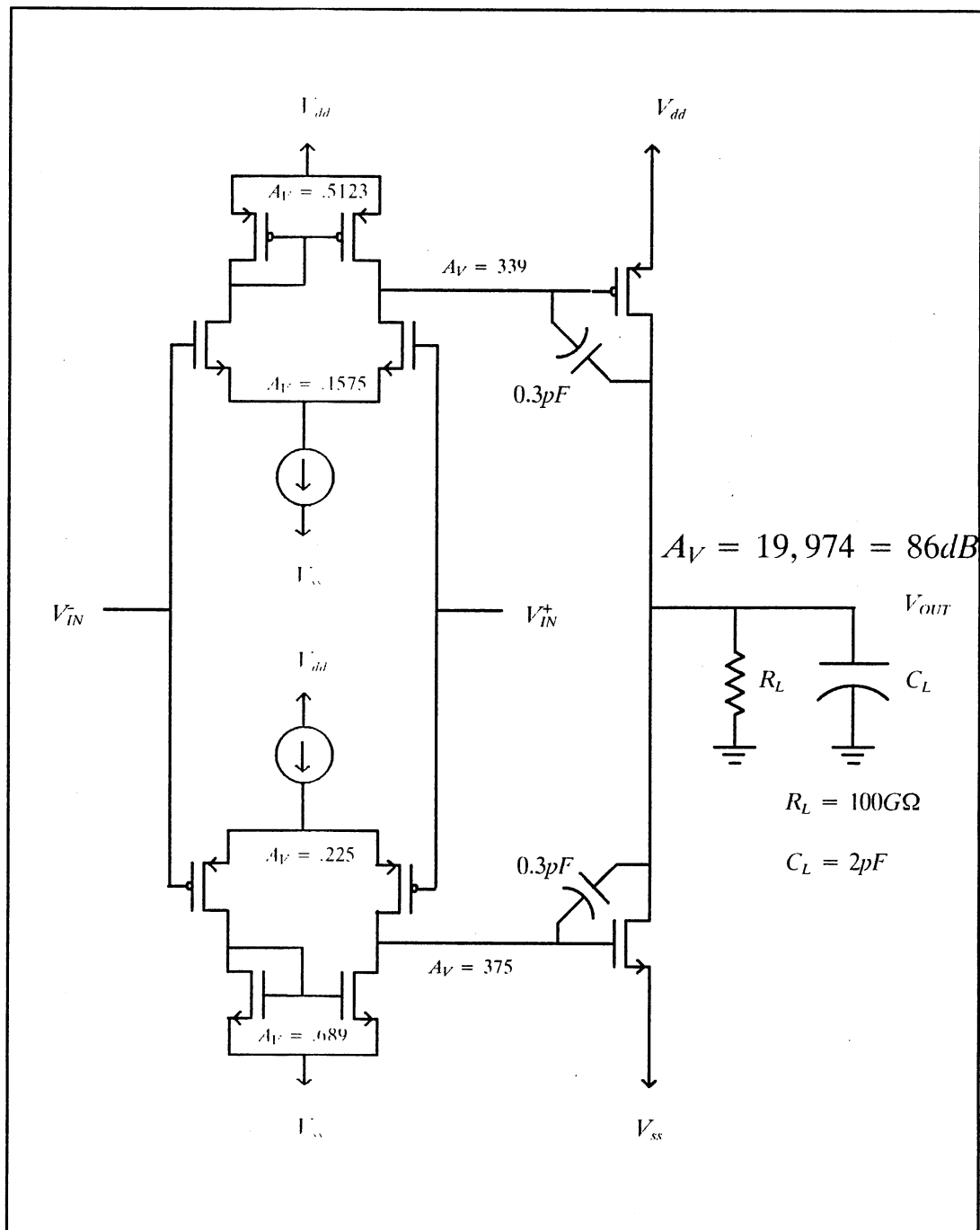


Figure 45. Manual Gain Calculation Results.

that would be redundant since we already have Spice. The advantage, however of such a program would be that one could associate a pole with a specific

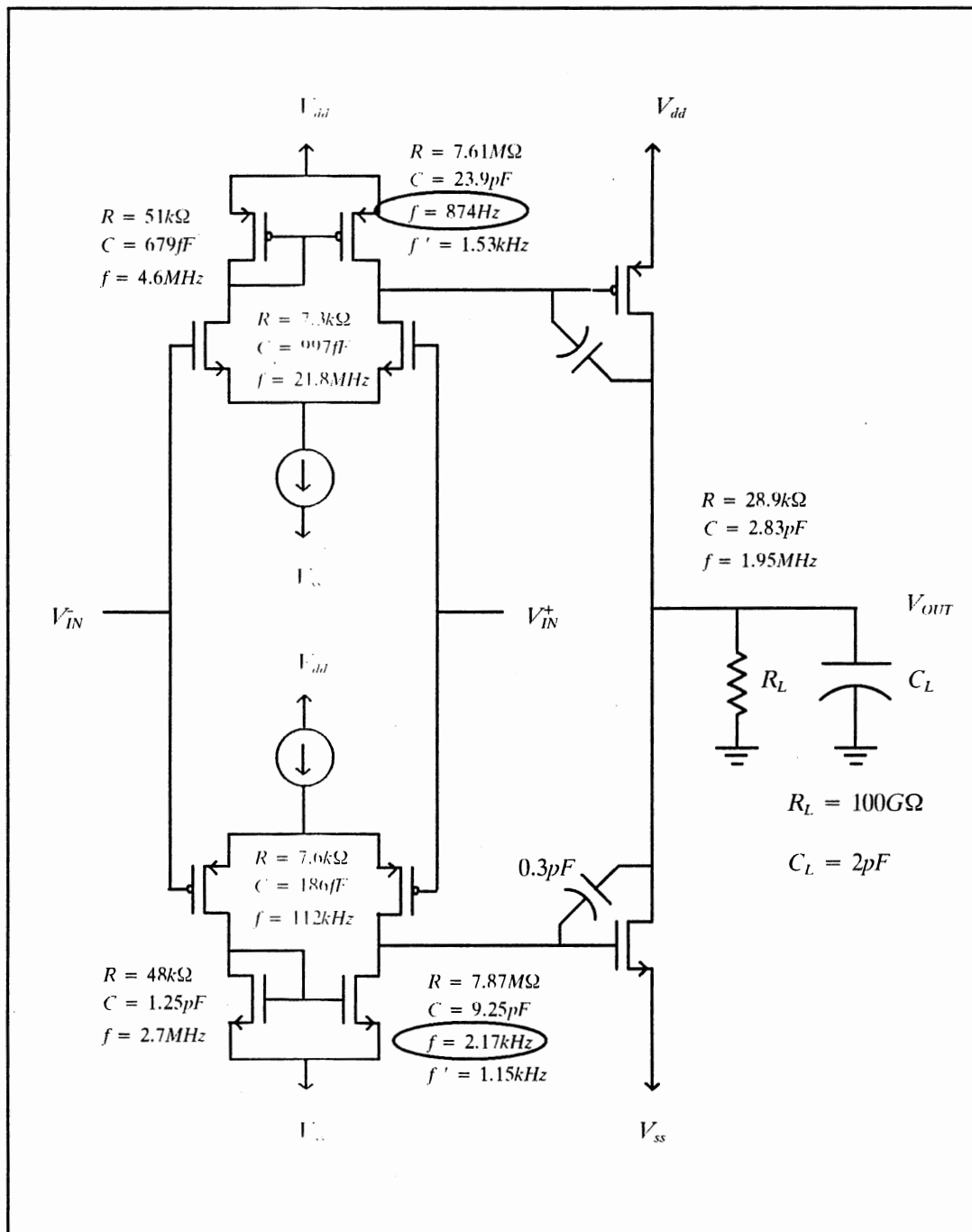


Figure 46. Manual Frequency Response Calculation Results.

node, which Spice does not do. Such information would be useful in a redesign effort.

TABLE II
POLE LOCATIONS BY SPICE AND BY HAND

Spice Locations	Hand Locations	Alternate Hand Locations
1: $-1.64405E + 03$	$-0.87400E + 03$	$-1.53000E + 3$
2: $-1.68507E + 04$	$-2.17000E + 03$	$-1.15000E + 3$
3: $-4.61966E + 06$	$-1.12000E + 05$	
4: $-1.20448E + 07$	$-1.95000E + 06$	
5: $-2.20154E + 07$	$-2.70000E + 06$	
6: $-2.63654E + 07$	$-4.60000E + 06$	
7: $-6.56648E + 08$	$-2.18300E + 07$	
8: $-3.20282E + 09$	-----	
9: $-1.02869E + 10$	-----	
10: $-1.11375E + 10$	-----	
11: $-3.91983E + 10$	-----	

It becomes obvious from examining the AC response, Figure 47 below, and the rest of the pole-zero report from Spice, Table III below, that the AC response of this circuit contains zeroes. It is not obvious how those zeroes are generated.

Zero Generation

Upon detailed analysis of the circuit, there seem to be two main generators of zeroes: source and drain degeneration, and parallel signal paths.

Any time an amplification stage has source or drain degeneration there is a rolloff frequency associated with the feedback device, so that above a certain frequency, the degeneration decreases, causing the gain to increase, which is, in effect, a zero.

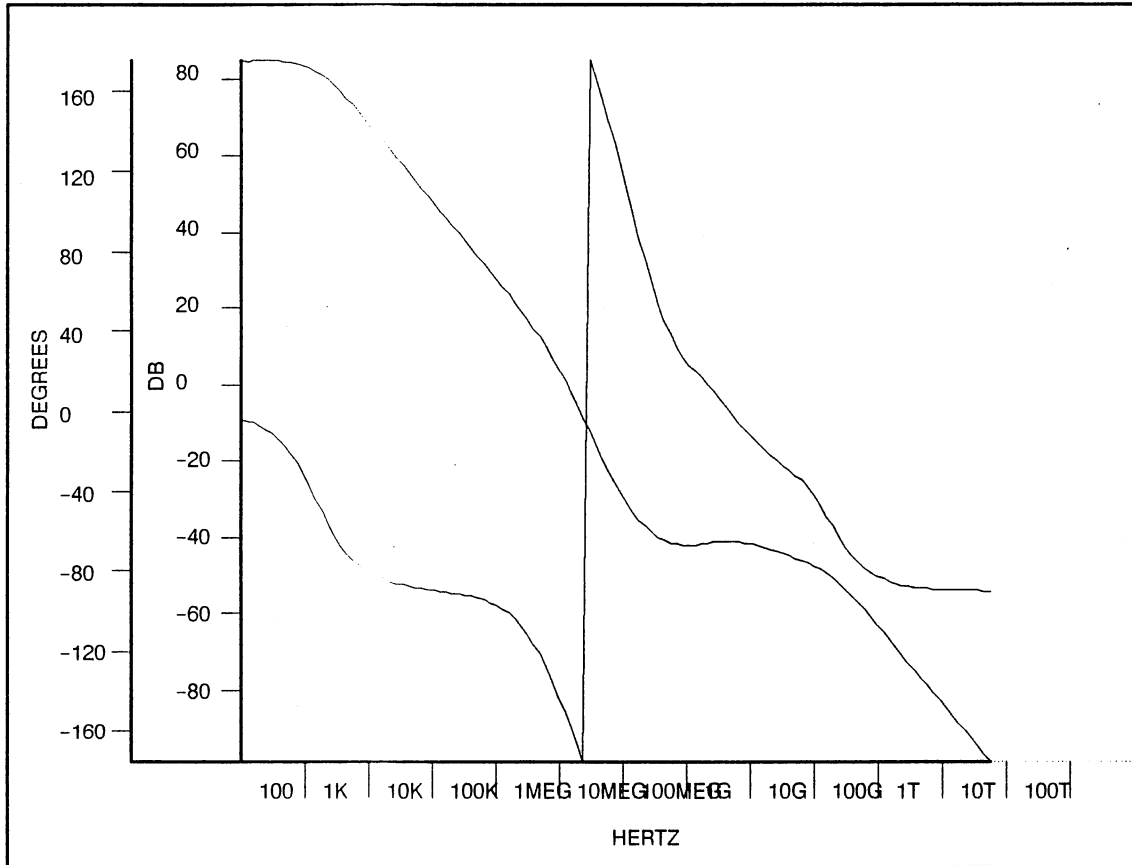


Figure 47. Spice Generated AC Response Curve.

Equation II.1 below, predicts the gain of this circuit at low frequency.

Equation II.1

$$A_V = \frac{[r_{ds} + (1 + \mu)R_s] \parallel R_d}{R_s + \frac{1}{g_m}}$$

Equation II.2 below, performs the same operation but in terms of impedance.

Equation II.2

$$A_V = \frac{[r_{ds} + (1 + \mu)Z_s] \parallel Z_d}{Z_s + \frac{1}{g_m}}$$

TABLE III
SPICE GENERATED ZERO LOCATIONS

SIGNIFICANT ZEROS				
RAD/SEC		HERTZ		
REAL	IMAG.	REAL	IMAG.	Q
7.46358E + 08	0.00000E + 00	1.18787E + 08	0.00000E + 00	5.00000e-01
2.65595E + 09	0.00000E + 00	4.22708E + 08	0.00000E + 00	5.00000e-01
-1.00652E + 05	0.00000E + 00	-1.60193E + 04	0.00000E + 00	-5.00000e-01
-1.27324E + 08	0.00000E + 00	-2.02642E + 07	0.00000E + 00	5.00000e-01
-1.56042E + 08	0.00000E + 00	-2.48349E + 07	0.00000E + 00	5.00000e-01
-6.20459E + 10	0.00000E + 00	-9.87491E + 09	0.00000E + 00	-5.00000e-01
-8.89957E + 10	0.00000E + 00	-1.41641E + 10	0.00000E + 00	-5.00000e-01
-4.20526E + 11	0.00000E + 00	-6.69288E + 10	0.00000E + 00	5.00000e-01

Equations II.3 and II.4, describe the impedances, Z_d and Z_s .

$$\text{Equation II.3} \quad Z_d = \frac{R_d}{(sC_dR_d + 1)}$$

$$\text{Equation II.4} \quad Z_s = \frac{R_s}{(sC_sR_s + 1)}$$

From examining the equations above it becomes apparent that what was originally presumed to be a single pole amplification stage, actually involves three poles, two of which are probably complex, and two zeroes. A similar effect occurs for common drain amplifiers with drain degeneration and current sources

which employ source or drain degeneration, i.e. a cascoded current source. What is more alarming is that fact that the pole and zero locations are not functions of just the source components or just the drain components, but an algebraic combination of both.

It can be shown that the gain equation as a function of frequency takes the form of equation II.5.

Equation II.5
$$A_v(s) = \frac{As^2 + Bs^1 + Cs^0}{Ds^3 + Es^2 + Fs^1 + Gs^0}$$

The other source of zero generation is parallel signal paths. Figure 48 below illustrates a simple block-diagram of two single pole system blocks placed in parallel by tying their inputs together and adding their outputs.

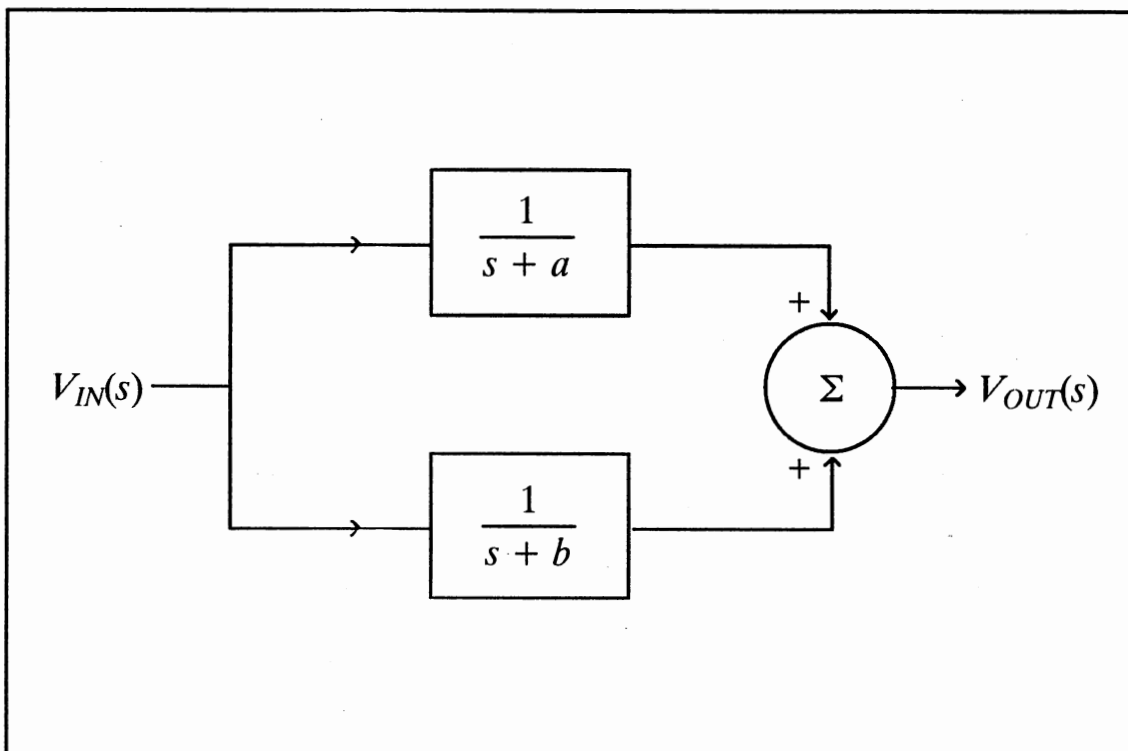


Figure 48. Single Pole Blocks Placed in Parallel.

Naturally, one would expect the resulting transfer function to contain two poles. What is not so obvious until one works the math is that a zero is generated at the average time constant. Equations II.6 and II.7 below illustrate this effect. Equation II.6 is the unreduced transfer function obtained by adding the two blocks' transfer functions. Equation II.7 shows the result of reducing this equation to a single numerator and denominator.

$$\text{Equation II.6} \quad H(s) = \frac{1}{s+a} + \frac{1}{s+b}$$

$$\text{Equation II.7} \quad H(s) = \frac{s + \frac{(a+b)}{2}}{s^2 + (a+b)s + ab} \times 2$$

As one can see by examining equation II.7, a zero has been generated at the average time constant of the two original poles. If the two blocks have something greater than 1 in the numerator, then the zero will be generated at a lower frequency. If the two blocks have something less than 1 in the numerator, the zero will be generated at a higher frequency.

The effects of putting blocks containing multiple poles in parallel can be derived by extending the behavior of the example system above. If the total number of poles placed in parallel equals P , and the number of generated zeroes equals Z , then

$$\text{Equation II.8} \quad Z = P - 1.$$

Since the simplified hand analysis resulted in 7 pole values, six of which involve source or drain degeneration, one could estimate that 19 poles and 12 zeroes will be generated before the parallel block effects described above are considered. Each diff amp represents three poles being placed in parallel and the output stage represents six poles being placed in parallel. This implies that 9 zeroes will be added from parallel block effects. This overabundance of zeroes (21 zeroes versus 19 poles) implies a non-causal system. This is of course ridic-

ulous since the system in question is a circuit constructed of individual transistors which are certainly causal devices. The simple model described here has neglected poles and zeroes internal to the transistors and those internal to the current references. Spice pole-zero analysis finds 23 poles and zeroes which cancel each other out and implies that more exist at higher frequencies but has to terminate the analysis because the numbers exceed capability of the machine

The purpose of this pole-zero discussion has been to establish the difficulty of predicting the AC response of the circuit in anything but a rough sense as a result of the introduction of zeroes into the system and to describe how those zeroes are generated.

Conclusion

This chapter has discussed the proposed analog CMOS comparator design and its AC and DC characteristics along with an analysis of the differences between hand calculations and computer simulations. The Spice generated output files and graphs are appended to this thesis for completeness and the convenience of the reader as a reference. The following chapter will discuss efforts to model this circuit in order to evaluate its performance in the context of the proposed neural median filter.

CHAPTER III

MODELLING THE COMPARATOR DESIGN

Introduction

This Chapter describes efforts made toward accurately modelling the comparator described in Chapter II. It first presents a justification of the effort, followed by a detailed description of three approaches to the modelling problem and their strengths and weaknesses. The chapter concludes by briefly discussing the final model and the reasons for choosing it above the other two.

Justifying the Modelling Effort

As any one who has done Spice analysis knows, the transient analysis of MOS devices is one of the most time consuming analyses possible. The algorithm that Spice employs for the transient analysis uses cpu time in a manner that is exponential in circuit complexity. The proposed neural median filter being examined in this thesis is a circuit that depends on transient behavior, therefore one could expect any proposed implementation of it to undergo heavy transient simulation.

Toward that end, transient analyses of the proposed circuit design were performed to estimate slew rate and therefore the speed with which the proposed system converges to an estimate of the median. A single run on an HP Apollo 400 series workstation, with TISpice3 running under Domain/OS 10.3.5, took over five hours to finish with no other jobs running on the node.

When faced with the number of simulations that would be required to fully characterize the proposed median filter system, along with the fact that those simulations will include multiple copies of the proposed circuit design all contending with one another to drive the output voltage, it becomes obvious that Spice's exponential-in-circuit-complexity algorithm is insufficient for the job on the available hardware.

The Proposed Models

Three methods were considered as possibilities: A purely numerical, system level simulation, a very precise macro model Spice simulation, and a simplified macro model Spice simulation.

The numerical simulation was based on Runge-Kutta numerical analysis equations. It was designed to model an all-pole, three-pole system with some offset voltage and output resistance. The output current rails were symmetrical except for the effects of the output resistance. It was written in the C programming language on an IBM-286 compatible machine and on the HP series 400 workstation.

The precise macro model was based Spice subcircuit definitions. It was intended to model the DC and AC characteristics of the proposed design exactly.

The simplified macro model was also based on Spice subcircuit definitions. It was intended to model the DC characteristics of the proposed design exactly. The AC characteristics were approximated so that the resulting model had a similar roll-off frequency and a similar stability characteristic with 180 degree phase crossover frequencies in the right order of magnitude.

A detailed discussion of each of these proposed circuit models follows.

The System Level Simulation

The Runge–Kutta Numerical Method

The Runge–Kutta numerical method for solving differential equations is a standard means of simulating complicated dynamic systems. For the purposes of this investigation, a second order Runge–Kutta method was chosen. The order of a Runge–Kutta solution is the number of derivative approximations that are used. For a second–order Runge–Kutta solution, two derivative approximations are made and their average is used to find the final estimate of the next point on the solution.

If the value of state variable q is known at time $k-1$ and the value of the driving input signal is known at time $k-1$, then the Runge–Kutta method follow directly from the state–variable equation shown in equation III.1 below.

$$\text{Equation III.1} \quad \dot{q}(k-1) = f [q(k-1), u(k-1), k-1] .$$

By applying Euler's method of numerical solution to this differential equation, an estimate of $q(k)$ is formed according to equation III.2 below

$$\text{Equation III.2} \quad q(k)_\epsilon = q(k-1) + T \times f [q(k-1), u(k-1), k-1] .$$

This estimate is then plugged back into the original differential equation , along with the known value of $u(k)$, to achieve an estimate of the derivative of q at time k .

$$\text{Equation III.3} \quad \dot{q}(k)_\epsilon = f [q_\epsilon(k), u(k), k] .$$

Finally, the increment to $q(k-1)$ to reach $q(k)$ is calculated using the average of the two derivatives .

$$\text{Equation III.4} \quad q(k) = q(k-1) + \frac{T}{2} \times [\dot{q}(k-1) + \dot{q}_\epsilon(k)] .$$

This numerical solution method produces an approximation of the true solution of the differential equation within an error that has order $3T$, so that halving the timestep size reduces the error by one eighth. This is an advantage over Euler's method, which has error of order $2T$. Therefore, a larger step size can be used by this method to achieve a comparable accuracy to Euler's method using a smaller stepsize, however, two derivative approximations are needed for the Runge-Kutta rather than one.

By applying this numerical solution method to a system of differential equations in state-variable form, one can easily simulate any dynamic system with a reasonable amount of accuracy. Equations III.5(a-d) below illustrate the state-variable form of a three-pole system.

$$\text{Equation III.5 (a)} \quad \dot{q}_1(t) = q_2(t) .$$

$$\text{Equation III.5 (b)} \quad \dot{q}_2(t) = q_3(t) .$$

$$\text{Equation III.5 (c)} \quad \dot{q}_3(t) = - (p_3q_3(t) + p_2q_2(t) + p_1q_1(t)) + p_1u(t) .$$

$$\text{Equation III.5 (d)} \quad y(t) = q_1(t) .$$

This system of equations could be used to represent the AC or dynamic characteristics of the proposed comparator design. By making the output variable $y(t)$ a non-linear function of $q_1(t)$ and adding two equations to represent the integration of the output variable and the losses through the output resistance of the comparator, the complete response of the comparator could be approximated. Equations III.6 (a) and (b) illustrate the necessary additions to the set of state-variable equations to fully model the comparator. Figure 49, below, illustrates the non-linear transfer function, $f(q_1(t))$.

$$\text{Equation III.6 (a)} \quad \dot{y}(t) = \frac{f(q_1(t))}{C} - \frac{y(t)}{RC} .$$

$$\text{Equation III.6 (b)} \quad u(t) = u_{in}(t) - y(t) .$$

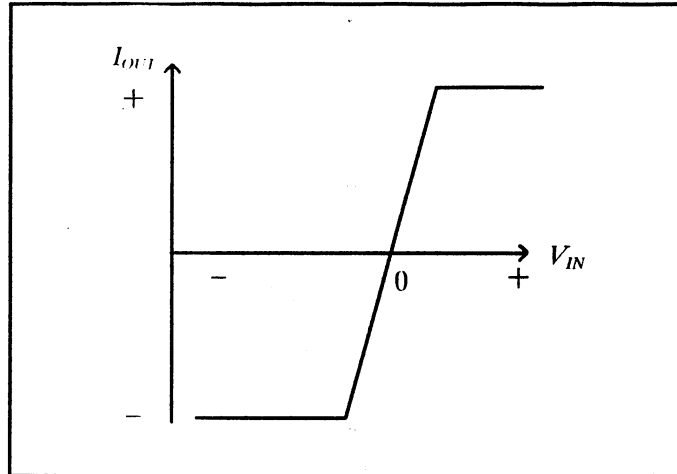


Figure 49. The Non-Linear Transfer Function Approximation for the Runge-Kutta System Level Simulation.

Application to the Neural Median Filter System

The resulting Runge-Kutta simulation algorithm for the proposed comparator and neural median filter system can then be represented by the flow chart shown below in Figure 51. Figure 50, below, represents the neural median filter system for reference when examining the simulation algorithm. The resulting Turbo-C program is appended to this thesis.

The resulting program successfully simulates the expected behavior of the proposed system. However, it is very slow and does not take the true non-linear behavior of CMOS transistors into account. Also, the stability of the simulation is directly dependant on the time-step size, as is the simulation error. The program's usefulness only evidences itself in a proof-of-concept context. Semi-realistic values can be entered into the simulation to evaluate whether the proposed median filter system is at all feasible. Any real simulation of the system's

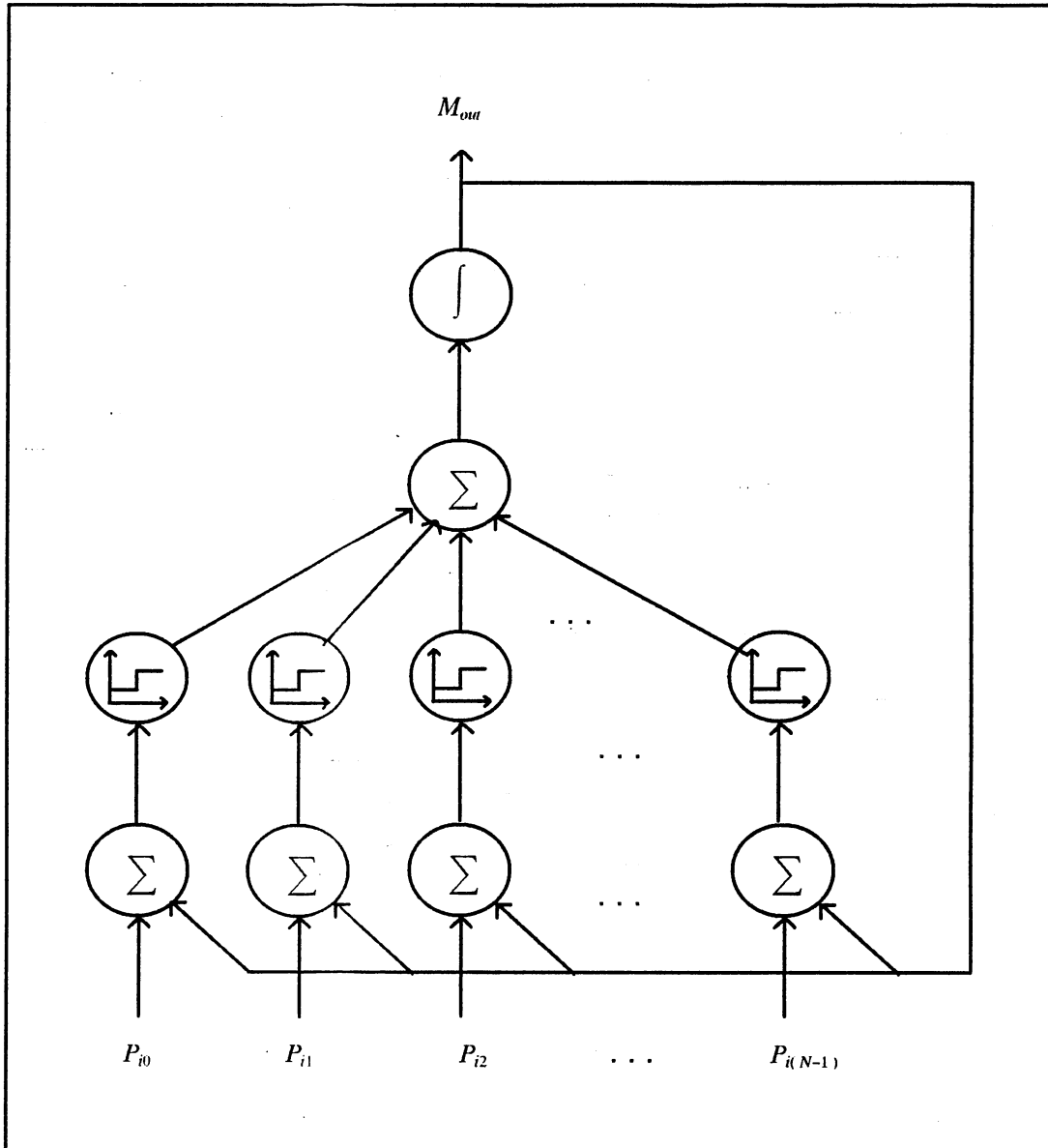


Figure 50. Neural Median Filter Architecture

behavior needs a more complex and powerful tool than this brute-force, numerical solution.

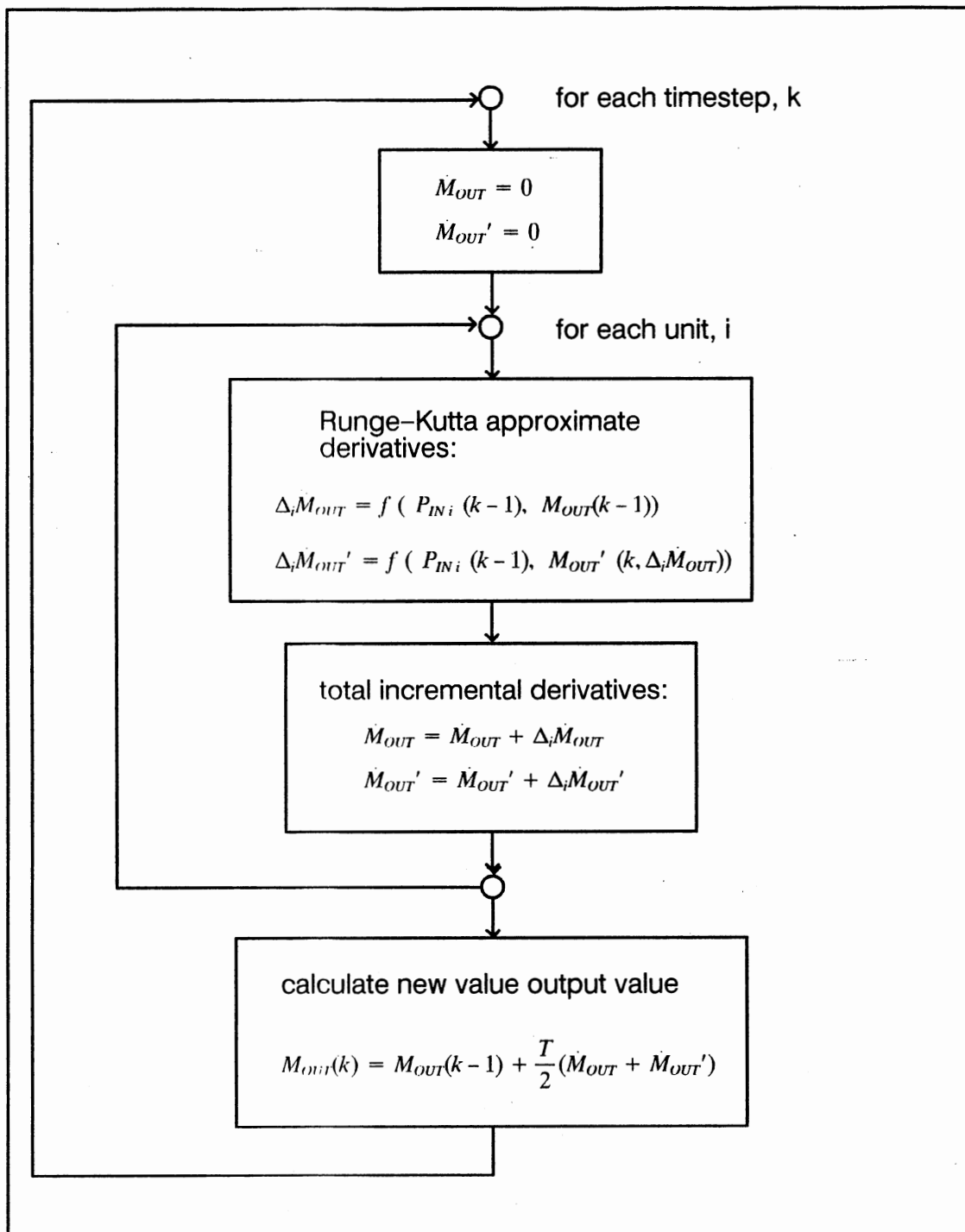


Figure 51. The Runge-Kutta Simulation Algorithm for the Proposed System.

The Spice Macro Model Simulation

The Macro Model Components

To model the AC and DC characteristics of the proposed comparator design, a few Spice subcircuits had to be developed. These were: an accurate DC output current characteristic model, a pole model, and a zero model.

The DC Output Characteristic Model. After a number of attempts at modeling the DC characteristics of the circuit using a simplified transistor based model, a piece-wise linear voltage controlled current source was decided upon. The model data for the current source was simply cut from the output of Spice DC analyses and pasted into the correct format for a Spice piece-wise linear model. The resulting model very closely approximates the DC output characteristics of the proposed model. Figure 52 below illustrates the DC model.

Figure 53a, below, compares the DC transfer function of the proposed design with the DC transfer function of the proposed model. Note the slight differences. These are a result of the 'resolution' of the model and the linear interpolation that Spice performs on the model data. The accuracy could be improved by increasing the resolution, that is decreasing the voltage step size when generating the model data. Figure 53b, verifies that the model is accurate for a different load conditions.

The Left Half Plane Pole Model. Designing the pole model was very straight forward. All that was required was that the subcircuit have a zero response at DC and roll-off at 20 dB/decade at a certain frequency corresponding to the pole location. A simple voltage dependant voltage source driving a series RC unit was chosen. The product of R and C determines the roll-off frequency so the resistor was parameterized to be proportional to the reciprocal of the pole

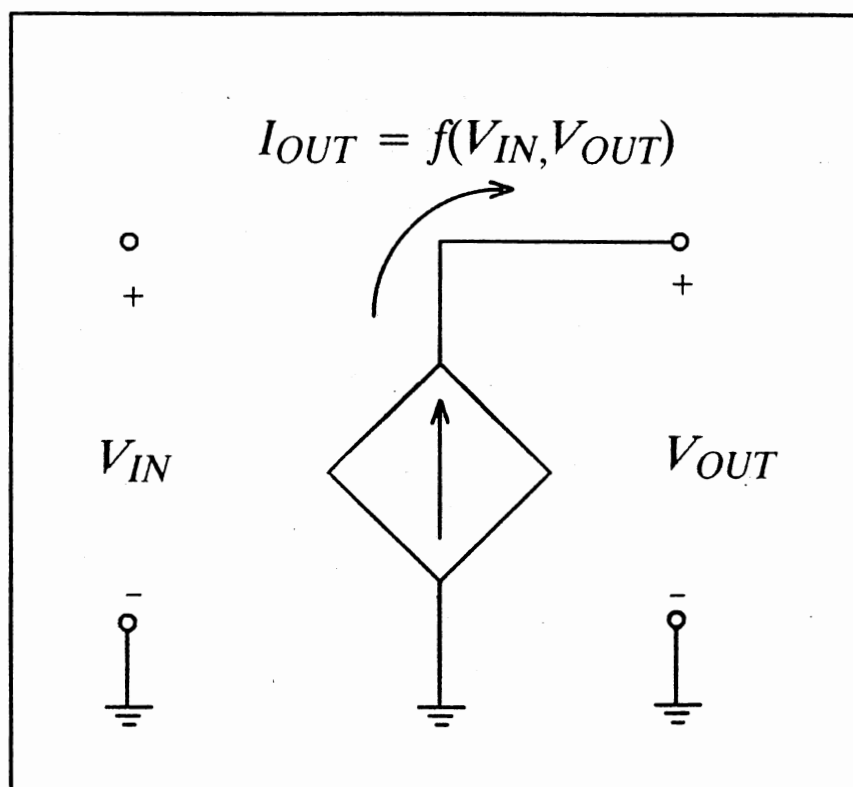


Figure 52. The DC Output Characteristic Model

location in radians/second. Figure 54 illustrates the subcircuit while Figure 55 depicts the resulting AC response.

The Left and Right Half Plane Zero Models. The zero model was not as straight forward. What is needed is a subcircuit that will start at zero DC and 'roll-up' at 20 db/dec at a certain frequency. Several ideas were tried but none proved to work until an attempt was made to derive the model from the differential equation that defines a zero.

Recall that a zero is a system block that has the transfer function shown in Equations III.7 and III.8 below.

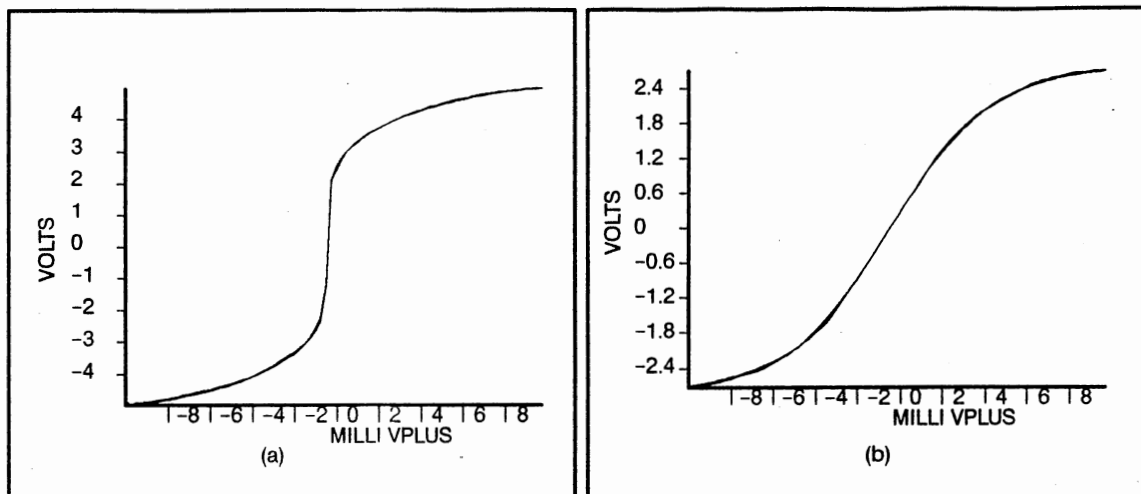


Figure 53. Comparison of the DC Transfer Function of the Design and the Model Driving (a) a Nominal Load, and (b) a Larger (Less Resistive) Load.

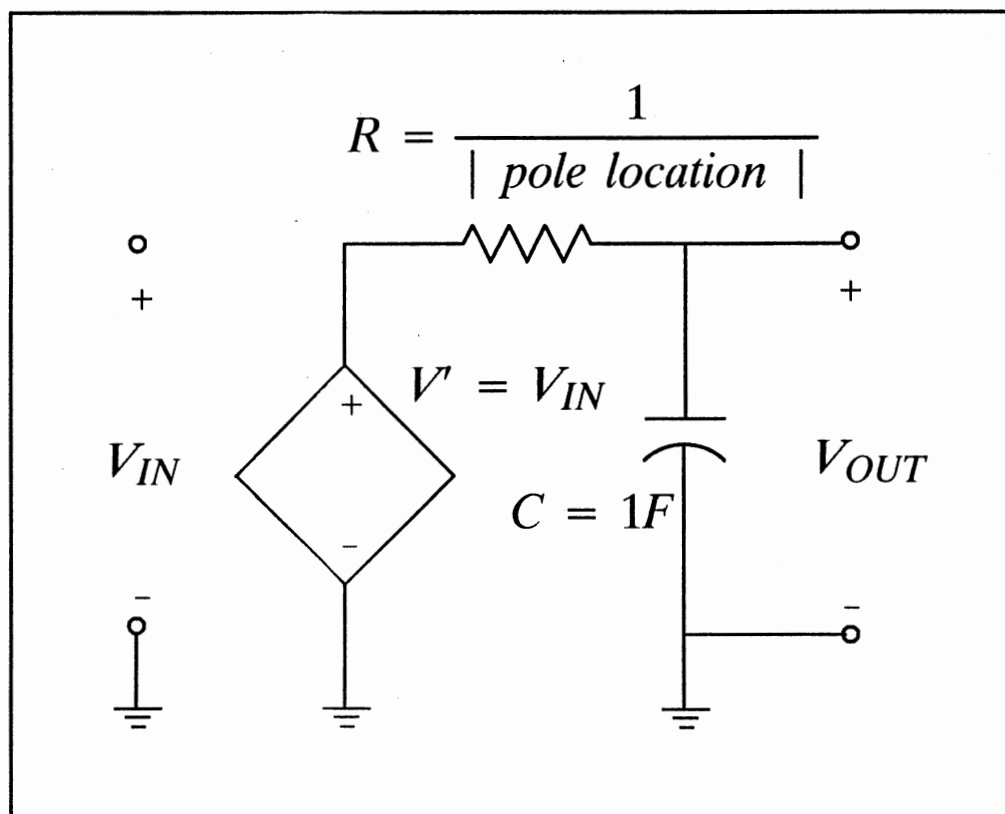


Figure 54. The Left-Half-Plane Pole Model

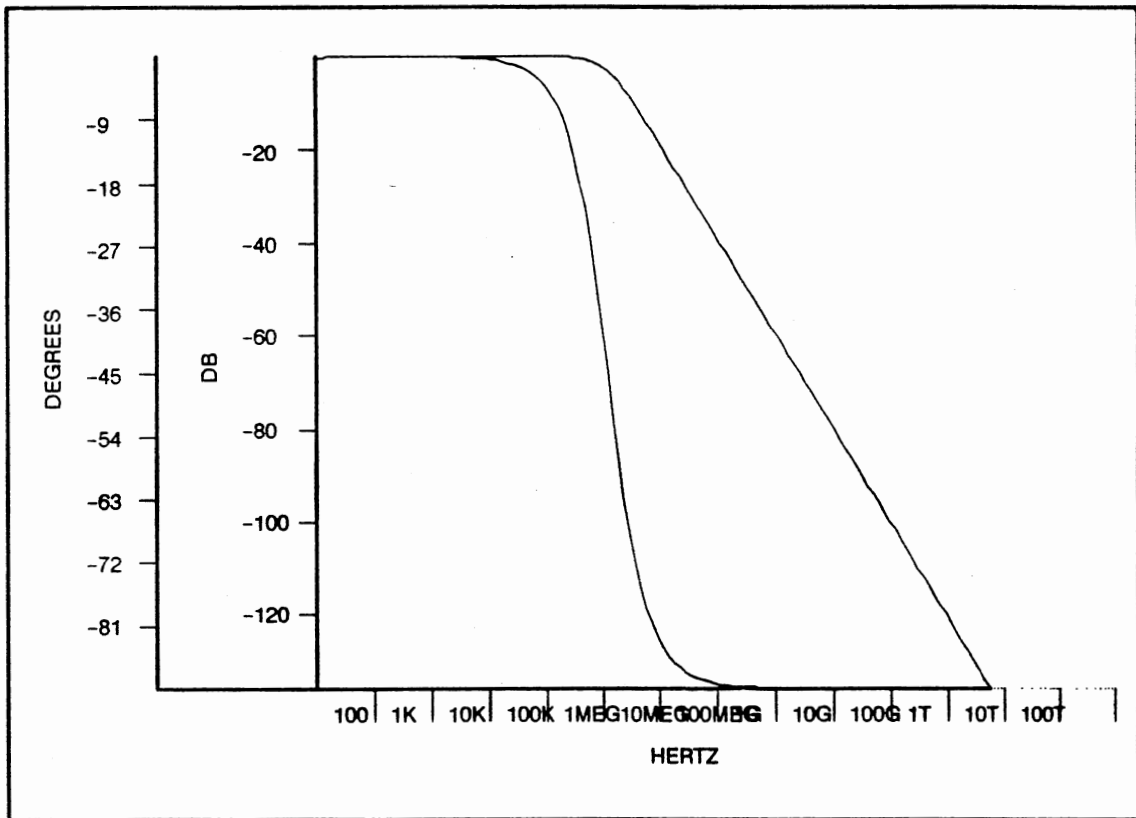


Figure 55. AC Response Curve of the Pole Model

Equation III.7
$$H(s) = \frac{s + a}{a}$$

Equation III.8
$$V_{out}(t) = \frac{1}{a} \frac{dV_{in}(t)}{dt} + V_{in}(t)$$

if one assumes zero initial conditions. So a circuit is needed that will reproduce this differential relationship. After some thought it will become apparent that the circuit in Figure 56 will produce just such a relationship between its input and output voltages.

A peculiarity of how Spice handles the sign of voltages requires that two subcircuit models be built, one for zeroes in the positive half of the s -plane, the other for poles in the negative half of the s -plane. They both produce the AC response curve shown below with the positive going phase curve being produced

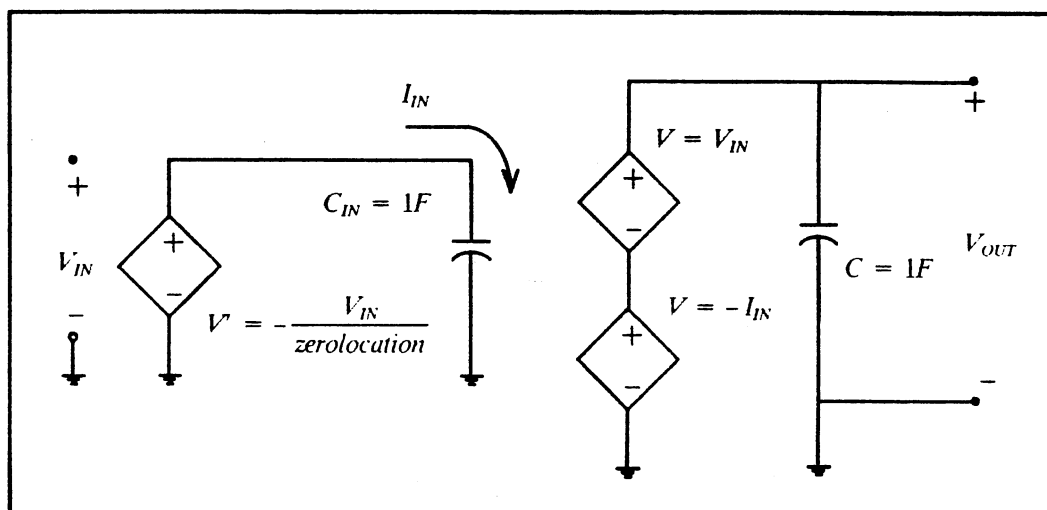


Figure 56. Schematic of the Zero Model

by the negative zero and the negative going phase curve being produced by the positive zero.

Synthesizing an Example AC Response Curve. These three models, the pole, the negative zero, and the positive zero can be combined to produce any desired AC response curve. For instance, say there is a zero at $-2\pi e + 05$ and two poles, one at $-2\pi e + 03$ and one at $-2\pi e + 07$. This example system is shown in Figure 58 below.

One would expect the frequency response curves to follow asymptotes as shown below in Figure 59. When the appropriate numbers are plugged into the macro models described above, the AC response characteristic in Figure 60 results. Note the excellent correspondence of the curve with the asymptotes in Figure 59.

With the AC and DC components described above, the AC and DC behavior of practically any circuit can be modelled. The next section will describe the attempts to use these components to precisely model the proposed comparator design.

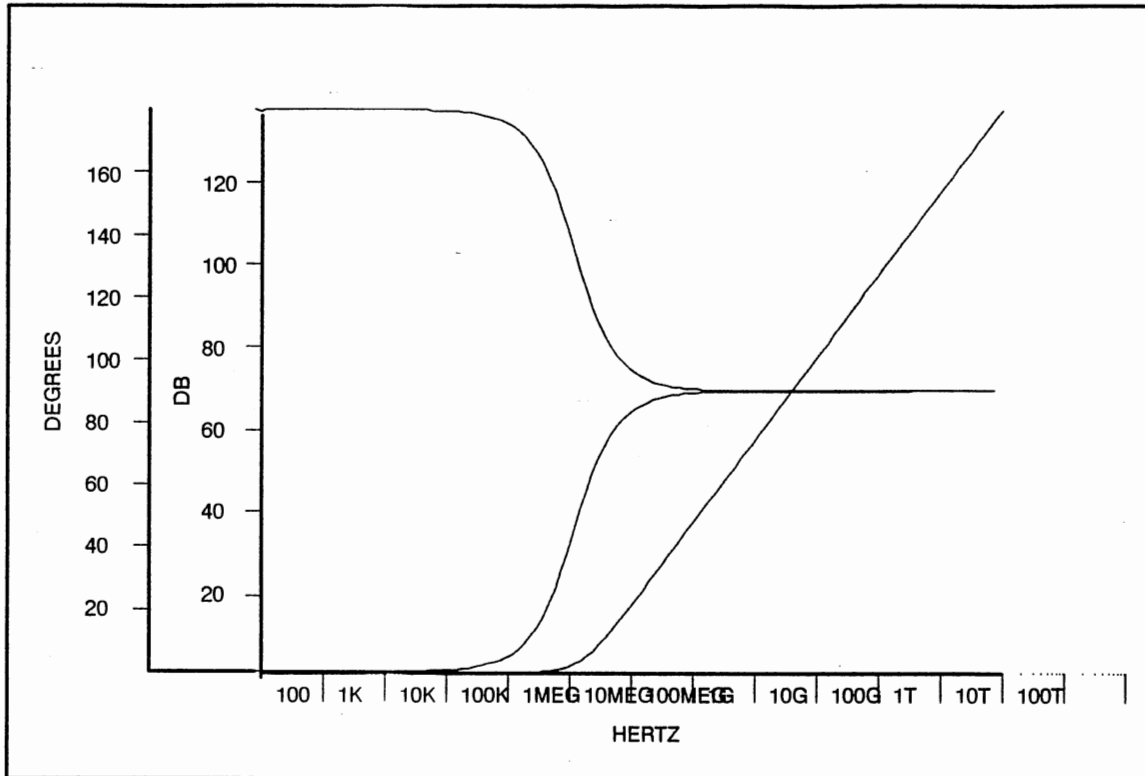


Figure 57. AC Response Curves of the Zero Models

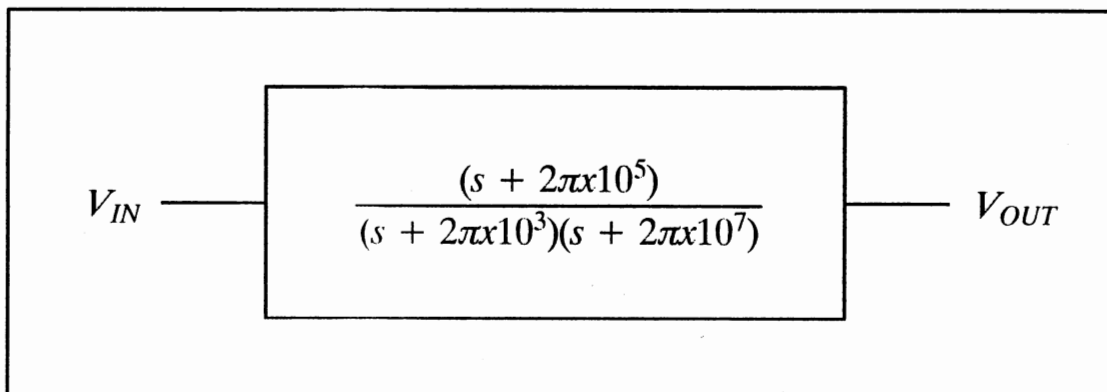


Figure 58. Example Pole-Zero System

The 'Precise' Macro Model

An attempt to exactly the model the AC and DC characteristics of the proposed comparator design was made. After using Spice to analyze the difference

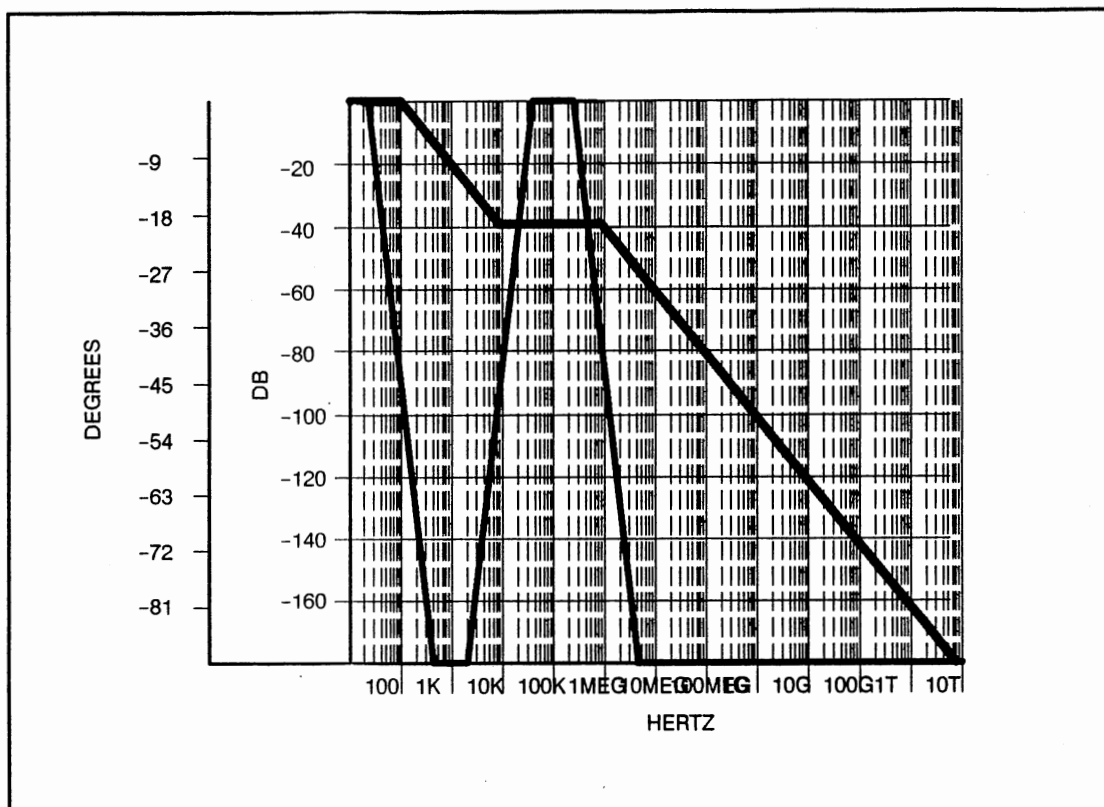


Figure 59. Asymptotic AC Response of the Example System

mode AC characteristics of the circuit and produce a list of poles and zeroes, these pole and zero values were used as the parameters of the pole and zero models described in the previous chapter. By concatenating a chain of zeroes and poles, the AC response of the circuit was constructed. By driving the DC output current model with the output of the series pole and zero macros, the DC response of the circuit was constructed. Adding capacitances to the output and input nodes completed the model shown in Figure 61 below.

Several experimental runs were made to see how effective this model would be. As expected, the DC and AC responses were modelled exactly and with much quicker convergence time. However, the transient analysis was almost as slow as the full blown circuit simulation. Also, for all the effort given to exactly model the AC response, it is only accurately modelled for a given set of

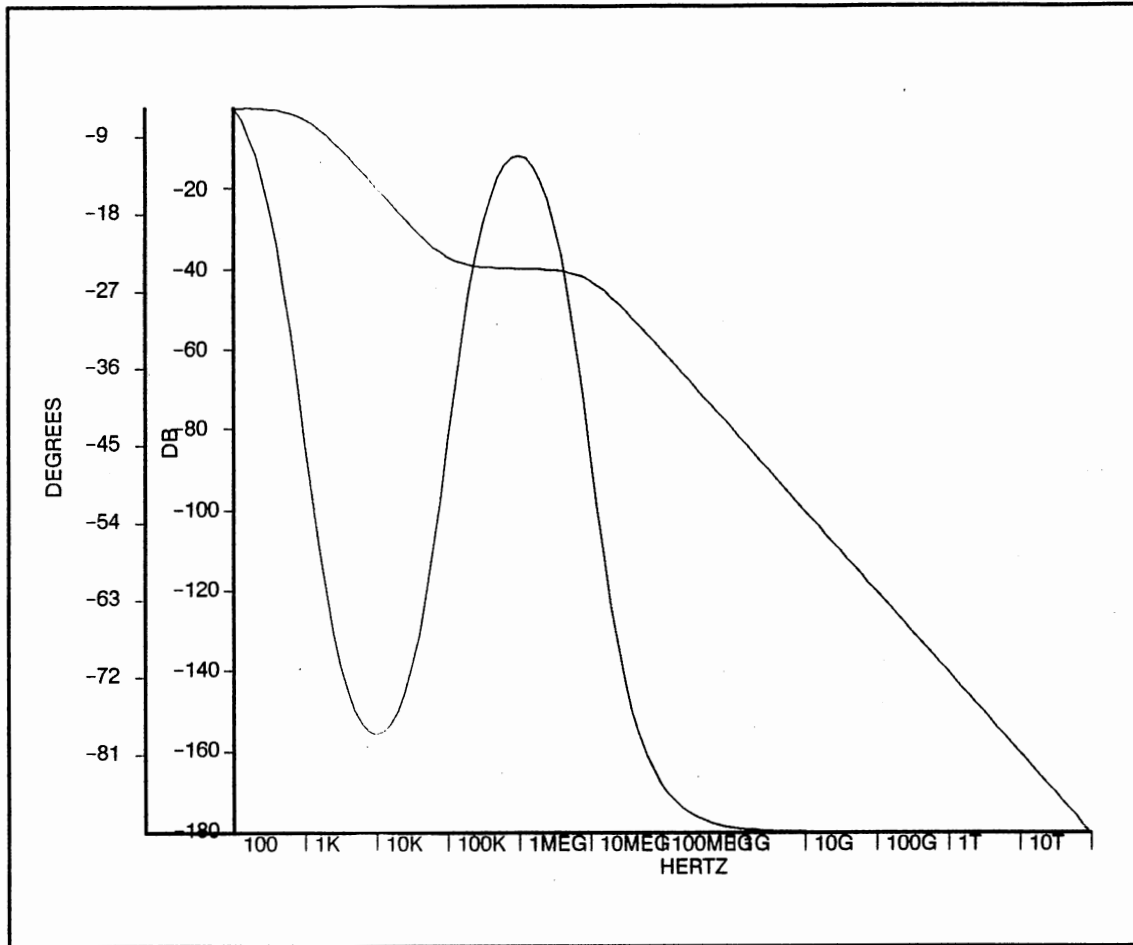


Figure 60. AC Response of the Example System

load conditions because Miller compensation was used in the design to insure stability. Therefore the exterior pole heavily influences interior pole locations. If the load capacitance is changed, the AC response is changed and a new model would have to be generated. Figure 62 illustrates once again the AC response of the circuit and the AC response of the macro model. They are identical. Figure 63 performs the same comparison under slightly different capacitive load conditions. The two curves diverge by a large amount.

When an attempt to do a full system simulation using this macro model was made, the simulation failed to converge, graphically illustrating the faults of the model, mainly that it is still too complex. As shown above, even if the simula-

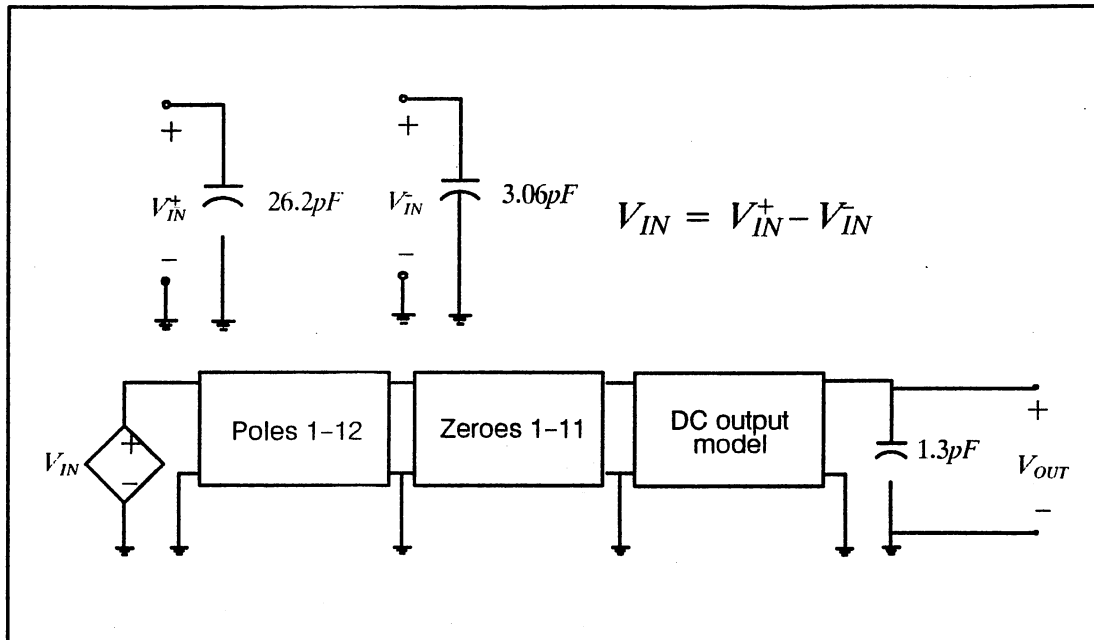


Figure 61. Schematic of the Precise Macro Model

tion did eventually converge, the results would not be accurate enough to warrant the effort since the model is not sensitive to differing load conditions.

An attempt to drastically simplify the model is described in the next section.

The Simplified Macro Model

The simple macro model was an attempt to model the most important things about the AC response, that is roll-off frequency and stability characteristics, and keep the fairly exact piece-wise linear DC model. To accomplish this end, the same pole and zero subcircuits were used as in the 'exact' model described above. For this simplified model, however, only the most dominant poles and zeroes were kept in the model. Figure 64, below, diagrams the simplified model.

As one would expect, the DC characteristics still match precisely, but the AC characteristics show marked differences. Figure 65, below, compares the

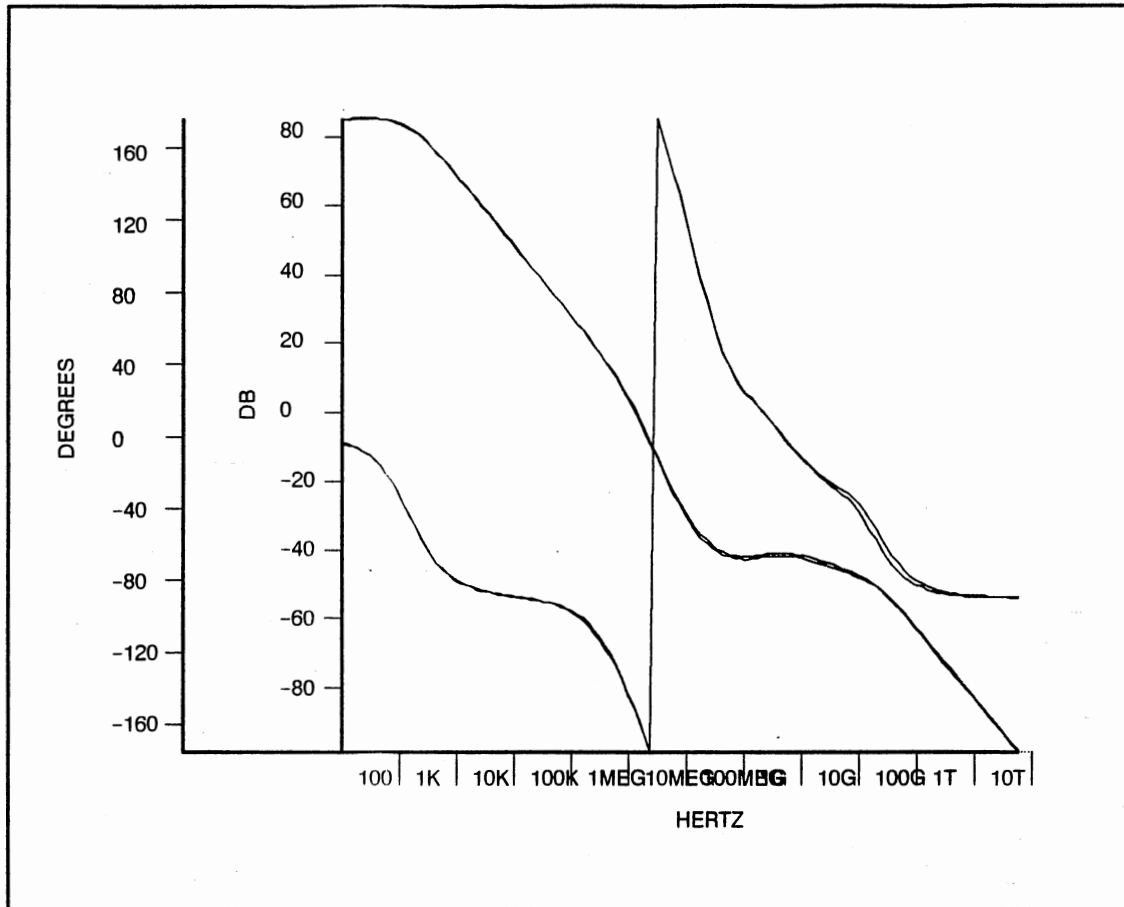


Figure 62. Comparison of the Circuit and Macro Model's AC Response

AC response curve of the proposed comparator design to the AC response curve of the simplified model.

The main disadvantage of this model is, of course, that the AC response is not exactly modelled. However the stability characteristics and roll-off frequency, arguably the most important features of the AC response, are modelled fairly closely. Figure 66, below, illustrates the similarity of the stability characteristics by comparing the AC responses of the two circuits under conditions of unity negative feedback (unity gain).

But the most attractive feature of this model is that it readily converges under transient analysis allowing the crucial system-level simulation needed to

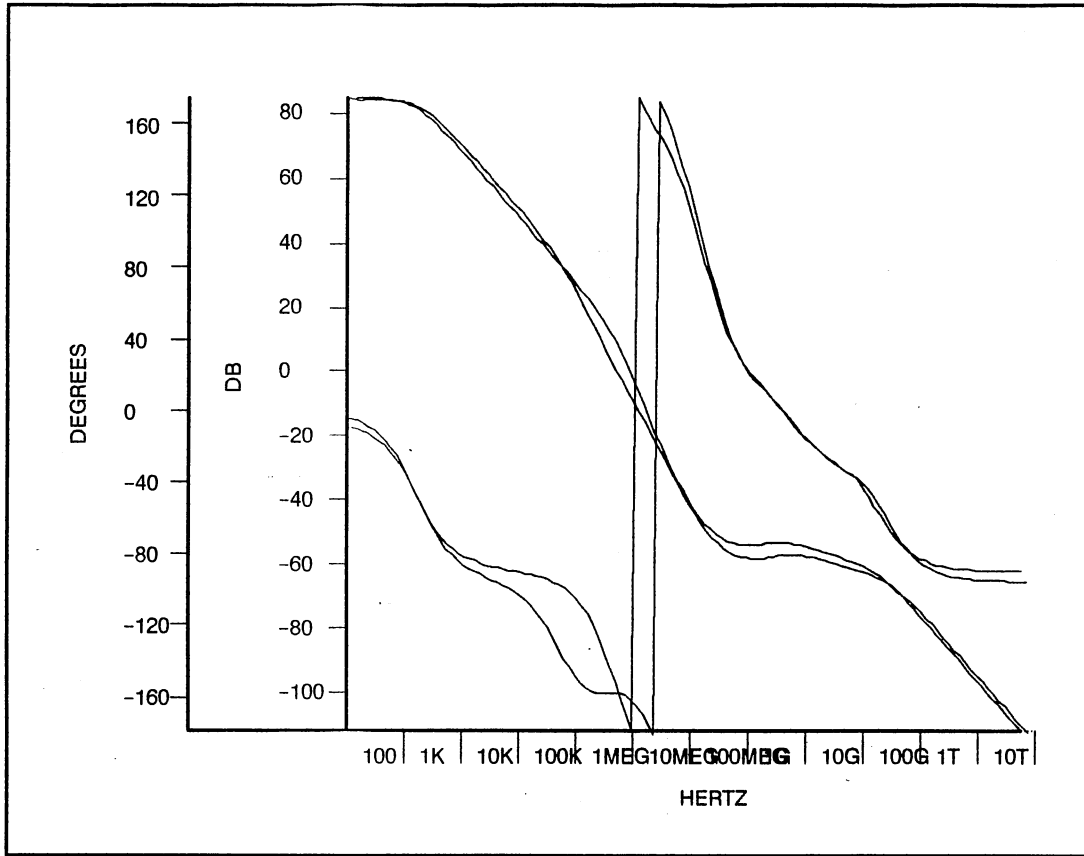


Figure 63. Comparison of the AC Responses Under Other Load Conditions

evaluate the performance of the designed circuit within the context of the proposed neural median filter system. Figure 67, below, illustrates the results of one such analysis.

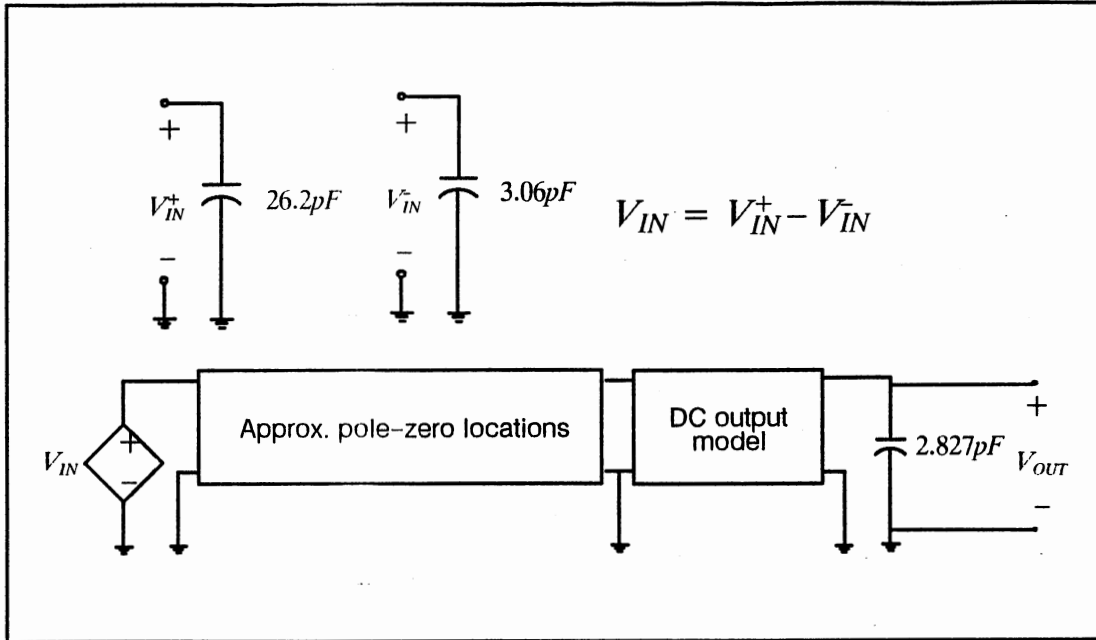


Figure 64. Schematic of the Simplified Macro Model

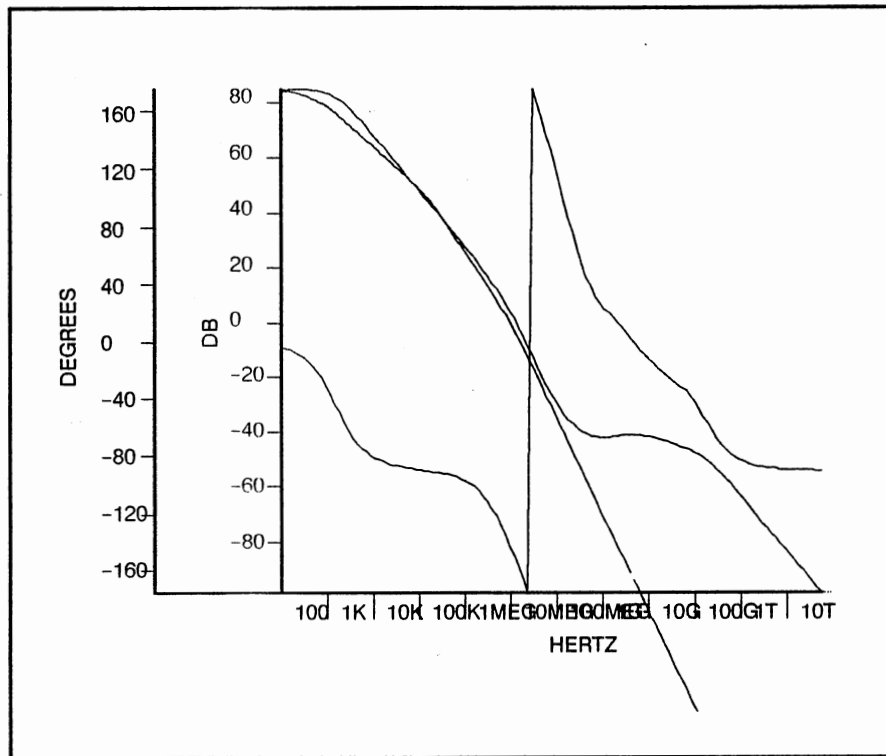


Figure 65. Comparison of the Circuit and Simplified Model's AC Responses

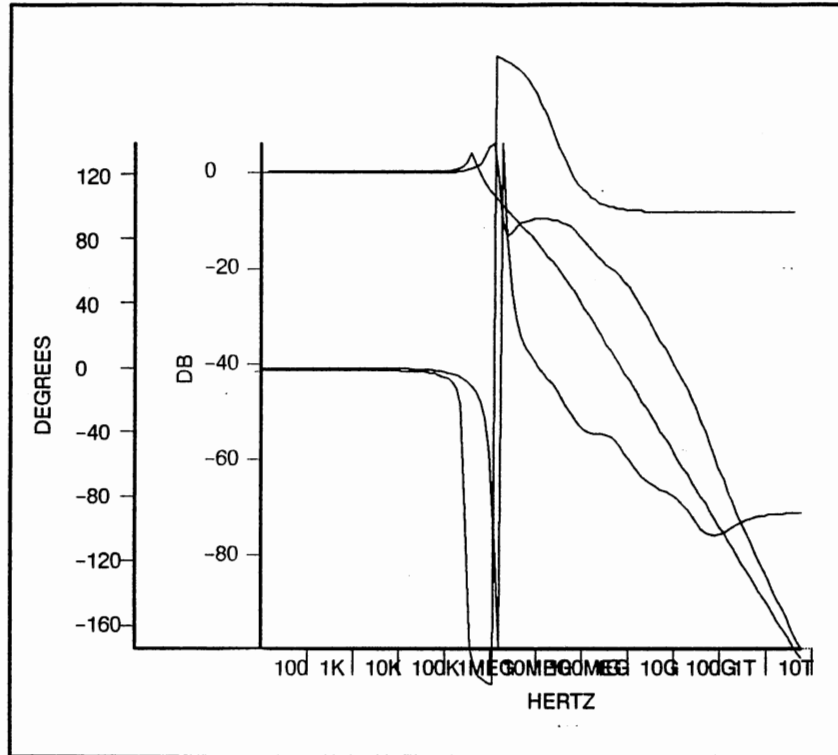


Figure 66. Comparison of the Circuit and Simplified Model's Unity Gain AC Responses

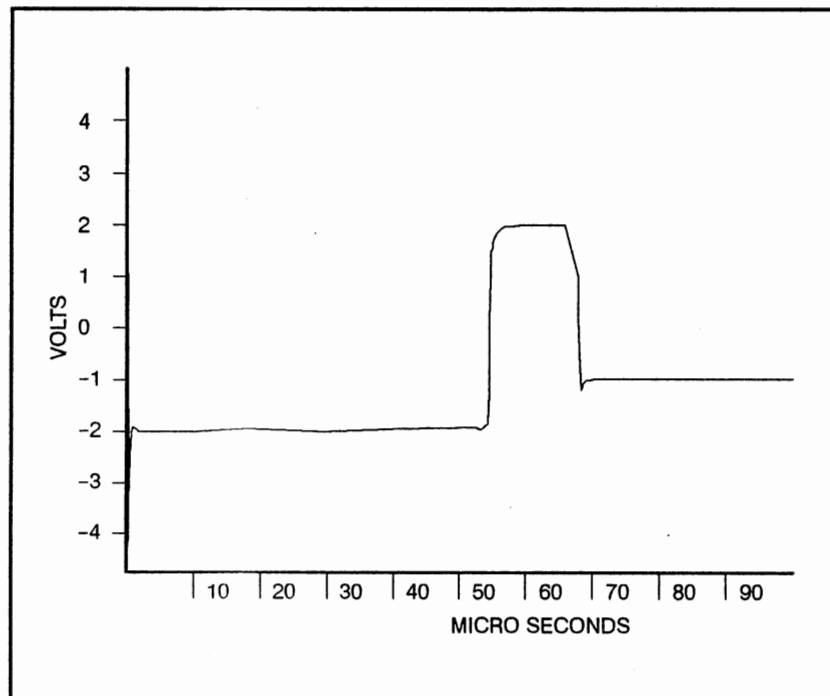


Figure 67. Results of Transient Analysis of Simplified Macro Model

CHAPTER IV

SYSTEM BEHAVIOR AND SIMULATION RESULTS

Introduction

An analog median filter based on neural network principles has been proposed along with a specific CMOS analog electronic implementation of it. Methods for modelling and evaluating the design have also been presented. In this chapter, the expected performance of the proposed circuit and system will be presented and compared with the simulation results. Sources of error in the system's performance will be discussed and quantified. Finally, the dominant error effects will be identified.

The Expected System Behavior

Analysis of the neural median filter behavior falls neatly into two categories: accuracy and convergence time. Naturally, for the system to be an effective solution to the median filtering problem, the result of the system's computation of the median would have to actually be the median and the result would have to be available within a reasonable amount of time.

The Expected Convergence Time

The expected convergence time characteristics of the proposed system and comparator design separates into the system dynamics and the comparator dynamics. The system dynamics can be described as the changing of the charging and discharging currents into the integrating capacitor as the output

voltage/ median approximation sweeps past input voltage values on its way toward convergence. The comparator dynamics, on the other hand, are those delays that are internal to the comparator units.

Since the total convergence time depends quite heavily on the initial condition of the network and the input window of voltages presented to the network, one can only estimate a worst case convergence time to represent the dynamic characteristics of the proposed system.

This worst case estimation problem does not have an obvious solution. The dynamic characteristics of the individual comparators are complex and non-linear enough that no straightforward derivation of the bounds on convergence time are possible.

In the interest of simplicity, a best guess approach has been taken. The total system delay or convergence time can be described by equation IV.1 where T_s is the system dynamic delay and T_c is the internal comparator delay.

$$\text{Equation IV.1} \quad T_{converge} = T_{system} + T_{comparator}$$

In the case of the comparator design that has been presented in this thesis, the internal comparator delay is so huge compared to the system delay that one is tempted to ignore the system delay. In the interest of preserving the reference to the physical mechanism that is performing the median calculation, however, an attempt to estimate it has been made.

The worst case convergence time is the case where the system is evaluating the median of a window with values distributed fairly evenly over the input voltage range and the system's initial estimate of the median is one or the other voltage rail. In this instance, the output voltage must sweep past almost half of the input voltage values to make it to the true median value. Between each of those input voltage values, the derivative of the output voltage decreases by even increments, getting slower and slower as it passes each input voltage val-

ue. Also, as the output voltage passes each input voltage value, there will be a delay as the internal delays of the comparators converge to take the new dynamic conditions into account. Equation IV.2 describes an estimate of that total delay value for an input window that is N bits long with each comparator unit having an internal delay of t_c and the system having a basic worst case system delay of t_s . This worst case system delay represents the amount of time it would take one unit to charge the output capacitance over the entire voltage range after the units internal poles had converged.

$$\text{Equation IV.2} \quad T_{converge} = \sum_{i=0}^{\frac{N-1}{2}} \frac{t_s}{2i+1} + \frac{(N-1)}{2} t_c .$$

For the comparator proposed in this thesis, t_s is approximated by equation IV.3 where I_{rail} is the output current rail, V_{DD} is the supply voltage difference, C_{out} it the output capacitance, and N is the number of units in the input window.

$$\text{Equation IV.3} \quad t_s = \frac{C_{out} N V_{DD}}{I_{rail}} .$$

For the proposed comparator, this number is $N * 7.24$ nano-seconds.

An approximation for t_c is a little more difficult. It would certainly seem reasonable to use some scaled version of the time constant of the dominant internal pole as an estimate. For a normal exponential decay, three time constants corresponds to 95% convergence and five time constants corresponds to 99% convergence. Since the only concern we have is that the effects of the input voltage have propagated through the comparator to the point that their effects can be seen at the output, neither of these figures is really appropriate. A better approximation would be a digital switching threshold of half the voltage swing of the internal pole. The time required to converge to the halfway point is approximately $2/3$ of the dominant pole time constant.

Under unity feedback conditions, the dominant pole of the proposed comparator design is located at approximately 1.6 kHz. The corresponding time constant for this pole is 99 micro-seconds. Two-thirds of this number is 66 micro-seconds.

Therefore, using the equations presented above, the expected worst case convergence time for different sized networks can be calculated. Table I, below lists such values for 3,5,7,9, and 25 bit input windows.

TABLE IV
EXPECTED CONVERGENCE TIMES FOR
DIFFERENT SIZED INPUT WINDOWS

SIZE(pixels)	TIME (micro-seconds)
3	66
5	133
7	198
9	265
25	795

These numbers are drastically dominated by the dominant pole of the proposed comparator design.

The Expected Accuracy

The remaining analysis to be done to determine the expected behavior of the proposed circuit concerns the accuracy of the median value calculated by

the system. The error of the calculated median value with respect to the actual median value separates into four major categories: algorithmic or system error; output resistance error; stability; and other sources of error including statistical parameter variation.

The Algorithmic or System Error. The algorithmic or system error is that error that is inherent to the proposed method of calculating the median. Another way to describe this error would be to ask the question: If everything about the system was ideal, that is to say, no internal delay, infinite output resistance, perfectly matched output current rails, no process variation, etc., would the output voltage be guaranteed to converge to the median value?

This question arises because the system's median calculation depends on there being the same number of samples larger than the median value as there are smaller than the median value. This is not the case when the input window contains discontinuities resulting in more than one pixel having the median value. In this instance, as soon as the output value reaches the median value, some of the processing units, or comparators, that had been saturated, that is, operating at the extreme of their output ranges, would be pulled back into the linear region perhaps allowing the output value to drift back in the direction of values it had already passed over. The discussion below guarantees that the output value will always converge to the median value within a margin of error equal to one half the width of the processing units' linear region.

If you have a sequence of numbers of length N , odd, arranged in ascending order, then the median value of that sequence is the $m = (N + 1)/2$ value. There are K values above or below the median and L values below or above the median. There are M numbers with the value m , the median value. So,

$$\text{Equation IV.4} \quad M + K + L = N ,$$

$$\text{Equation IV.5} \quad L \leq K \leq \frac{(N-1)}{2}$$

by virtue of the definition of the median. This notation implies that K represents the 'side' of the sorted input window with more samples that are not the median value.

If $K = L$, then the sequence is an unambiguous, balanced sequence of numbers, that is, the neural median filter will converge to the median value precisely because there are as many values in the input window above the median as there are below the median.

If $K > L$, then an error will be produced in the convergence and the sequence is called unbalanced because the saturated units in the network will not cancel out precisely. However, if M , the number of values at the median is greater than $K - L$, the number of unbalanced saturated units, the error is a fraction of the width of the linear region of the units' response curve.

So, is

$$\text{Equation IV.6} \quad M > K - L$$

true?

We know that:

$$\text{Equation IV.7} \quad 0 \leq K - L \leq \frac{(N-1)}{2}$$

by definition, and

$$\text{Equation IV.8} \quad 1 \leq M \leq N$$

since there has to be at least one value at the median and, at most, all of the values will be the median.

Also by definition:

So,

$$\text{Equation IV.9} \quad N = M + K + L = M + (K - L) + 2L .$$

$$\text{Equation IV.10} \quad (K - L) = N - M - 2L$$

and the question of the veracity of Equation IV.6 becomes:

$$\text{Equation IV.11} \quad M > N - M - 2L ? \text{ or}$$

$$\text{Equation IV.12} \quad 2M > N - 2L .$$

Case 1: $L = 0$.

$$2M > N ?$$

$$M = N - K - L = N - K \text{ so,}$$

$$2(N - K) > N ?$$

case1a: $K = 0$

$$2N > N \text{ QED}$$

case1b: $K = (N - 1)/2$

$$2(N - (N-1)/2) > N ?$$

$$2N - N + 1 > N ?$$

$$N + 1 > N \text{ QED}$$

Case 2: $L = (N-1)/2$

$$2M > N - 2L ?$$

$$2M > N - 2(N-1)/2 ?$$

$$2M > 1 \text{ QED}$$

which is always true since $M \geq 1$, so $M > K - L$ is guaranteed and the error can be derived as follows.

Convergence is reached in the unbalanced case when

$$\text{Equation IV.13} \quad M A_V (P_{in} - M_{out}) = \mp (K - L) .$$

So

$$\text{Equation IV.14} \quad \epsilon = \pm \frac{(K - L)}{MA_V} .$$

Equation IV.14, above, describes the magnitude of the error generated in the neural median filter when the input window contains more than one value equal to the median value. All other sources of error are associated with the realities of implementing this system in analog electronics. The most obvious of these realities is the fact that the proposed comparator design has a finite output impedance and therefore the output current rail will change with the output voltage.

Output Resistance Error. Figure 68, below, describes the neural median filter in compact form where I_{out} is the sum of the output currents of all the processing units, C_{out} is the total load capacitance, R_{out} , is the result of placing all the units' output resistances in parallel, and I_C is the total resulting charging or discharging current into the capacitor C_{out} .

Equation IV.15, below, describes the current I_C .

$$\text{Equation IV.15} \quad I_C = I_{out} - \frac{V_{out}}{R_{out}} = \sum_{i=0}^{N-1} f_i(V_{out}, V_i) - \frac{V_{out}}{R_{out}} .$$

where

$$f_i (V_{out}, V_i) = - I_{rail} \text{ sign } (V_{out} - V_i) .$$

except that the transition region between the two rails of $f_i()$ has a finite width and slope. Figures 69 and 70 illustrate $f_i()$ and $\sum f_i()$ respectively.

As you can see from Figure 70, the function $I_{out} = \sum f_i()$ crosses zero at the median value $\frac{V_{(N+1)}}{2}$ as it should. The effect of adding an output resistance to the system is that the whole function is tilted with a negative slope. See Figure 71, below.

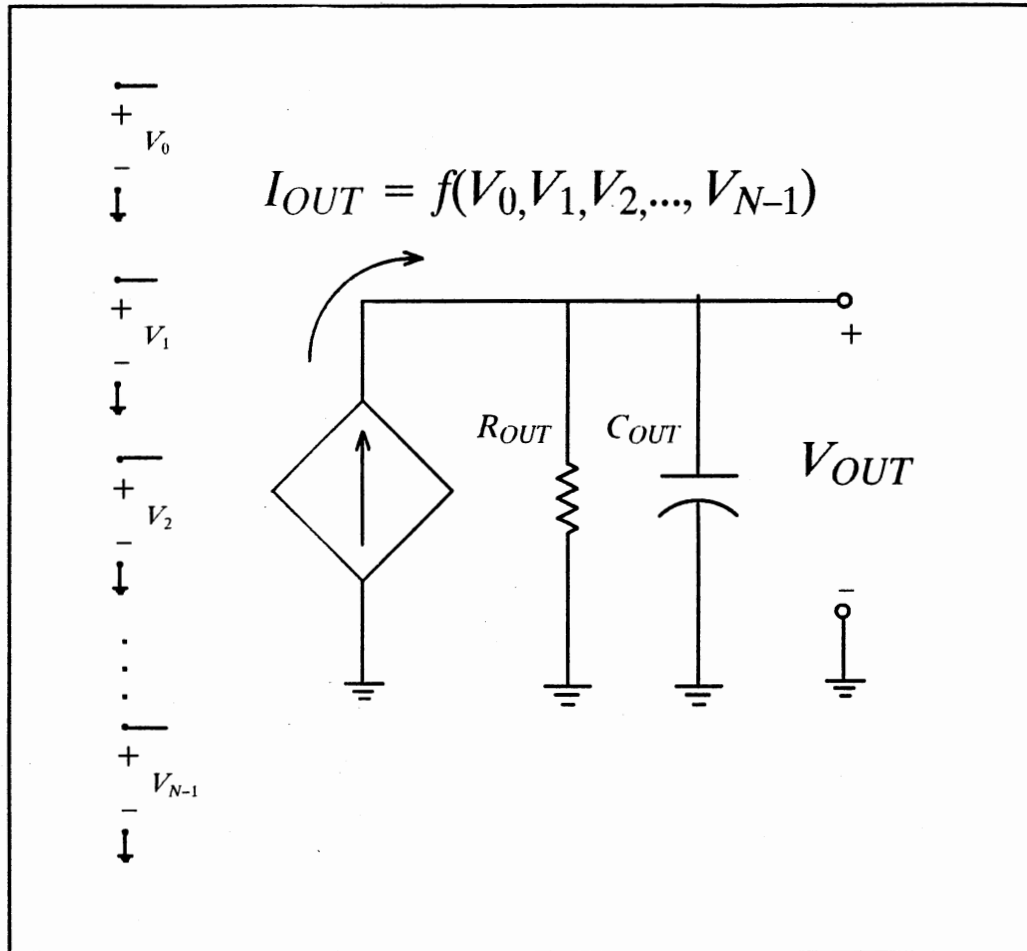


Figure 68. Compact Schematic Description of the Neural Median Filter

It is easily seen that as long as the added slope does not move output current at the median filter out of the associated transition region, the output voltage value will not stray more than half the transition region width away from the true median value. Figures 72, (a) and (b), compare the effect of two different output resistances on the median portion of the output current function, illustrating this effect.

To reiterate the implications of these observations, as long as the output resistance is such that deflection of the output curve is not greater than the output current rail, then the error in the output voltage will be less than one half the width of the output transition region or I_{rail}/G_{trans} . If the output is resistance is

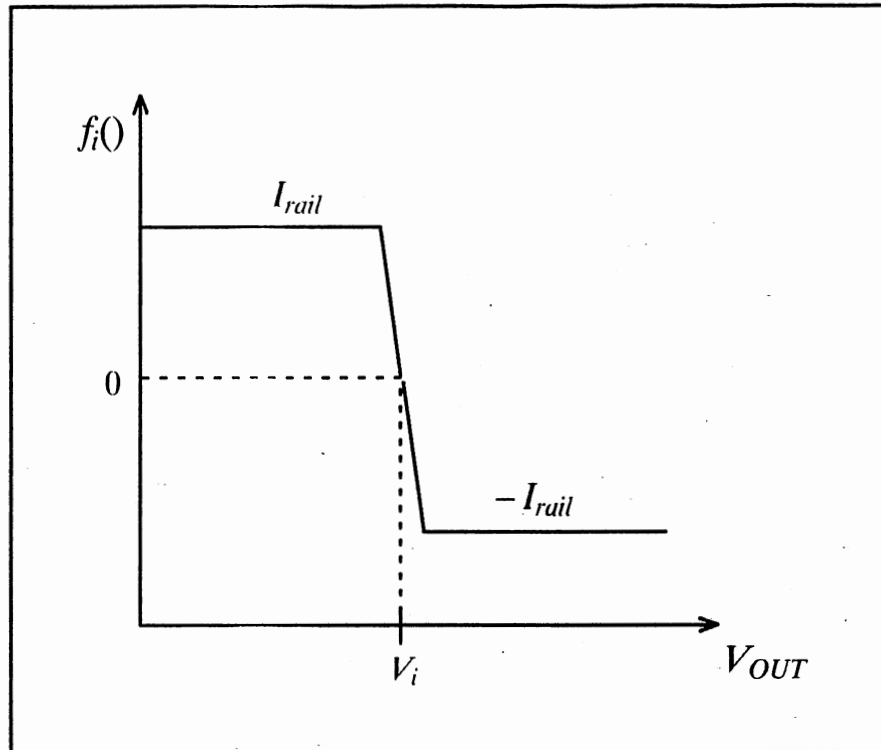


Figure 69. Illustration of $f_i()$

small enough that the error current pulls the output voltage out of the transition region, then the error is, in general, unpredictable, depending heavily on what the input voltages are.

Stability. The third source of error that was mentioned was stability. This is, in fact, not a source of error but a question of whether the design will work or not. If the load capacitance for the system is merely the sum of all the output capacitances, and the total output resistance is just the reciprocal of the sum of the output conductances of the comparators, and the individual comparators are stable, then total system will be stable because the output pole location will be the same no matter how many units are placed in parallel. For every incremental increase in the output capacitance, there will be an incremental decrease in the output resistance keeping their product constant insuring the stability of the

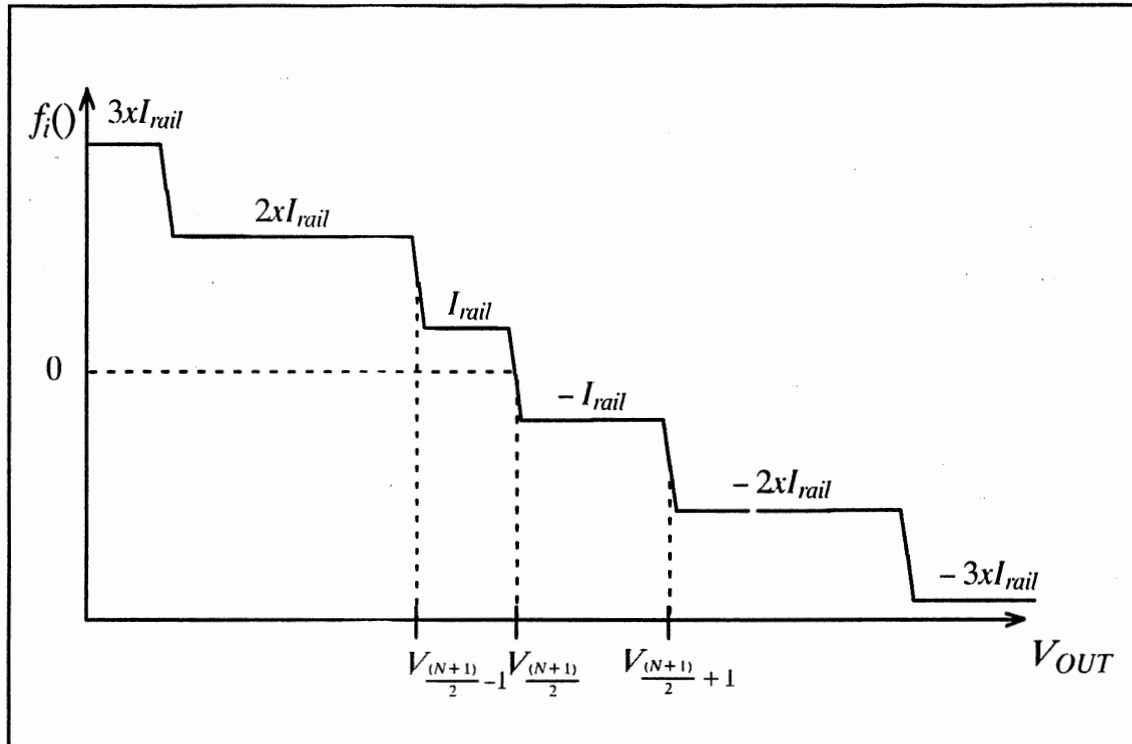


Figure 70. Illustration of $\sum f_i(V)$

system.

Process Variation. The final source of error is process variation. At this time this can be quantified by saying that if the total current error from process variation is less than the current rail of one comparator, then the error will be limited to one-half the transition width of the comparator. If the process variation current error is greater than that threshold, the error is, again, unpredictable in general.

Total Error Effects. In summary of the error analyses above, in order to keep the error within a predictable limit, it is necessary to keep the total output current error, less than the magnitude of the comparator current rail. A good portion of this margin for error is taken up by the system error from unbalanced

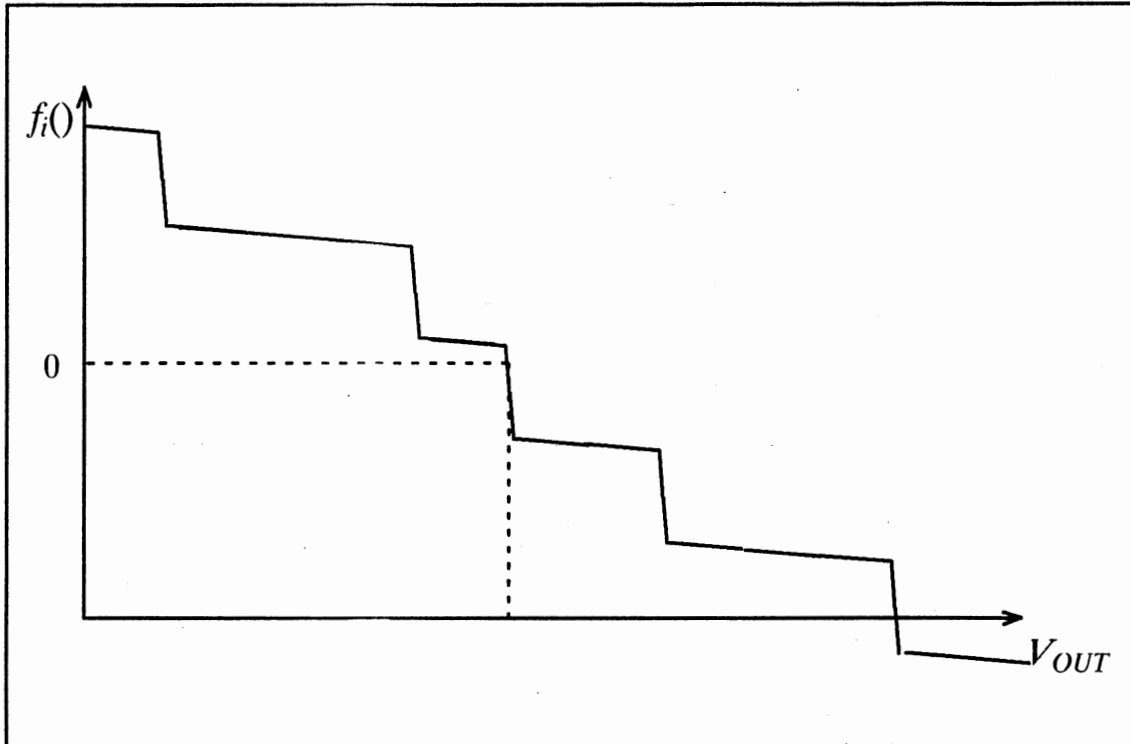


Figure 71. Effect of Adding Output Resistance to $I_{OUT} = \sum f_i(t)$

input windows. This could be anywhere from 50% for a 3 sample window to 96% for a 49 input window. The remaining margin must be greater than the sum of all the electronic sources of error such as output resistance and process variation.

In the following sections, the results of Runge–Kutta system level simulations isolating and combining all these effects will be presented and their correlation with the expected values described in the preceding sections will be evaluated. Finally, the results of Spice macro model simulation of the proposed design will be presented and evaluated

Runge–Kutta Simulation Results

Before the Runge–Kutta system level simulation results are presented, the reader should be reminded of the failings of this approach at modeling the pro-

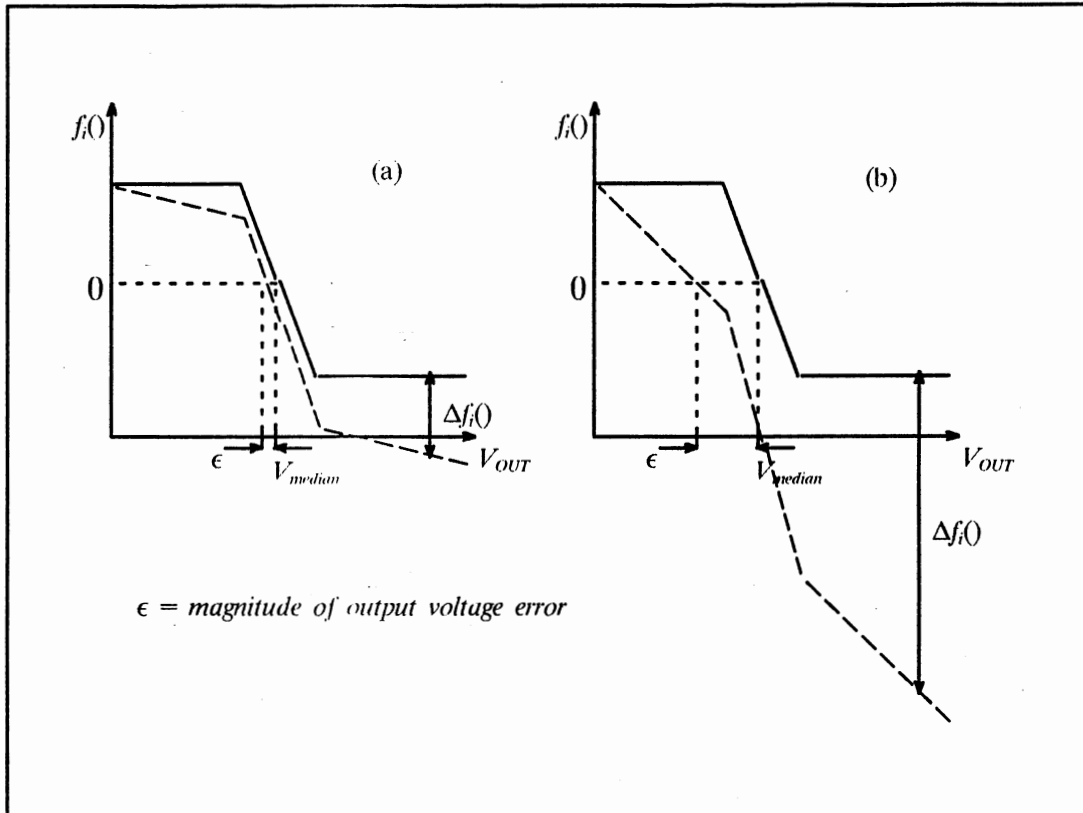


Figure 72. (a) Large Output Resistance's Effect on 'Median' Portion of the Curve, and (b) Small Output Resistance's Effect on the 'Median' Portion of the Curve.

posed comparator design: mainly, that simulations using the actual pole locations, current rails, transconductances, etc., were ridiculously slow so that the advantages of using a dedicated simulation over Spice simulation were lost. Therefore, the usefulness of the Runge-Kutta system level simulations is reduced to a proof-of-concept level. The following table lists the characteristics of the comparator unit modeled by the Runge-Kutta system level simulation.

TABLE V
NOMINAL SPECIFICATION FOR RUNGE-KUTTA
MODELLED COMPARATOR

CHARACTERISTIC	VALUE
Total Transconductance	100 μ A/V
Output Resistance	3 MegaOhms
Voltage Gain	300 (49.5 dB)
Output current rails	+/- 10 μ A
Output capacitance	200 nF
Rolloff frequency	1.66 Hz (yes Hz)
Estimated Slew Rate	20 msec / Volt
Estimated Internal Delay	35usec

Many Runge-Kutta simulations were run to evaluate the functionality of the proposed system under different conditions and with different levels of ideality. Specifically, simulations were run to independently evaluate the identified sources of error (delay or stability, output resistance, process variation, and unbalanced input windows) and finally to evaluate the combination of all of them. All of these simulations were run for 3, 5, and 9 sample input windows. Figure 73, below, illustrates a sample input window from the 9 sample input window simulations and Figure 74, following, illustrates the convergence of the output voltage to the median value of the input window.

Table 3, below summarizes the expected and the actual results of the Runge-Kutta system level simulations.

It should be explained that for all of the simulations, the output resistance was adjusted to insure small errors and the process variation error current was set also set for the same reasons. The purpose was to illustrate that one can set a lower threshold for the output resistance based on the current and voltage rails

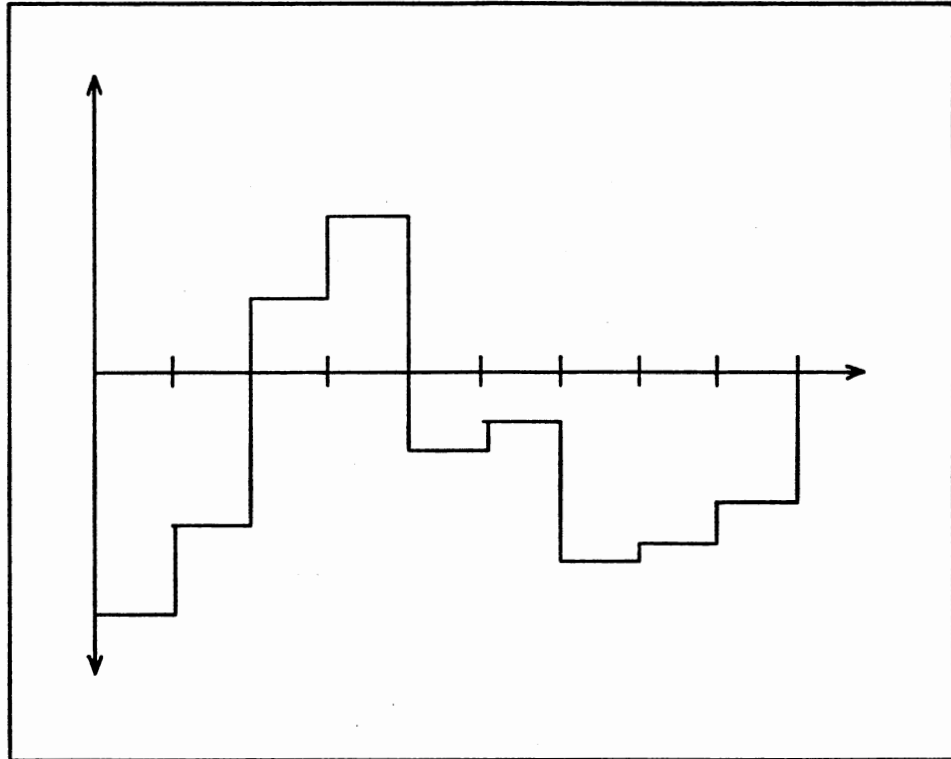


Figure 73. Example 9 Sample Input Window

of the proposed design to insure accuracy. The large magnitude of most of the prediction errors can be explained by the fact that the performance estimations were worst case. There are obviously some problems with the delay calculations, in particular for the nine sample window and for all the discontinuous window simulations. The purpose was to show that an upper limit on delay time could be set and the effort was not entirely successful.

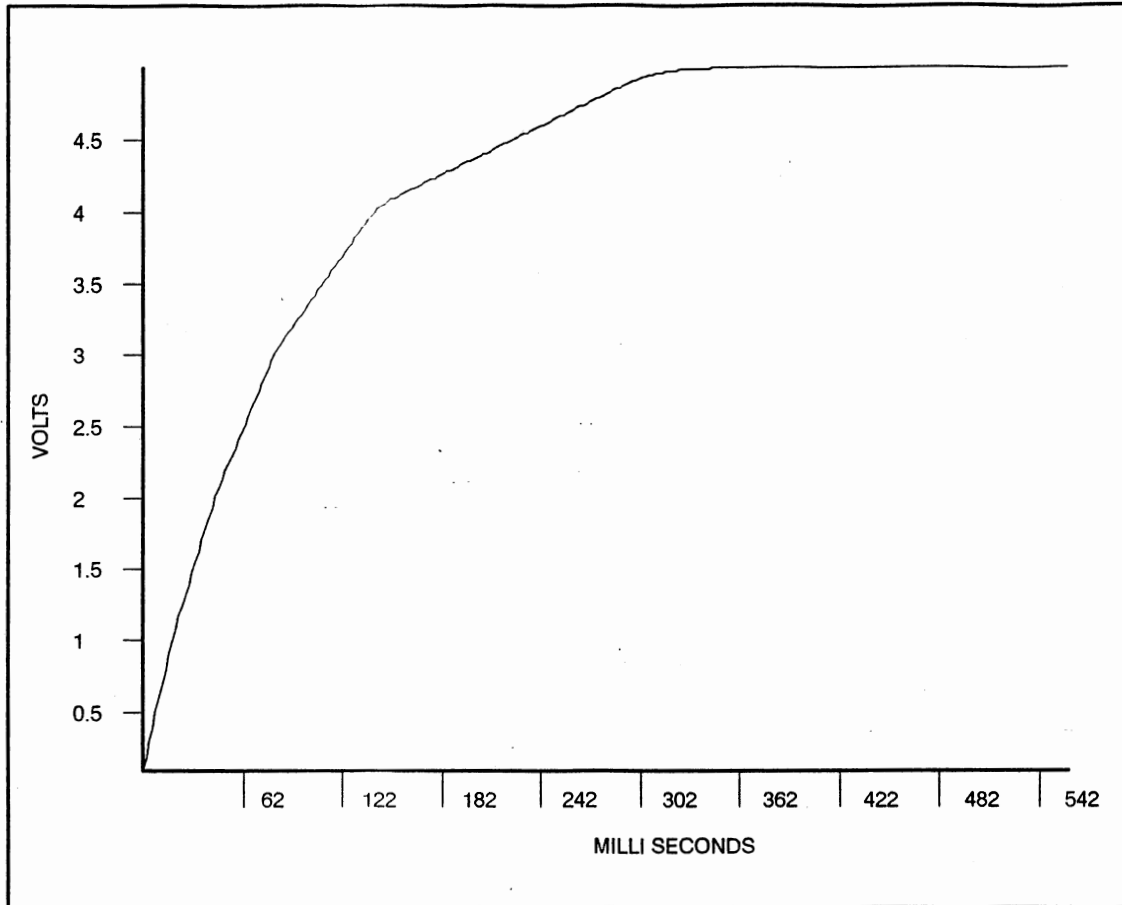


Figure 74. Example Output Voltage Convergence Curve

TABLE VI
COMPARISON OF EXPECTED AND ACTUAL
RUNGE-KUTTA RESULTS

Table 3. Comparison of Expected and Actual Runge Kutta Simulation Results

Window Length	Effect Simulated	Expected Time	Actual Time	Percent Error	Expected Accuracy	Actual Accuracy	Percent Error
3	Ideal System	166ms	147ms	-12%	0.0V	0.0V	0 %
	Unbalanced Window	"	289ms	74%	50mV	50mV	0 %
	Output Resistance	"	154ms	-8%	100mV	20mV	-80%
	Process Variation	"	121ms	-27%	"	90mV	-10%
	Internal Delay	"	139ms	-16%	0.0V	0.0V	0 %
	All Effects	"	100ms	-40%	100mV	39mV	-61%
5	Ideal System	307ms	264ms	-14%	0.0V	0.0V	0 %
	Unbalanced Window	"	440ms	43%	67mV	67mV	0 %
	Output Resistance	"	288ms	-6%	100mV	30mV	-70%
	Process Variation	"	218ms	-29%	"	10mV	-90%
	Internal Delay	"	256ms	-16%	0.0V	0.0V	0 %
	All Effects	"	351ms	14%	100mV	60mV	-40%

TABLE VI (Continued)

9	Ideal System	329ms	519ms	58%	0.0mV	0.0V	0%
	Unbalanced Window	"	751ms	128%	80mV	80mV	0%
	Output Resistance	"	638ms	94%	100mV	50mV	-50%
	Process Variation	"	433ms	32%	"	90mV	-10%
	Internal Delay	"	511ms	55%	0.0V	0.0V	0%
	All Effects	"	700ms	123%	0mV	79mV	-21%

Macro Model Simulation Results

Several Spice macro model simulations were run to evaluate the proposed comparator's actual performance in the context of the proposed neural median filter. These were, specifically, simulations of balanced and unbalanced windows covering small and large voltage ranges, and for ranges that were centered about zero and ranges that were offset toward one rail or the other. These simulations were performed for 3, 5, and 9 sample windows. Figures 75(a-f) offer examples of the six different types of input windows that were presented to the proposed system.

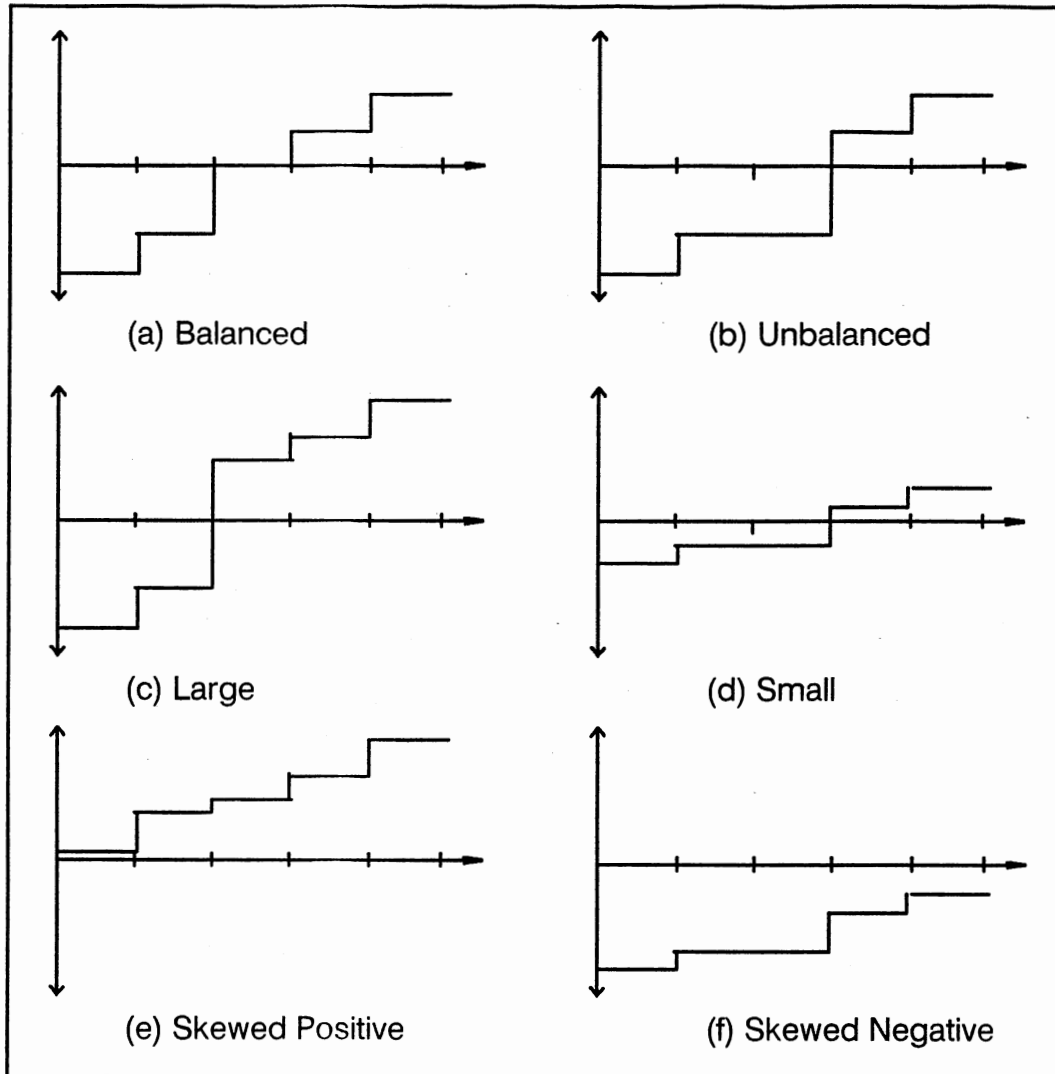


Figure 75. Sample Input Windows

TABLE VII
SUMMARY OF SPICE MACRO-MODEL/SYSTEM LEVEL
SIMULATIONS

Window Length	Effect Simulated	Expected Time	Actual Time	Percent Error	Expected Accuracy	Actual Accuracy	Percent Error
3	Balanced Window	66us	89us	34%	7mV	448uV	-94%
	Small Window	"	94us	42%	"	2.7mV	-62%
	Skewed Positive	"	90us	36%	"	28mV	300%
	Skewed Negative	"	71us	8%	"	10mV	43%
5	Balanced Window	132us	124us	-6%	"	.4mV	-95%
	Unbalanced Window	"	148us	12%	"	3mV	-67%
	Large Window	"	67us	-52%	"	1mV	-86%
	Small Window	"	60us	-56%	"	1mV	-86%
	Skewed Negative	"	59us	-55%	"	.3mV	-96%
	Skewed Positive	"	203us	154%	"	3mV	-67%

TABLE VII (Continued)

9	Balanced Window	265us	120us	-55%	"	.4mV	-95%
	Unbalanced Window	"	177us	-33%	"	6mV	-15%
	Large Window	"	59us	-78%	"	5mV	-29%
	Small Window	"	112us	-57%	"	.3mV	-96%
	Skewed Negative	"	71us	-73%	"	6mV	-15%
	Skewed Positive	"	428us	61%	"	8mV	14%

The majority of the results detailed above indicate that the proposed system design will work reasonably well in the accuracy sense. The fact that the majority of the delays detailed above are attributable to the internal poles of the comparator indicates that it may be possible to improve the response time by a redesign effort. A simulation that was not detailed here, however, shows a large departure from the expected behavior.

The simulation was of a 9 sample system. The input window was a step function from the negative voltage rail to the positive voltage rail. Under these conditions, the error current from the output resistance was large enough to drag the output voltage down to the 2 volt range when the answer should have been 5 volts. It also took around 800 usec to converge. These results indicate that while the results shown above are promising, the proposed comparator design has definite accuracy deficiencies as well.

It should also be noted here that no account at all has been given here to the effects of process parameter variation in these macro model simulations. While the time estimations are often pessimistic, they are not always so. Also, the accuracy estimations are not always pessimistic as they were intended. These results make it clear that a better method of calculating the worst case accuracy and delay needs to be developed for this system. The methods derived and presented are not fully satisfactory.

The final chapter of this thesis follows and will offer conclusions about the design and suggestions for future work on this subject.

CHAPTER V

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

Introduction

This thesis has presented an approach to solving the problem of median filtering. After describing the problems that median filtering is intended to solve and the problems inherent in serial implementations of median filtering, a review of hardware implementations of serial and parallel median filters was presented along with a review of neural network principles and hardware implementations. An analog CMOS, parallel median filter based on neural network architectural principles was proposed and the major building block of the system was identified, a comparator. A design for the required comparator in the ORBIT CMOS process was proposed and analyzed. Modelling techniques to evaluate the effectiveness of the design were developed and applied to the proposed comparator design in the context of the neural median filter system. Methods to estimate the response time and accuracy of the proposed comparator and system were proposed and compared to simulation data.

Summary and Conclusions

The results of the simulations detailed in Chapter IV clearly indicate that, while the proposed design shows some promise, there are enough problems that a redesign effort should be performed. The current design is much too slow to compete with other methods of median filtering. Simulation results indicate that one could only expect around 2 Kilo-medians per second out of the pro-

posed design. This is in no way competitive with the 20 Mega-medians per second reported in the literature review. Also, while most simulations show enough accuracy to qualify the system as accurate to within 10 bits, there are some example of atrocious and unpredictable levels of error. The small output resistance is responsible for this bad performance, again warranting a redesign.

On the positive side, however, this design effort has proven that an analog CMOS implementation of the neural median filter system is possible and that the resulting system behaves fairly close to expectations. This brings the development effort one step farther away from 'blue-sky' brainstorming and one step closer to successful implementation.

Suggestions for Future Investigation

There are a number of directions that could be taken from this point.

The most obvious direction is a simple redesign of the proposed comparator. The goals of this redesign effort should be to make the output pole the dominant factor in the transient response of the circuit while increasing the output resistance to a sufficient size to insure convergence to within some small voltage of the median voltage value. This small voltage should represent the maximum deviation from the median value and therefore will define the accuracy of the system by comparison with the voltage rail difference. For example, if the voltage rail difference is 10V and the maximum deviation from the median value is 10mV, then the accuracy is one part in 1000 or close to 10 bits of accuracy. The error analysis shown in Chapter IV makes it clear that for the system to be expected to work, the output resistance must be large enough that the maximum error current $(V_{dd} - V_{ss}) / R_{out}$ is smaller than the output current rail minus the output current error caused by discontinuities and unbalanced input windows.

Two possible alterations to the circuit are shown in Figures 76 (a) and (b). The first possibility, the folded cascode, has the advantage of using fewer transistors however, it may not be as easy to work with as the second example with a simple common gate buffered output stage. Also, it is not immediately apparent how these alterations will affect the pole-zero locations of the circuit. Great attention should be given to the frequency response, and therefore the transient behavior, of any redesign of this circuit.

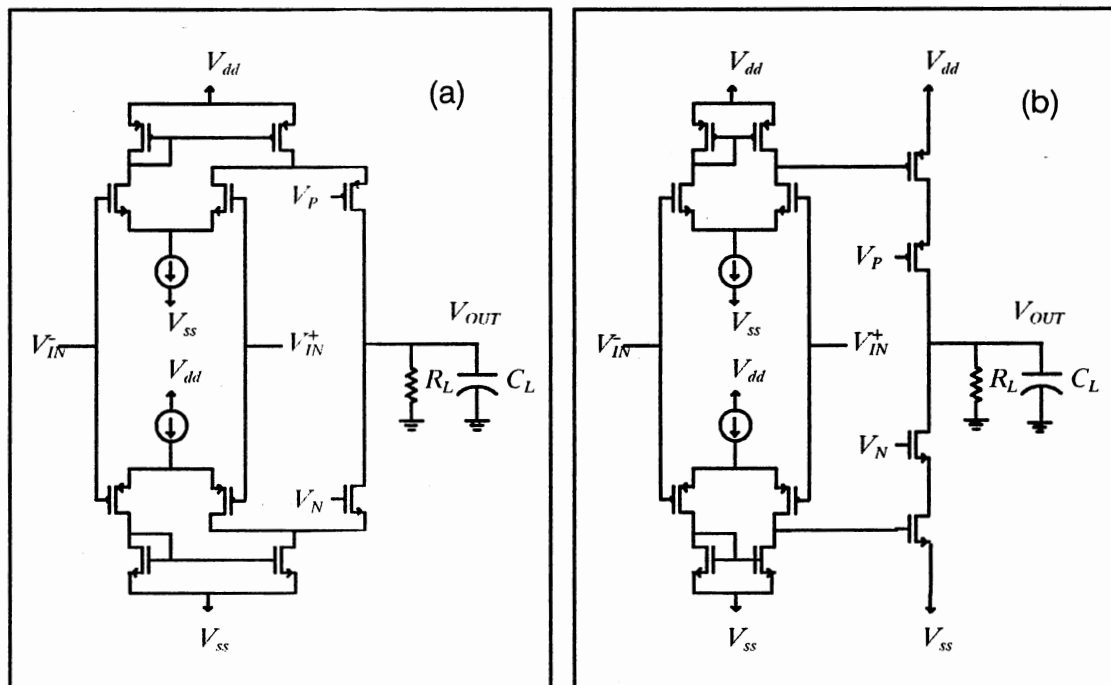


Figure 76. (a) Folded Cascode. (b) Cascoded Output Stage

Other possibilities for investigation include an evaluation of the Gain-Bandwidth tradeoffs of the circuit as they effect accuracy and response time. Also, no attention has been paid to process variation between dies and on the same die (other than grossly in the Runge-Kutta simulations). Also to be addressed in the

future are area consumption issues and the support hardware that would be necessary to make the proposed analog design practical.

These issues will strongly affect the practicality of the design in terms of speed and chip area. If the parallel, analog solution is faster and as accurate as any other median filter implementations, but the necessary support hardware is serial, then most or all of the inherent advantage of parallelism is lost.

Finally, an all digital implementation of the proposed system should be attempted and evaluated. Perhaps by using asynchronous logic, speeds could be achieved that rival or surpass other, present digital hardware implementations of median filtering. One could see how an all digital solution would eliminate some of the chip area, accuracy, and process variation problems of the analog solution proposed in this thesis. But a synchronous solution is liable to be much slower than the hardware sorting systems already implemented.

This becomes apparent when one considers that, to avoid overshooting the median value, the increment/decrement size or the output variable must be below some threshold value. This is equivalent to stability/settling-time concerns in analog design. To insure that the median value is not overshoot, one must over-quantize the output variable so that if every unit in the system is incrementing or decrementing its value, the median value can not be overshoot in a single iteration.

It is fairly straightforward to realize that the output variable must be quantized to $N + \log_2(M)$ bits where N is the desired bits of accuracy and M is the number samples in the input window. Unfortunately this over-quantization also means that to reach the median value, the output variable may have to cover the entire range of values available to it by 1 bit increments so that $2^{(N + \log_2(M))}$ iterations are required. On a 20 MegaHz clock, the time required for a 9 sample window with 10 bits of accuracy would be 461 micro seconds. Of course faster

clocks would improve that number as would smaller windows and lower desired accuracy.

In any case, response time must be an issue in any future design effort toward realizing the proposed median filter system, whether analog or digital.

So in conclusion, this thesis has demonstrated the value of the proposed neural median filter concept by illustrating its implementability. It has not proven the neural median filter's superiority over other more traditional implementations, but it has left suggestions for directions to develop the idea further.

REFERENCES

1. Agranat, and Yariv, 'Semiparallel Microelectronic Implementation of Neural Network Models Using CCD Technology,' Electronics Letters, Volume 23, No. 11, pp. 580-581.
2. Ahmad, M. Omair, and Sundararajan, Duraisamy, 'A Fast Algorithm for Two-Dimensional Median Filtering,' IEEE Transactions on Circuits and Systems, Volume CAS-34, No. 11, pp. 1364-1374.
3. Allen, Phillip E., and Holberg, Douglas R., CMOS Analog Circuit Design, New York, NY: Holt, Rinehart, and Winston, Inc.. 1987.
4. Alspector, and Allen, 'A Neuromorphic VLSI Learning System,' Proceedings of the 1987 Stanford Conference on Advanced Research in VLSI, pp. 313-349.
5. Arce, Gonzalo R., Warter, Peter J., and Foster, Russel E., 'Theory and VLSI Implementation of Multilevel Median Filters,' ISCAS, 1988, pp. 2795-2798.
6. Beyer, William H., Ed. CRC Standard Mathematical Tables, 27th Edition, Boca Raton, FL: CRC Press, Inc., 1984.
7. Boahen, Kwabena A., Pouliquen, Phillippe O., Andreou, Andreas G., and Jenkins, Robert, E., 'A Heteroassociative Memory Using Current-Mode MOS Analog VLSI Circuits,' IEEE Transactions on Circuits and Systems, Volume 36, No. 5, pp. 747-755.
8. Bout, and Miller, 'A Digital Architecture Employing Stochasticism for the Simulation of Hopfield Neural Nets,' IEEE Transactions on Circuits and Systems, Volume 36, No. 5, pp. 732-738.
9. Cleary, John G., 'A Simple VLSI Connectionist Architecture,' Proceedings of the First International Conference on Neural Networks, San Diego, CA 1987, Volume III, pp. 419-426.
10. Close, Charles M., and Frederick, Dean K., Modeling and Analysis of Dynamic Systems, Boston, MA: Houghton Mifflin Company, 1978.
11. El-Leithy, Newcomb, and Zaghlour, 'A Basic MOS Neural-Type Junction: A Perspective on Neural-Type Microsystems,' Proceedings of the First International Conference on Neural Networks, San Diego CA 1987, Volume III, pp. 469-477.
12. Furman, and Abidi, 'An Analog CMOS Backward Error-Propagation LSI,' Proceedings of 22nd Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, GA, November 1988, pp. 645-652.

13. Habib, Akel, 'A Digital Neuron-Type Processor and Its VLSI Design,' IEEE Transactions on Circuits and Systems, Volume 36, No. 5, pp. 739-746.
14. Hochet, R., 'Multivalued MOS Memory for Variable Synapse Neural Networks,' Electronics Letters, Volume 25, No. 10, pp. 669-670.
15. Hodges, David A., and Jackson, Horace G., Analysis and Design of Digital Integrated Circuits, New York, NY: McGraw-Hill Book Company, 1983.
16. Hopfield, John J., 'Neurons with Graded Response Have Collective Computational Properties Like Those of Graded Neurons,' Proceedings of the National Academy of Sciences, USA, Volume 81, pp. 3088-3092.
17. Howard, Richard E., Schwartz, Daniel B., Denker, John S., Epworth, Roger W., Graf, Hans Peter, Hubbard, Wayne E., Jackel, Lawrence D., Straughn, Brian L., Tennant, D. M., 'An Associative Memory Based on an Electronic Neural Network Architecture,' IEEE Transactions on Electronic Devices, Volume ED-34, No. 7, July 1987, pp. 1553-1556.
18. Huang, Thomas S., Yang, George J., Tang, Gregory Y., 'A Fast Two-Dimensional Median Filtering Algorithm,' IEEE Transactions on Acoustics, Speech, and Signal Processing, Volume ASSP-27, No. 1, pp. 13-18.
19. Jackel, L. D., Grap, H. P., and Howard, R. E., 'Electronic Neural Network Chips,' Applied Optics, Volume 26, No. 23, pp. 5077-5080.
20. Karaman, Mustafa, Levent, Onural, and Abdullah, Atalar, 'Design and Implementation of a General-Purpose Median Filter Unit in CMOS VLSI,' IEEE Journal of Solid-State Circuits, Volume 25, No. 2, pp. 505-513.
21. Ko, Sung-Jea, Lee, Yong Hoon, and Fam, Adly T., 'Software and VLSI Algorithms for Recursive Median Filters,' ISCAS, 1989, pp. 417-420.
22. Lee, Charng Long, and Jen, Chein-Wei, 'A Novel Design of Binary Majority Gate and Its Application to Median Filtering,' Source Unknown, 1990.
23. Moopenn, A., Lambe, John, and Thakoor, A. P., 'Electronic Implementation of Associative Memory Based on Neural Network Models,' IEEE Transactions on Systems, Man, and Cybernetics, Volume SMC-17, No. 2, pp. 325-331.
24. Moopenn, Thakoor, Duong, and Khanna, 'A Neurocomputer Based on Analog-Digital Hybrid Architecture,' Proceedings of the First International Conference on Neural Networks, San Diego, CA 1987, Volume III, pp. 479-486.
25. Raffel, Mann, Berger, Soares, and Gilbert, 'A Generic Architecture for Wafer-Scale Neuromorphic Systems,' Proceedings of the First International Conference on Neural Networks, San Diego, CA 1987, Volume III, pp. 501-513.
26. Reed, Russel D., and Geiger, Randall L., 'A Multiple Input OTA Circuit for Neural Networks,' IEEE Transactions on Circuits and Systems, Volume 36, No. 5, pp. 767-769.

27. Satyanarayana, Tsividis, 'Analogue Neural Networks with Distributed Neurons,' Electronics Letters, Volume 25, No. 5, pp. 302–304.
28. Schwartz, Daniel B., Howard, Richard E., and Hubbard, Wayne E., 'A Programmable Analog Neural Network Chip,' IEEE Journal of Solid-State Circuits, Volume 24, No. 2, pp. 313–319.
29. Shimabukuro, R. L., Reedy, R. E., Garcia, G. A., 'Dual Polarity Non-volatile MOS Analogue Memory Cell for Neural-Type Circuitry,' Electronics Letter, Volume 24, No. 19, pp. 1231–1232.
30. Sivilotti, Missimo A., Emerling, Michael R., and Mead, Carver A., 'VLSI Architectures for Implementation of Neural Networks,' The 1987 Snowbird Conference on Neural Networks for Computing, AIP Conference Proceedings #151, pp. 408–413.
31. Tsividis, and Anastassiou, 'Switched Capacitor Neural Networks,' Electronics Letters, Volume 23, No. 18, pp. 958–959.
32. Tsividis, Y., and Satyanarayana, S., 'Analogue Circuits for Variable Synapse Electronic Neural Networks,' Electronics Letters, Volume 23, No. 24, pp. 1313–1314.
33. Verleysen, Sirletti, Vandemeulebroecke, and Jespers, 'Neural Networks for High-Storage Content-Addressable Memory: VLSI Circuit and Learning Algorithm,' IEEE Journal of Solid-State Circuits, Volume 24, No. 3, pp. 562–568.

APPENDIXES

APPENDIX A

EXAMPLE SPICE INPUT DECK FOR THE PROPOSED COMPARATOR DESIGN

* Comparator design
 * begun on computer 6/25/91
 *

* Analysis Section

.WIDTH IN = 80 OUT = 80

*.DC VPLUS -.1 .1 .25M

*.PUNCH DC I(VOUT) V(11 13)

*.PRINT DC I(VOUT)

.AC DEC 10 100 1E14

.PUNCH AC VDB(4) VP(4)

*.PZ 6 0 4 0 VOL

.INITIAL OP 22 -3.915 23 -3.918 31 2.498 32 3.895

+33 3.896 1 5.000 2 -5.000 3 3.722

+4 0.106 5 -1.402 6 -4.25E-04 7 0.000

+8 1.643 9 -3.734 11 0.452 13 -0.239

+21 -2.648

*.TRAN .1N 20N

*.PUNCH TR V(4)

*.TF V(4) VPLUS

.options itl1 = 100000 itl2 = 100000 ITL4 = 2000 RMIN = 1E-14 ITLPZ = 1000

* Power Rails and Inputs

VDD 1 0 5

VSS 2 0 -5

* -.425M offset voltage

VPLUS 6 0 DC -.425M AC 1

VMINUS 7 4 DC 0

*VOUT 4 0 0

Rout 4 0 100G

COU 4 0 2P

* Compensation

CPCOMP 13 4 .3P

CNCOMP 11 4 .3P

* Parallel Diff Amp Stage

* N CHANNEL DIFF AMP

MNLDIFF 11 6 5 2 NCH W = 100U L = 10U AD = 300P AS = 300P ASD = 106U
 ASS = 106U

MNRDIFF 3 7 5 2 NCH W = 88U L = 10U AD = 264P AS = 264P ASD = 94U
 ASS = 94U

MPLLOAD 3 3 1 1 PCH W = 38U L = 15U AD = 114P AS = 114P ASD = 44U
 ASS = 44U

MPRLOAD 11 3 1 1 PCH W = 38U L = 15U AD = 114P AS = 114P ASD = 44U
 ASS = 44U

MPROUT 4 11 1 1 PCH W = 397U L = 15U AD = 1125P AS = 1125P ASD = 381U
 ASS = 381U

```

INDIFF 5 2 9.93U
CNDIFF 5 0 17.66F
*****
* P CHANNEL DIFF AMP
*****
MPLDIFF 13 6 8 1 PCH W=375U L=15U AD=1125P AS=1125P ASD=381U
ASS=381U
MPRDIFF 9 7 8 1 PCH W=330U L=15U AD=990P AS=990P ASD=336U
ASS=336U
MNLLOAD 9 9 2 2 NCH W=10U L=10U AD=30P AS=30P ASD=16U
ASS=16U
MNRLOAD 13 9 2 2 NCH W=10U L=10U AD=30P AS=30P ASD=16U
ASS=16U
MNROUT 4 13 2 2 NCH W=70U L=10U AD=210P AS=210P ASD=76U
ASS=76U
IPDIFF 8 1 -10.2U
CPDIFF 8 0 205.1F
*****
* OUTPUT STAGE
*****
*MNOUTG 4 13 2 2 NCH W=70U L=2U AD=210P AS=210P ASD=76U
ASS=76U
*MPOUTG 4 11 1 1 PCH W=375U L=3U AD=1125P AS=1125P ASD=381U
ASS=381U
*****
* N Diff Amp Current Source
*****
*MNCTOP 5 21 22 2 NCH W=50U L=10U AD=300P AS=300P ASD=106U
ASS=106U
*MNCBOT 22 23 2 2 NCH W=50U L=10U AD=300P AS=300P ASD=106U
ASS=106U
*MNRTOP 21 21 23 2 NCH W=50U L=10U AD=300P AS=300P ASD=106U
ASS=106U
*MNRBOT 23 23 2 2 NCH W=50U L=10U AD=300P AS=300P ASD=106U
ASS=106U
*****
* P Diff Amp Current Source
*****
*MPCTOP 8 31 32 1 PCH W=188U L=15U AD=1125P AS=1125P ASD=381U
ASS=381U
*MPCBOT 32 33 1 1 PCH W=188U L=15U AD=1125P AS=1125P ASD=381U
ASS=381U
*MPRTOP 31 31 33 1 PCH W=188U L=15U AD=1125P AS=1125P
ASD=381U ASS=381U
*MPRBOT 33 33 1 1 PCH W=188U L=15U AD=1125P AS=1125P ASD=381U
ASS=381U
*****
* Bias setting transistor
*****
*Rbiasp 31 0 250K
*Rbiasn 21 0 265K
*Mbiasn 0 0 21 2 NCH W=5 L=25
*Mbiasp 0 0 31 1 PCH W=5 L=5
*****

```

* Models

* N-Channel

```
.MODEL NCH NMOS LD=.1353E-6 XJ=475E-9 TOX=41.6E-9
+VTO=0.792 UO=619 NSUB=3.632E15
+RSH=27.02 DELTA=1.731 TPG=1
+UCRIT=149.6E3 UEXP=199.4E-3 NFS=543E+9
+LEVEL=2 CJ=25.8E-6 MJSW=.999
+VMAX=44.88E+3 PB=759.4E-3 CJSW=117.3E-12 MJ=.1918
+CGSO=1.99E-10 CGDO=1.99E-10 CGBO=4.98E-10
+IS=100E-18 JS=100E-6 NEFF=5.491
+FC=500E-3
```

* P-Channel

```
.MODEL PCH PMOS LD=.240E-6 XJ=582.5E-9 TOX=41.6E-9
+VTO=-0.7879 UO=237.8 NSUB=8.461E+15
+RSH=80.41 DELTA=1E-3 TPG=-1
+UCRIT=114.4E3 UEXP=308.6E-3 NFS=307.0E+9
+LEVEL=2 CJ=278.6E-6 MJSW=.999
+VMAX=32.95E+3 PB=1.180 CJSW=45.9E-12 MJ=.416
+CGSO=1.123E-10 CGDO=1.123E-10 CGBO=3.32E-10
+IS=100E-18 JS=100E-6 NEFF=2.763
+FC=500E-3
```

* End of Spice Deck

.END

APPENDIX B

EXAMPLE SPICE OUTPUT DECK FOR THE
PROPOSED COMPARATOR DESIGN


```

SSSS  UU  UU PPPPPP PPPPPP  LL  EEEEEEEE  3333
SS  SS  UU  UU PP  PP  PP  PP  LL  EE  33  33
SS    UU  UU PP  PP  PP  PP  LL  EE  33
SS    UU  UU PP  PP  PP  PP  LL  EE  33
SS    UU  UU PPPPPP PPPPPP  LL  EEEEEEEE  333
SS  UU  UU PP  PP  LL  EE  33
SS  UU  UU PP  PP  LL  EE  33
SS  UU  UU PP  PP  LL  EE  33
SS  SS  UU  UU PP  PP  LL  EE  33  33
SSSS  UUUU  PP  PP  LLLLLLLL EEEEEEEE  3333
    
```


* SUPPLE3 version 0.03 - October 18, 1991 *

SUPPLE3 IS A GENERAL PURPOSE CIRCUIT ANALYSIS PROGRAM
 MAINTAINED AT TEXAS INSTRUMENTS BY THE
 DESIGN AUTOMATION DIVISION IN DALLAS
 PROPERTY OF TEXAS INSTRUMENTS -- FOR UNRESTRICTED
 INTERNAL
 USE ONLY -- UNAUTHORIZED REPRODUCTION AND/OR
 DISTRIBUTION
 IS STRICTLY PROHIBITED. THIS PRODUCT IS PROTECTED UNDER
 COPYRIGHT LAW AND TRADE SECRET LAW AS AN UNPUBLISHED
 WORK.
 CREATED 1991, (C) COPYRIGHT 1991, TEXAS INSTRUMENTS. ALL
 RIGHTS RESERVED.

THESE COMMODITIES ARE UNDER U. S. GOVERNMENT
 DISTRIBUTION
 LICENSE CONTROL. AS SUCH, THEY ARE NOT TO BE RE-EXPORTED
 WITHOUT PRIOR APPROVAL OF THE U.S. DEPARTMENT OF
 COMMERCE.

* INPUT LISTING *
* TEMPERATURE = 27.000 *
* RUN DATE = 1/21/92 RUN NUMBER 1 RUN TIME = 12:53:57 *
*

* Comparator design
 * begun on computer 6/25/91
 *

```

*****
* Analysis Section
*****
.WIDTH IN = 80 OUT = 80
*.DC VPLUS -.1 .1 .25M
*.PUNCH DC I(VOUT) V(11 13)
*.PRINT DC I(VOUT)
.AC DEC 10 100 1E14
.PUNCH AC VDB(4) VP(4)
*.PZ 6 0 4 0 VOL
.INITIAL OP 22 -3.915 23 -3.918 31 2.498 32 3.895
+33 3.896 1 5.000 2 -5.000 3 3.722
+4 0.106 5 -1.402 6 -4.25E-04 7 0.000
+8 1.643 9 -3.734 11 0.452 13 -0.239
+21 -2.648
*.TRAN .1N 20N
*.PUNCH TR V(4)
*.TF V(4) VPLUS
.options itl1 = 100000 itl2 = 100000 ITL4 = 2000 RMIN = 1E-14 ITLPZ = 1000
*****
* Power Rails and Inputs
*****
VDD 1 0 5
VSS 2 0 -5
* -.425M offset voltage
VPLUS 6 0 DC -.425M AC 1
VMINUS 7 4 DC 0
*VOUT 4 0 0
Rout 4 0 100G
COUT 4 0 2P
*****
* Compensation
*****
CPCOMP 13 4 .3P
CNCOMP 11 4 .3P
*****
* Parallel Diff Amp Stage
*****
* N CHANNEL DIFF AMP
*****
MNLDIFF 11 6 5 2 NCH W = 100U L = 10U AD = 300P AS = 300P ASD = 106U
ASS = 106U
MNRDIFF 3 7 5 2 NCH W = 88U L = 10U AD = 264P AS = 264P ASD = 94U
ASS = 94U
MPLLOAD 3 3 1 1 PCH W = 38U L = 15U AD = 114P AS = 114P ASD = 44U
ASS = 44U
MPRLOAD 11 3 1 1 PCH W = 38U L = 15U AD = 114P AS = 114P ASD = 44U
ASS = 44U
MPROUT 4 11 1 1 PCH W = 397U L = 15U AD = 1125P AS = 1125P ASD = 381U
ASS = 381U
INDIFF 5 2 9.93U
CNDIFF 5 0 17.66F
*****

```

```

* P CHANNEL DIFF AMP
*****
MPLDIFF 13 6 8 1 PCH W=375U L=15U AD=1125P AS=1125P ASD=381U
ASS=381U
MPRDIFF 9 7 8 1 PCH W=330U L=15U AD=990P AS=990P ASD=336U
ASS=336U
MNLLOAD 9 9 2 2 NCH W=10U L=10U AD=30P AS=30P ASD=16U
ASS=16U
MNRLOAD 13 9 2 2 NCH W=10U L=10U AD=30P AS=30P ASD=16U
ASS=16U
MNROUT 4 13 2 2 NCH W=70U L=10U AD=210P AS=210P ASD=76U
ASS=76U
IPDIFF 8 1 -10.2U
CPDIFF 8 0 205.1F
*****
* OUTPUT STAGE
*****
*MNOUTG 4 13 2 2 NCH W=70U L=2U AD=210P AS=210P ASD=76U
ASS=76U
*MPOUTG 4 11 1 1 PCH W=375U L=3U AD=1125P AS=1125P ASD=381U
ASS=381U
*****
* N Diff Amp Current Source
*****
*MNCTOP 5 21 22 2 NCH W=50U L=10U AD=300P AS=300P ASD=106U
ASS=106U
*MNCBOT 22 23 2 2 NCH W=50U L=10U AD=300P AS=300P ASD=106U
ASS=106U
*MNRTOP 21 21 23 2 NCH W=50U L=10U AD=300P AS=300P ASD=106U
ASS=106U
*MNRBOT 23 23 2 2 NCH W=50U L=10U AD=300P AS=300P ASD=106U
ASS=106U
*****
* P Diff Amp Current Source
*****
*MPCTOP 8 31 32 1 PCH W=188U L=15U AD=1125P AS=1125P
ASD=381U ASS=381U
*MPCBOT 32 33 1 1 PCH W=188U L=15U AD=1125P AS=1125P
ASD=381U ASS=381U
*MPRTOP 31 31 33 1 PCH W=188U L=15U AD=1125P AS=1125P
ASD=381U ASS=381U
*MPRBOT 33 33 1 1 PCH W=188U L=15U AD=1125P AS=1125P
ASD=381U ASS=381U
*****
* Bias setting transistor
*****
*Rbiasp 31 0 250K
*Rbiasn 21 0 265K
*Mbiasn 0 0 21 2 NCH W=5 L=25
*Mbiasp 0 0 31 1 PCH W=5 L=5
*****
* Models
*****
* N-Channel

```

```
*****
.MODEL NCH NMOS LD=.1353E-6 XJ=475E-9 TOX=41.6E-9
+VTO=0.792 UO=619 NSUB=3.632E15
+RSH=27.02 DELTA=1.731 TPG=1
+UCRIT=149.6E3 UEXP=199.4E-3 NFS=543E+9
+LEVEL=2 CJ=25.8E-6 MJSW=.999
+VMAX=44.88E+3 PB=759.4E-3 CJSW=117.3E-12 MJ=.1918
+CGSO=1.99E-10 CGDO=1.99E-10 CGBO=4.98E-10
+IS=100E-18 JS=100E-6 NEFF=5.491
+FC=500E-3
*****
```

* P-Channel

```
*****
.MODEL PCH PMOS LD=.240E-6 XJ=582.5E-9 TOX=41.6E-9
+VTO=-0.7879 UO=237.8 NSUB=8.461E+15
+RSH=80.41 DELTA=1E-3 TPG=-1
+UCRIT=114.4E3 UEXP=308.6E-3 NFS=307.0E+9
+LEVEL=2 CJ=278.6E-6 MJSW=.999
+VMAX=32.95E+3 PB=1.180 CJSW=45.9E-12 MJ=.416
+CGSO=1.123E-10 CGDO=1.123E-10 CGBO=3.32E-10
+IS=100E-18 JS=100E-6 NEFF=2.763
+FC=500E-3
*****
```

* End of Spice Deck

.END

* MESSAGE * Setting LVLSUB = 0 because a NON-Transient Analysis is requested.

```
* CAPACITOR MODEL PARAMETERS *
* TEMPERATURE = 27.000 *
* RUN DATE = 1/21/92 RUN NUMBER 1 RUN TIME = 12:53:57
*
```

```
NAME C$MOD$
CJ 0.000
CJSW 0.000
DEFW 1.00E-06
NARROW 0.000
```

```
* Level 2 MOSFET MODEL PARAMETERS *
* TEMPERATURE = 27.000 *
```


* RUN DATE = 1/21/92 RUN NUMBER 1 RUN TIME = 12:53:57

*

NAME	PCH	NCH
TYPE	PMOS	NMOS
LEVEL	2.000	2.000
VTO	-0.788	0.792
KP	2.00E-05	2.00E-05
GAMMA	0.000	0.000
PHI	0.600	0.600
LAMBDA	0.000	0.000
RD	0.000	0.000
RS	0.000	0.000
CBD	0.000	0.000
CBS	0.000	0.000
IS	1.00E-16	1.00E-16
PB	1.180	0.759
CGSO	1.12E-10	1.99E-10
CGDO	1.12E-10	1.99E-10
CGBO	3.32E-10	4.98E-10
RSH	80.410	27.020
CJ	2.79E-04	2.58E-05
MJ	0.416	0.192
CJSW	4.59E-11	1.17E-10
MJSW	0.999	0.999
JS	1.00E-04	1.00E-04
TOX	4.16E-08	4.16E-08
NSUB	8.46E+15	3.63E+15
NSS	0.000	0.000
NFS	3.07E+11	5.43E+11
TPG	-1.000	1.000
XJ	5.83E-07	4.75E-07
LD	2.40E-07	1.35E-07
UO	237.800	619.000
UCRIT	1.14E+05	1.50E+05
UEXP	0.309	0.199
VMAX	3.30E+04	4.49E+04
NEFF	2.763	5.491
KF	0.000	0.000
AF	1.000	1.000
FC	0.500	0.500
DELTA	0.001	1.731
TNOM	27.000	27.000

* RESISTOR MODEL PARAMETERS *
* TEMPERATURE = 27.000 *
* RUN DATE = 1/21/92 RUN NUMBER 1 RUN TIME = 12:53:57


```

NAME  R$MOD$
TC1   0.000
TC2   0.000
RSH   0.000
DEFW  1.00E-06
NARROW 0.000

```

```

*****
*****
*                CIRCUIT ELEMENT SUMMARY                *
*                TEMPERATURE = 27.000                    *
* RUN DATE = 1/21/92      RUN NUMBER 1      RUN TIME = 12:53:57
*
*****
*****

```

```

=====
=====
=====
=                PARAMETERS                =
=====
=====
=====

```

NAME	VALUE	TC1	TC2
PL	1.00E+00	0.00E+00	0.00E+00
PC	1.00E+00	0.00E+00	0.00E+00
PR	1.00E+00	0.00E+00	0.00E+00

```

=====
=====
=====
=                INDEPENDENT VOLTAGE SOURCES                =
=====
=====
=====

```

NAME	N+	N-	DC VALUE	AC VALUE	AC PHASE
TRANSIENT					
VMINUS	7	4	0.000	0.000	0.000
VPLUS	6	0	-4.25E-04	1.000	0.000
VSS	2	0	-5.000	0.000	0.000
VDD	1	0	5.000	0.000	0.000

```

=====
=====
=====
=                CAPACITORS                =
=====
=====
=====

```

NAME	N+	N-	PARAMETER	IC	VALUE
CPDIFF	8	0	PC	0.000	2.05E-13
CNDIFF	5	0	PC	0.000	1.77E-14
CNCOMP	11	4	PC	0.000	3.00E-13
CPCOMP	13	4	PC	0.000	3.00E-13
COUT	4	0	PC	0.000	2.00E-12

```

=====
=====
=====
=
Level 2 MOSFETS
=====
=====
=====
=====

```

NAME	MPRDIFF	MPLDIFF	MPROUT	MPRLOAD	MPLLOAD	MNRROUT
------	---------	---------	--------	---------	---------	---------

DRAIN	9	13	4	11	3	4
GATE	7	6	11	3	3	13
SOURCE	8	8	1	1	1	2
BULK	1	1	1	1	1	2
MODEL	PCH	PCH	PCH	PCH	PCH	NCH
W	3.30E-04	3.75E-04	3.97E-04	3.80E-05	3.80E-05	7.00E-05
L	1.50E-05	1.50E-05	1.50E-05	1.50E-05	1.50E-05	1.00E-05
CGS	0.000	0.000	0.000	0.000	0.000	0.000
CGD	0.000	0.000	0.000	0.000	0.000	0.000
CGB	0.000	0.000	0.000	0.000	0.000	0.000
AD	9.90E-10	1.13E-09	1.13E-09	1.14E-10	1.14E-10	2.10E-10
AS	9.90E-10	1.13E-09	1.13E-09	1.14E-10	1.14E-10	2.10E-10
PD	3.36E-04	3.81E-04	3.81E-04	4.40E-05	4.40E-05	7.60E-05
PS	3.36E-04	3.81E-04	3.81E-04	4.40E-05	4.40E-05	7.60E-05
NRD	80.410	80.410	80.410	80.410	80.410	27.020
NRS	80.410	80.410	80.410	80.410	80.410	27.020
ICVDS	0.000	0.000	0.000	0.000	0.000	0.000
ICVGS	0.000	0.000	0.000	0.000	0.000	0.000
ICVBS	0.000	0.000	0.000	0.000	0.000	0.000
ON/OFF	ON	ON	ON	ON	ON	ON

NAME	MNRLOAD	MNLLOAD	MNRDIFF	MNLDIFF
DRAIN	13	9	3	11
GATE	9	9	7	6
SOURCE	2	2	5	5
BULK	2	2	2	2
MODEL	NCH	NCH	NCH	NCH
W	1.00E-05	1.00E-05	8.80E-05	1.00E-04
L	1.00E-05	1.00E-05	1.00E-05	1.00E-05
CGS	0.000	0.000	0.000	0.000
CGD	0.000	0.000	0.000	0.000
CGB	0.000	0.000	0.000	0.000
AD	3.00E-11	3.00E-11	2.64E-10	3.00E-10
AS	3.00E-11	3.00E-11	2.64E-10	3.00E-10
PD	1.60E-05	1.60E-05	9.40E-05	1.06E-04
PS	1.60E-05	1.60E-05	9.40E-05	1.06E-04
NRD	27.020	27.020	27.020	27.020

```

NRS      27.020  27.020  27.020  27.020
ICVDS    0.000   0.000   0.000   0.000
ICVGS    0.000   0.000   0.000   0.000
ICVBS    0.000   0.000   0.000   0.000
ON/OFF   ON      ON      ON      ON
  
```

```

=====
=====
=====
=
RESISTORS
=====
=====
=====
  
```

NAME	N+	N-	PARAMETER	VALUE	TC1	TC2
ROUT	4	0	PR	1.00E+11	0.000	0.000

```

=====
=====
=====
=
INDEPENDENT CURRENT SOURCES
=====
=====
=====
  
```

NAME	N+	N-	DC VALUE	AC VALUE	AC PHASE
TRANSIENT					
IPDIFF	8	1	-1.02E-05	0.000	0.000
INDIFF	5	2	9.93E-06	0.000	0.000

```

*****
*****
*
*           ELEMENT NODE TABLE           *
*           TEMPERATURE = 27.000         *
* RUN DATE = 1/21/92      RUN NUMBER 1    RUN TIME = 12:53:57
*
*****
*****
  
```

```

0  CNDIFF  COUT  CPDIFF  ROUT  VDD  VPLUS
   VSS
1  IPDIFF  MPLDIFF  MPLLOAD  MPLLOAD  MPRDIFF  MPRLOAD
   MPRLOAD  MPROUT  MPROUT  VDD
2  INDIFF  MNLDIFF  MNLLOAD  MNLLOAD  MNRDIFF  MNRLOAD
   MNRLOAD  MNROUT  MNROUT  VSS
3  MNRDIFF  MPLLOAD  MPLLOAD  MPRLOAD
4  CNCOMP  COUT  CPCOMP  MNROUT  MPROUT  ROUT
  
```

VMINUS

5 CNDIFF INDIFF MNLDIFF MNRDIFF
 6 MNLDIFF MPLDIFF VPLUS
 7 MNRDIFF MPRDIFF VMINUS
 8 CPDIFF IPDIFF MPLDIFF MPRDIFF
 9 MNLLOAD MNLLOAD MNRLOAD MPRDIFF
 11 CNCOMP MNLDIFF MPRLOAD MPROUT
 13 CPCOMP MNRLOAD MNROUT MPLDIFF

```
*****
*****
*
*           OPTION SUMMARY
*           TEMPERATURE = 27.000
* RUN DATE = 1/21/92      RUN NUMBER 1      RUN TIME = 12:53:57
*
*****
*****
```

DC ANALYSIS -

GMIN = 1.00E-12
 TOL = 1.00E-05
 ABSTOL = 1.00E-12
 LVLCOD = 1
 ITL1 = 100000
 ITL2 = 100000
 DCRTOL = 1.00E-06
 DCATOL = 1.00E-12
 DCVTOL = 1.00E-06
 CKDC = 2.50E+00
 CKBJT = 2.50E+01
 ITLBJT = 300

TRANSIENT ANALYSIS -

METHOD = GEAR
 MAXORD = 2
 TSATOL = 1.00E-03
 LVLTIM = 4
 ITL3 = 4
 ITL4 = 2000
 ITL5 = 50000
 TRATOL = 1.00E-12

MISCELLANEOUS -

LIMPTS = 10001
 LIMITIM = 5.00E+00
 ANTIME = 0.00E+00
 NUMDGT = 4
 TNOM = 2.70E+01
 CAPMAX = 1.00E-03

MINOUT = 50
 RMIN = 1.00E-14
 LVLSUB = 0

MACRO -
 VDIODE = -1.00E+00
 VMAX = 5.00E+00
 VBB = 0.00E+00

* SMALL SIGNAL BIAS SOLUTION *
 * TEMPERATURE = 27.000 *
 * RUN DATE = 1/21/92 RUN NUMBER 1 RUN TIME = 12:53:57
 *

NODE	VOLTAGE	NODE	VOLTAGE	NODE	VOLTAGE	NODE	VOLTAGE
1	5.000	2	-5.000	3	3.722	4	6.18E-08
5	-1.402	6	-4.25E-04	7	6.18E-08	8	1.644
9	-3.733	11	0.452	13	-0.235		

VOLTAGE SOURCE CURRENTS

NAME	CURRENT
VPLUS	0.000
VSS	2.13E-03
VDD	-2.13E-03
VMINUS	-1.55E-17

TOTAL POWER DISSIPATION 0.021 WATTS

* OPERATING POINT INFORMATION *
 * TEMPERATURE = 27.000 *
 * RUN DATE = 1/21/92 RUN NUMBER 1 RUN TIME = 12:53:57
 *

=====
 =====
 =====
 =====
 = MOSFET-Level 2 model =
 =====
 =====
 =====

NAME	MPRDIFF	MPLDIFF	MPROUT	MPRLOAD	MPLLOAD	MNROUT	MNRLOAD
MODEL	PCH	PCH	PCH	PCH	PCH	NCH	NCH
ID	-4.97E-06	-5.23E-06	-2.11E-03	-5.05E-06	-4.88E-06	2.11E-03	
	5.23E-06						
VGS	-1.644	-1.644	-4.548	-1.278	-1.278	4.765	1.267
VDS	-5.377	-1.879	-5.000	-4.548	-1.278	5.000	4.765
VBS	3.356	3.356	0.000	0.000	0.000	0.000	0.000
VT	-1.531	-1.536	-0.881	-0.823	-0.827	0.847	0.843
GM	6.34E-05	6.87E-05	1.01E-03	2.02E-05	1.96E-05	9.30E-04	
	2.18E-05						
GDS	9.74E-08	1.51E-07	1.40E-05	3.80E-08	7.43E-08	2.08E-05	
	5.48E-08						
GMBS	9.32E-06	1.01E-05	1.90E-04	6.63E-06	6.48E-06	1.14E-04	
	4.98E-06						
CBD(D)	1.16E-13	1.58E-13	1.63E-13	1.69E-14	2.44E-14	4.87E-15	
	7.88E-16						
CBS(D)	1.62E-13	1.84E-13	3.12E-13	3.38E-14	3.38E-14	1.36E-14	
	2.65E-15						
CGS(V)	2.65E-12	3.01E-12	3.19E-12	3.05E-13	3.05E-13	3.77E-13	
	5.38E-14						
CGD(V)	0.000	0.000	0.000	0.000	0.000	0.000	
	4.85E-15						
CGB(V)	4.82E-15	4.82E-15	4.82E-15	4.82E-15	4.82E-15	4.85E-15	
	4.85E-15						

NAME	MNLLOAD	MNRDIFF	MNLDIFF
MODEL	NCH	NCH	NCH
ID	4.97E-06	4.88E-06	5.05E-06
VGS	1.267	1.402	1.401
VDS	1.267	5.124	1.853
VBS	0.000	-3.598	-3.598
VT	0.847	1.309	1.314
GM	2.08E-05	6.63E-05	7.09E-05
GDS	1.11E-07	1.27E-07	1.93E-07
GMBS	4.81E-06	5.98E-06	6.39E-06
CBD(D)	1.35E-15	5.08E-15	6.70E-15
CBS(D)	2.65E-15	6.80E-15	7.71E-15
CGS(V)	5.38E-14	4.74E-13	5.38E-13
CGD(V)	0.000	0.000	0.000
CGB(V)	4.85E-15	4.85E-15	4.85E-15

```

=====
=====
=====
=
=
RESISTORS
=
=====
=====
=====
=====
=====

```

NAME	ROUT
VOLTAGE	6.18E-08
CURRENT	6.18E-19
POWER	3.82E-26

```

*****
*****

```

```

*           NODE CAPACITANCE TABLE           *
*           TEMPERATURE = 27.000             *
* RUN DATE = 1/21/92      RUN NUMBER 1      RUN TIME = 12:53:57
*

```

```

*****
*****

```

NODE MOSFET AREA	TOTAL CAPACITANCE	NODE MOSFET AREA	JUNCTION VOLTAGE	SIDEWALL MOSFET AREA
0	2.22E-12	0.000	0.000	0.000
1	4.70E-12	1.90E-03	5.000	5.58E-09
2	5.60E-13	5.08E-04	-5.000	1.40E-09
3	6.80E-13	1.38E-04	3.722	3.78E-10
4	2.83E-12	4.57E-04	6.18E-08	1.33E-09
5	1.08E-12	2.00E-04	-1.402	5.64E-10
6	3.69E-12	0.000	-4.25E-04	0.000
7	3.24E-12	0.000	6.18E-08	0.000
8	6.29E-12	7.17E-04	1.644	2.11E-09
9	2.77E-13	3.52E-04	-3.733	1.02E-09
11	3.63E-12	1.50E-04	0.452	4.14E-10
13	9.13E-13	3.97E-04	-0.235	1.15E-09

```

*****
*****

```

```

*           AC ANALYSIS           *
*           TEMPERATURE = 27.000             *
* RUN DATE = 1/21/92      RUN NUMBER 1      RUN TIME = 12:53:57
*

```

```

*****
*****

```

```

*****
*****

```

```

*           JOB STATISTICS SUMMARY           *
*****
*****

```

```

Global Signals = 0
Subckt Signals = 0
Total Signals = 0

Number Equations = 36
Number Non-Zeros (Before) = 186
Number Non-Zeros (After) = 215
Percent Zeros = 83.41
Number Matrix Operations = 419
LVL COD = 1

RERUNS = 0
TEMPERATURES = 1

Number Resistors = 1
Number Capacitors = 5
Number Inductors = 0
Number Voltage Sources = 4

```


Number Current Sources = 2
 Number Dependent Sources = 0
 Number Diodes = 0
 Number BJTs = 0
 Number JFETs = 0
 Number MOSFETs = 10
 Total number of elements = 22

 Number Diode models = 0
 Number BJT models = 0
 Number JFET models = 0
 Number MOS models = 2

DC SOLUTION ANALYSIS:

Number of Iterations = 46
 Load Time = 0.00 SEC
 Solve Time = 0.00 SEC
 Analysis Time = 0.00 SEC

AC ANALYSIS:

Number of points = 121
 Analysis Time = 1.45 SEC

MEMORY USAGE:

Memory used for Devices = 0.02 MB
 Memory used for Matrix = 0.02 MB
 Memory used for RHS = 0.02 MB
 Total Memory used in ANALYSIS = 0.11 MB
 Total Memory used in OUTPUT = 0.03 MB

RUN (CPU) TIMES:

	user +	sys	
Topology Check =	0.00		SEC
Matrix Setup =	0.02		SEC
Re-order =	0.05		SEC
READIN =	0.40 +	0.22 =	0.62 SEC
SETUP =	0.10 +	0.20 =	0.30 SEC
ANALYSIS =	2.23 +	0.33 =	2.57 SEC
OUTPUT =	0.38 +	0.07 =	0.45 SEC
TOTAL =	3.25 +	1.15 =	4.40 SEC

0 Error(s).
 0 Warning(s).

JOB CONCLUDED

APPENDIX C

EXAMPLE SPICE INPUT DECK FOR THE 'PRECISE' MACRO MODEL

```

* Comparator design
* begun on computer 6/25/91
*
*****
* Analysis Section
*****
.WIDTH IN = 80 OUT = 80
*.DC VPLUS -.01 .01 .25M
*.PUNCH DC I(VOUT)
*.PRINT DC I(VOUT)
.AC DEC 10 100 1E14
.PUNCH AC VDB(4) VP(4)
*.PZ 6 0 4 0 VOL
*.FORCE NODE 4 5.0
*.TRAN .1N 1U
*.PUNCH TR V(4)
*.INITIAL TR NODE 4 -5.0
*.TF V(4) VPLUS
*.OP
.options itl1 = 100000 itl2 = 100000 ITL4 = 2000 RMIN = 1E-14 ITLPZ = 10000
+ PZMAXR = 5E11
*****
* Power Rails and Inputs
*****
* -.425M offset voltage
VPLUS 6 0 DC -.425M AC 1
VMINUS 7 0 DC 0
XCOMP 6 7 4 COMP
*
*VP1 1 0 5
*VP2 2 0 -5
*VP3 3 0 -5
*VP5 5 0 -5
*VP6 6 0 4.0
*XCOMP1 1 4 4 COMP
*XCOMP2 2 4 4 COMP
*XCOMP3 3 4 4 COMP
*XCOMP5 5 4 4 COMP
*XCOMP6 6 4 4 COMP
*COU 4 0 10P

*****
* macro modeling section
*****
.SUBCKT POLE 1 2 POLEVAL = 1E3
CPOLE 2 0 1
EPOLE 3 0 1 0 1
RPOLE 2 3 #1/POLEVAL#
.ENDS POLE

.SUBCKT PZERO 4 3 ZEROVAL = 1E3
EIN 1 0 0 4 #1/ZEROVAL#
VIN 1 5 0

```

```

CIN 5 0 1
HIN 2 0 VIN -1
EOUT 2 3 4 0 1
COUT 3 0 1
.ENDS PZERO

```

```

.SUBCKT NZERO 4 3 ZEROVAL = 1E3
EIN 1 0 0 4 #1/ZEROVAL#
VIN 1 5 0
CIN 5 0 1
HIN 2 0 VIN 1
EOUT 2 3 4 0 -1
COUT 3 0 1
.ENDS NZERO

```

```

.SUBCKT PATHAC 1 26
CIN 1 0 1P
XP1 1 2 POLE POLEVAL = 1.03299E + 04
XP2 2 3 POLE POLEVAL = 1.05876E + 05
XP3 3 4 POLE POLEVAL = 1848G
XP4 4 5 POLE POLEVAL = 7.56800E + 07
XP5 5 6 POLE POLEVAL = 1.38327E + 08
XP6 6 7 POLE POLEVAL = 1.65659E + 08
XP7 7 8 POLE POLEVAL = 4.12584E + 09
XP8 8 9 POLE POLEVAL = 2.01239E + 10
XP9 9 10 POLE POLEVAL = 6.46340E + 10
XP10 10 11 POLE POLEVAL = 6.99790E + 10
XP11 11 12 POLE POLEVAL = 2.46290E + 11
*XP12 12 27 POLE POLEVAL = 6.36000E + 08
*
XZ1 12 14 PZERO ZEROVAL = 7.46358E + 08
XZ2 14 15 PZERO ZEROVAL = 2.65595E + 09
XZ3 15 16 NZERO ZEROVAL = -1.00651E + 05
XZ4 16 17 NZERO ZEROVAL = -3.15066E + 07
XZ5 17 18 NZERO ZEROVAL = -1.27324E + 08
XZ6 18 19 NZERO ZEROVAL = -1.56042E + 08
XZ7 19 20 NZERO ZEROVAL = -4.79228E + 09
XZ8 20 21 NZERO ZEROVAL = -1.38885E + 10
XZ9 21 22 NZERO ZEROVAL = -6.20459E + 10
XZ10 22 23 NZERO ZEROVAL = -8.89957E + 10
XZ11 23 24 NZERO ZEROVAL = -4.20526E + 11
EGAIN 26 0 24 0 1
ROUT 26 0 1MEG
.ENDS PATHAC

```

```

.SUBCKT PATHDC 1 2
GOUT 0 2 PWL(2) 1 0 2 0 IOUT
.LIBRARY DSN = /user/jwagnon/thesis/spice/input/compout.pwl NOPRINT
.ENDS PATHDC

```

* input capacitance is 30picoFarads

```

.SUBCKT COMP 1 2 5
CINPLUS 1 0 26.2P
CINMINUS 2 0 3.02P

```

EIN 3 0 1 2 1
XAC 3 4 PATHAC
XDC 4 5 PATHDC
COUT 5 0 9.3P
ROUT 5 0 100G
.ENDS COMP

.END

APPENDIX D

EXAMPLE SPICE INPUT DECK FOR THE SIMPLIFIED MACRO MODEL

```

* Comparator design
* begun on computer 6/25/91
*
*****
* Analysis Section
*****
.WIDTH IN = 80 OUT = 80
*.DC VPLUS -.1 .1 .25M
*.PUNCH DC I(VOUT)
*.PRINT DC I(VOUT)
*.AC DEC 10 100 1E14
*.PUNCH AC VDB(4) VP(4)
*.PZ 6 0 4 0 VOL
.FORCE NODE 4 5.0 XCOMP1.4 0 XCOMP2.4 0 XCOMP3.4 0 XCOMP6.4 0
XCOMP5.4 0
+ XCOMP1.3 0 XCOMP2.3 0 XCOMP3.3 0 XCOMP6.3 0 XCOMP5.3 0
*.FORCE 4 -5.0
.TRAN .01N 1M
.PUNCH TR V(4)
*.INITIAL TR NODE 4 -5.0 7 -5.0
*.TF V(4) VPLUS
*.OP
.options itl1 = 100000 itl2 = 100000 ITL4 = 2000 RMIN = 1E-14 ITLPZ = 10000
+ PZMAXR = 5E11
*****
* Power Rails and Inputs
*****
* -.425M offset voltage
*.VPLUS 6 0 DC 0 AC 1
*.VMINUS 7 4 DC 0
*.XCOMP 6 7 4 COMP
*.VOUT 4 0 0
*.Rf 7 4 .1

*XZERON 6 5 NZERO ZEROVAL = -6.28E + 05
*XPOLE1 5 3 POLE POLEVAL = 6.28E + 03
*XPOLE2 3 4 POLE POLEVAL = 6.28E + 07

VP1 1 0 -2
VP2 2 0 -2
VP3 3 0 -2
VP5 5 0 -2
VP6 6 0 -1
VP7 7 0 2
VP8 8 0 2.3
VP9 9 0 3.1
VP10 10 0 4
XCOMP1 1 4 4 COMP
XCOMP2 2 4 4 COMP
XCOMP3 3 4 4 COMP
XCOMP5 5 4 4 COMP
XCOMP6 6 4 4 COMP
XCOMP7 7 4 4 COMP
XCOMP8 8 4 4 COMP
XCOMP9 9 4 4 COMP

```

XCOMP10 10 4 4 COMP

* macro modeling section

.SUBCKT POLE 1 2 POLEVAL = 1E3

CPOLE 2 0 1

EPOLE 3 0 1 0 1

RPOLE 2 3 #1/POLEVAL#

.ENDS POLE

.SUBCKT PZERO 4 3 ZEROVAL = 1E3

EIN 1 0 0 4 #1/ZEROVAL#

VIN 1 5 0

CIN 5 0 1

HIN 2 0 VIN -1

EOUT 2 3 4 0 1

COU 3 0 1

.ENDS PZERO

.SUBCKT NZERO 4 3 ZEROVAL = 1E3

EIN 1 0 0 4 #1/ZEROVAL#

VIN 1 5 0

CIN 5 0 1

HIN 2 0 VIN 1

EOUT 2 3 4 0 -1

COU 3 0 1

.ENDS NZERO

.SUBCKT PATHAC 1 26

CIN 1 0 1P

XP1 1 2 POLE POLEVAL = 1.03067E + 04

XP3 2 4 POLE POLEVAL = 2.90000E + 09

EGAIN 26 0 4 0 1

ROUT 26 0 1MEG

.ENDS PATHAC

.SUBCKT PATHDC 1 2

GOUT 0 2 PWL(2) 1 0 2 0 IOU

.LIBRARY DSN = /user/jwagnon/thesis/spice/input/compout.pwl NOPRINT

.ENDS PATHDC

* input capacitance is 30picoFarads

.SUBCKT COMP 1 2 5

CINPLUS 1 0 26.2P

CINMINUS 2 0 3.02P

EIN 3 0 1 2 1

XAC 3 4 PATHAC

XDC 4 5 PATHDC

COU 5 0 2.827P

ROUT 5 0 100G

.ENDS COMP

.END

APPENDIX E

SPICE LIBRARY DECK FOR THE PIECE-WISE LINEAR
DC OUTPUT CURRENT MODEL

* Comparator output current model

.MODEL IOUT PWL(2) SYMMETRY NO SOURCE CARDS DATA

+ 4.354725E-11 -5.000000E+00 -5.0
+ 4.459009E-11 -5.000000E-02 -5.0 4.464315E-11 -4.900000E-02 -5.0
+ 4.470261E-11 -4.800000E-02 -5.0 4.476646E-11 -4.700000E-02 -5.0
+ 4.483519E-11 -4.600000E-02 -5.0 4.490928E-11 -4.500000E-02 -5.0
+ 4.498926E-11 -4.400000E-02 -5.0 4.507572E-11 -4.300000E-02 -5.0
+ 4.516936E-11 -4.200000E-02 -5.0 4.527118E-11 -4.100000E-02 -5.0
+ 4.538191E-11 -4.000000E-02 -5.0 4.550263E-11 -3.900000E-02 -5.0
+ 4.563467E-11 -3.800000E-02 -5.0 4.577940E-11 -3.700000E-02 -5.0
+ 4.593848E-11 -3.600000E-02 -5.0 4.611389E-11 -3.500000E-02 -5.0
+ 4.630788E-11 -3.400000E-02 -5.0 4.652322E-11 -3.300000E-02 -5.0
+ 4.676305E-11 -3.200000E-02 -5.0 4.703134E-11 -3.100000E-02 -5.0
+ 4.733279E-11 -3.000000E-02 -5.0 4.767315E-11 -2.900000E-02 -5.0
+ 4.805954E-11 -2.800000E-02 -5.0 4.850080E-11 -2.700000E-02 -5.0
+ 4.900815E-11 -2.600000E-02 -5.0 4.959589E-11 -2.500000E-02 -5.0
+ 5.028261E-11 -2.400000E-02 -5.0 5.109313E-11 -2.300000E-02 -5.0
+ 5.206088E-11 -2.200000E-02 -5.0 5.323235E-11 -2.100000E-02 -5.0
+ 5.467416E-11 -2.000000E-02 -5.0 5.648553E-11 -1.900000E-02 -5.0
+ 5.882173E-11 -1.800000E-02 -5.0 6.194156E-11 -1.700000E-02 -5.0
+ 6.631704E-11 -1.600000E-02 -5.0 7.293220E-11 -1.500000E-02 -5.0
+ 8.435360E-11 -1.400000E-02 -5.0 1.114440E-10 -1.300000E-02 -5.0
+ 5.904042E-08 -1.200000E-02 -5.0 1.298633E-05 -1.100000E-02 -5.0
+ 6.904875E-05 -1.000000E-02 -5.0 1.728371E-04 -9.000000E-03 -5.0
+ 3.250595E-04 -8.000000E-03 -5.0 5.167490E-04 -7.000000E-03 -5.0
+ 7.158951E-04 -6.000000E-03 -5.0 9.380506E-04 -5.000000E-03 -5.0
+ 1.179743E-03 -4.000000E-03 -5.0 1.437597E-03 -3.000000E-03 -5.0
+ 1.708335E-03 -2.000000E-03 -5.0 1.988749E-03 -1.000000E-03 -5.0
+ 2.275661E-03 3.469447E-17 -5.0 2.565836E-03 1.000000E-03 -5.0
+ 2.855831E-03 2.000000E-03 -5.0 3.141697E-03 3.000000E-03 -5.0
+ 3.418279E-03 4.000000E-03 -5.0 3.677539E-03 5.000000E-03 -5.0
+ 3.901880E-03 6.000000E-03 -5.0 4.019950E-03 7.000000E-03 -5.0
+ 4.021127E-03 8.000000E-03 -5.0 4.027023E-03 9.000000E-03 -5.0
+ 4.031565E-03 1.000000E-02 -5.0 4.035337E-03 1.100000E-02 -5.0
+ 4.038600E-03 1.200000E-02 -5.0 4.041488E-03 1.300000E-02 -5.0
+ 4.044092E-03 1.400000E-02 -5.0 4.046472E-03 1.500000E-02 -5.0
+ 4.048657E-03 1.600000E-02 -5.0 4.050687E-03 1.700000E-02 -5.0
+ 4.052585E-03 1.800000E-02 -5.0 4.054369E-03 1.900000E-02 -5.0
+ 4.056052E-03 2.000000E-02 -5.0 4.057645E-03 2.100000E-02 -5.0
+ 4.059159E-03 2.200000E-02 -5.0 4.060601E-03 2.300000E-02 -5.0
+ 4.061977E-03 2.400000E-02 -5.0 4.063295E-03 2.500000E-02 -5.0
+ 4.064559E-03 2.600000E-02 -5.0 4.065772E-03 2.700000E-02 -5.0
+ 4.066939E-03 2.800000E-02 -5.0 4.068065E-03 2.900000E-02 -5.0
+ 4.069150E-03 3.000000E-02 -5.0 4.070198E-03 3.100000E-02 -5.0
+ 4.071212E-03 3.200000E-02 -5.0 4.072194E-03 3.300000E-02 -5.0
+ 4.073145E-03 3.400000E-02 -5.0 4.074067E-03 3.500000E-02 -5.0
+ 4.074963E-03 3.600000E-02 -5.0 4.075833E-03 3.700000E-02 -5.0
+ 4.076679E-03 3.800000E-02 -5.0 4.077503E-03 3.900000E-02 -5.0
+ 4.078304E-03 4.000000E-02 -5.0 4.079085E-03 4.100000E-02 -5.0
+ 4.079846E-03 4.200000E-02 -5.0 4.080588E-03 4.300000E-02 -5.0
+ 4.081313E-03 4.400000E-02 -5.0 4.082020E-03 4.500000E-02 -5.0
+ 4.082711E-03 4.600000E-02 -5.0 4.083386E-03 4.700000E-02 -5.0
+ 4.084046E-03 4.800000E-02 -5.0 4.084692E-03 4.900000E-02 -5.0
+ 4.157902E-03 5.000000E+00 -5.0

+ -1.432149E-03 -5.000000E+00 -4.0
+ -1.419437E-03 -5.000000E-02 -4.0 -1.419366E-03 -4.900000E-02 -4.0
+ -1.419255E-03 -4.800000E-02 -4.0 -1.419142E-03 -4.700000E-02 -4.0
+ -1.419026E-03 -4.600000E-02 -4.0 -1.418908E-03 -4.500000E-02 -4.0
+ -1.418787E-03 -4.400000E-02 -4.0 -1.418663E-03 -4.300000E-02 -4.0
+ -1.418536E-03 -4.200000E-02 -4.0 -1.418406E-03 -4.100000E-02 -4.0
+ -1.418273E-03 -4.000000E-02 -4.0 -1.418136E-03 -3.900000E-02 -4.0
+ -1.417996E-03 -3.800000E-02 -4.0 -1.417852E-03 -3.700000E-02 -4.0
+ -1.417704E-03 -3.600000E-02 -4.0 -1.417551E-03 -3.500000E-02 -4.0
+ -1.417394E-03 -3.400000E-02 -4.0 -1.417232E-03 -3.300000E-02 -4.0
+ -1.417065E-03 -3.200000E-02 -4.0 -1.416893E-03 -3.100000E-02 -4.0
+ -1.416715E-03 -3.000000E-02 -4.0 -1.416531E-03 -2.900000E-02 -4.0
+ -1.416340E-03 -2.800000E-02 -4.0 -1.416143E-03 -2.700000E-02 -4.0
+ -1.415937E-03 -2.600000E-02 -4.0 -1.415724E-03 -2.500000E-02 -4.0
+ -1.415501E-03 -2.400000E-02 -4.0 -1.415269E-03 -2.300000E-02 -4.0
+ -1.415026E-03 -2.200000E-02 -4.0 -1.414771E-03 -2.100000E-02 -4.0
+ -1.414504E-03 -2.000000E-02 -4.0 -1.414221E-03 -1.900000E-02 -4.0
+ -1.413922E-03 -1.800000E-02 -4.0 -1.413605E-03 -1.700000E-02 -4.0
+ -1.413266E-03 -1.600000E-02 -4.0 -1.412903E-03 -1.500000E-02 -4.0
+ -1.412510E-03 -1.400000E-02 -4.0 -1.412082E-03 -1.300000E-02 -4.0
+ -1.411367E-03 -1.200000E-02 -4.0 -1.397973E-03 -1.100000E-02 -4.0
+ -1.341503E-03 -1.000000E-02 -4.0 -1.237321E-03 -9.000000E-03 -4.0
+ -1.084624E-03 -8.000000E-03 -4.0 -8.919681E-04 -7.000000E-03 -4.0
+ -6.672865E-04 -6.000000E-03 -4.0 -3.946295E-04 -5.000000E-03 -4.0
+ -9.489473E-05 -4.000000E-03 -4.0 2.261977E-04 -3.000000E-03 -4.0
+ 5.647468E-04 -2.000000E-03 -4.0 9.172898E-04 -1.000000E-03 -4.0
+ 1.280558E-03 3.469447E-17 -4.0 1.651320E-03 1.000000E-03 -4.0
+ 2.026202E-03 2.000000E-03 -4.0 2.401381E-03 3.000000E-03 -4.0
+ 2.771905E-03 4.000000E-03 -4.0 3.130049E-03 5.000000E-03 -4.0
+ 3.466760E-03 6.000000E-03 -4.0 3.707157E-03 7.000000E-03 -4.0
+ 3.828067E-03 8.000000E-03 -4.0 3.924955E-03 9.000000E-03 -4.0
+ 3.986571E-03 1.000000E-02 -4.0 4.016890E-03 1.100000E-02 -4.0
+ 4.023770E-03 1.200000E-02 -4.0 4.026657E-03 1.300000E-02 -4.0
+ 4.029250E-03 1.400000E-02 -4.0 4.031619E-03 1.500000E-02 -4.0
+ 4.033795E-03 1.600000E-02 -4.0 4.035816E-03 1.700000E-02 -4.0
+ 4.037706E-03 1.800000E-02 -4.0 4.039482E-03 1.900000E-02 -4.0
+ 4.041157E-03 2.000000E-02 -4.0 4.042744E-03 2.100000E-02 -4.0
+ 4.044251E-03 2.200000E-02 -4.0 4.045687E-03 2.300000E-02 -4.0
+ 4.047057E-03 2.400000E-02 -4.0 4.048369E-03 2.500000E-02 -4.0
+ 4.049627E-03 2.600000E-02 -4.0 4.050835E-03 2.700000E-02 -4.0
+ 4.051998E-03 2.800000E-02 -4.0 4.053118E-03 2.900000E-02 -4.0
+ 4.054198E-03 3.000000E-02 -4.0 4.055242E-03 3.100000E-02 -4.0
+ 4.056251E-03 3.200000E-02 -4.0 4.057229E-03 3.300000E-02 -4.0
+ 4.058176E-03 3.400000E-02 -4.0 4.059094E-03 3.500000E-02 -4.0
+ 4.059986E-03 3.600000E-02 -4.0 4.060852E-03 3.700000E-02 -4.0
+ 4.061695E-03 3.800000E-02 -4.0 4.062514E-03 3.900000E-02 -4.0
+ 4.063312E-03 4.000000E-02 -4.0 4.064090E-03 4.100000E-02 -4.0
+ 4.064847E-03 4.200000E-02 -4.0 4.065586E-03 4.300000E-02 -4.0
+ 4.066308E-03 4.400000E-02 -4.0 4.067012E-03 4.500000E-02 -4.0
+ 4.067700E-03 4.600000E-02 -4.0 4.068372E-03 4.700000E-02 -4.0
+ 4.069029E-03 4.800000E-02 -4.0 4.069672E-03 4.900000E-02 -4.0
+ 4.142560E-03 5.000000E+00 -4.0
+ -2.573592E-03 -5.000000E+00 -3.0
+ -2.547029E-03 -5.000000E-02 -3.0 -2.546872E-03 -4.900000E-02 -3.0

+ -2.546641E-03 -4.800000E-02 -3.0 -2.546404E-03 -4.700000E-02 -3.0
+ -2.546163E-03 -4.600000E-02 -3.0 -2.545916E-03 -4.500000E-02 -3.0
+ -2.545663E-03 -4.400000E-02 -3.0 -2.545405E-03 -4.300000E-02 -3.0
+ -2.545140E-03 -4.200000E-02 -3.0 -2.544868E-03 -4.100000E-02 -3.0
+ -2.544590E-03 -4.000000E-02 -3.0 -2.544304E-03 -3.900000E-02 -3.0
+ -2.544011E-03 -3.800000E-02 -3.0 -2.543709E-03 -3.700000E-02 -3.0
+ -2.543400E-03 -3.600000E-02 -3.0 -2.543081E-03 -3.500000E-02 -3.0
+ -2.542753E-03 -3.400000E-02 -3.0 -2.542415E-03 -3.300000E-02 -3.0
+ -2.542066E-03 -3.200000E-02 -3.0 -2.541706E-03 -3.100000E-02 -3.0
+ -2.541334E-03 -3.000000E-02 -3.0 -2.540949E-03 -2.900000E-02 -3.0
+ -2.540551E-03 -2.800000E-02 -3.0 -2.540138E-03 -2.700000E-02 -3.0
+ -2.539709E-03 -2.600000E-02 -3.0 -2.539262E-03 -2.500000E-02 -3.0
+ -2.538797E-03 -2.400000E-02 -3.0 -2.538311E-03 -2.300000E-02 -3.0
+ -2.537803E-03 -2.200000E-02 -3.0 -2.537271E-03 -2.100000E-02 -3.0
+ -2.536711E-03 -2.000000E-02 -3.0 -2.536120E-03 -1.900000E-02 -3.0
+ -2.535495E-03 -1.800000E-02 -3.0 -2.534831E-03 -1.700000E-02 -3.0
+ -2.534122E-03 -1.600000E-02 -3.0 -2.533362E-03 -1.500000E-02 -3.0
+ -2.532539E-03 -1.400000E-02 -3.0 -2.531643E-03 -1.300000E-02 -3.0
+ -2.530247E-03 -1.200000E-02 -3.0 -2.516344E-03 -1.100000E-02 -3.0
+ -2.459424E-03 -1.000000E-02 -3.0 -2.354803E-03 -9.000000E-03 -3.0
+ -2.201576E-03 -8.000000E-03 -3.0 -2.007853E-03 -7.000000E-03 -3.0
+ -1.755339E-03 -6.000000E-03 -3.0 -1.427570E-03 -5.000000E-03 -3.0
+ -1.064333E-03 -4.000000E-03 -3.0 -6.728582E-04 -3.000000E-03 -3.0
+ -2.594774E-04 -2.000000E-03 -3.0 1.730146E-04 -1.000000E-03 -3.0
+ 6.213611E-04 3.469447E-17 -3.0 1.082476E-03 1.000000E-03 -3.0
+ 1.553248E-03 2.000000E-03 -3.0 2.030246E-03 3.000000E-03 -3.0
+ 2.515231E-03 4.000000E-03 -3.0 2.977901E-03 5.000000E-03 -3.0
+ 3.392113E-03 6.000000E-03 -3.0 3.676455E-03 7.000000E-03 -3.0
+ 3.807797E-03 8.000000E-03 -3.0 3.906842E-03 9.000000E-03 -3.0
+ 3.969537E-03 1.000000E-02 -3.0 4.000355E-03 1.100000E-02 -3.0
+ 4.007343E-03 1.200000E-02 -3.0 4.010215E-03 1.300000E-02 -3.0
+ 4.012796E-03 1.400000E-02 -3.0 4.015153E-03 1.500000E-02 -3.0
+ 4.017317E-03 1.600000E-02 -3.0 4.019328E-03 1.700000E-02 -3.0
+ 4.021208E-03 1.800000E-02 -3.0 4.022975E-03 1.900000E-02 -3.0
+ 4.024642E-03 2.000000E-02 -3.0 4.026220E-03 2.100000E-02 -3.0
+ 4.027719E-03 2.200000E-02 -3.0 4.029147E-03 2.300000E-02 -3.0
+ 4.030511E-03 2.400000E-02 -3.0 4.031816E-03 2.500000E-02 -3.0
+ 4.033067E-03 2.600000E-02 -3.0 4.034270E-03 2.700000E-02 -3.0
+ 4.035426E-03 2.800000E-02 -3.0 4.036540E-03 2.900000E-02 -3.0
+ 4.037615E-03 3.000000E-02 -3.0 4.038653E-03 3.100000E-02 -3.0
+ 4.039658E-03 3.200000E-02 -3.0 4.040630E-03 3.300000E-02 -3.0
+ 4.041572E-03 3.400000E-02 -3.0 4.042486E-03 3.500000E-02 -3.0
+ 4.043373E-03 3.600000E-02 -3.0 4.044235E-03 3.700000E-02 -3.0
+ 4.045072E-03 3.800000E-02 -3.0 4.045888E-03 3.900000E-02 -3.0
+ 4.046682E-03 4.000000E-02 -3.0 4.047455E-03 4.100000E-02 -3.0
+ 4.048209E-03 4.200000E-02 -3.0 4.048944E-03 4.300000E-02 -3.0
+ 4.049662E-03 4.400000E-02 -3.0 4.050362E-03 4.500000E-02 -3.0
+ 4.051046E-03 4.600000E-02 -3.0 4.051716E-03 4.700000E-02 -3.0
+ 4.052369E-03 4.800000E-02 -3.0 4.053009E-03 4.900000E-02 -3.0
+ 4.125516E-03 5.000000E+00 -3.0
+ -3.424143E-03 -5.000000E+00 -2.0
+ -3.382589E-03 -5.000000E-02 -2.0 -3.382330E-03 -4.900000E-02 -2.0
+ -3.381968E-03 -4.800000E-02 -2.0 -3.381599E-03 -4.700000E-02 -2.0
+ -3.381221E-03 -4.600000E-02 -2.0 -3.380835E-03 -4.500000E-02 -2.0

+ -3.380439E-03 -4.400000E-02 -2.0 -3.380034E-03 -4.300000E-02 -2.0
+ -3.379620E-03 -4.200000E-02 -2.0 -3.379195E-03 -4.100000E-02 -2.0
+ -3.378759E-03 -4.000000E-02 -2.0 -3.378312E-03 -3.900000E-02 -2.0
+ -3.377853E-03 -3.800000E-02 -2.0 -3.377381E-03 -3.700000E-02 -2.0
+ -3.376896E-03 -3.600000E-02 -2.0 -3.376398E-03 -3.500000E-02 -2.0
+ -3.375884E-03 -3.400000E-02 -2.0 -3.375355E-03 -3.300000E-02 -2.0
+ -3.374809E-03 -3.200000E-02 -2.0 -3.374246E-03 -3.100000E-02 -2.0
+ -3.373664E-03 -3.000000E-02 -2.0 -3.373061E-03 -2.900000E-02 -2.0
+ -3.372437E-03 -2.800000E-02 -2.0 -3.371790E-03 -2.700000E-02 -2.0
+ -3.371118E-03 -2.600000E-02 -2.0 -3.370419E-03 -2.500000E-02 -2.0
+ -3.369691E-03 -2.400000E-02 -2.0 -3.368930E-03 -2.300000E-02 -2.0
+ -3.368135E-03 -2.200000E-02 -2.0 -3.367300E-03 -2.100000E-02 -2.0
+ -3.366423E-03 -2.000000E-02 -2.0 -3.365498E-03 -1.900000E-02 -2.0
+ -3.364519E-03 -1.800000E-02 -2.0 -3.363479E-03 -1.700000E-02 -2.0
+ -3.362368E-03 -1.600000E-02 -2.0 -3.361175E-03 -1.500000E-02 -2.0
+ -3.359886E-03 -1.400000E-02 -2.0 -3.358480E-03 -1.300000E-02 -2.0
+ -3.356381E-03 -1.200000E-02 -2.0 -3.341926E-03 -1.100000E-02 -2.0
+ -3.284515E-03 -1.000000E-02 -2.0 -3.179413E-03 -9.000000E-03 -2.0
+ -3.025608E-03 -8.000000E-03 -2.0 -2.830721E-03 -7.000000E-03 -2.0
+ -2.548059E-03 -6.000000E-03 -2.0 -2.159204E-03 -5.000000E-03 -2.0
+ -1.726895E-03 -4.000000E-03 -2.0 -1.260541E-03 -3.000000E-03 -2.0
+ -7.653465E-04 -2.000000E-03 -2.0 -2.414221E-04 -1.000000E-03 -2.0
+ 3.229713E-04 3.469447E-17 -2.0 8.877817E-04 1.000000E-03 -2.0
+ 1.441784E-03 2.000000E-03 -2.0 1.976549E-03 3.000000E-03 -2.0
+ 2.483204E-03 4.000000E-03 -2.0 2.950455E-03 5.000000E-03 -2.0
+ 3.367119E-03 6.000000E-03 -2.0 3.653449E-03 7.000000E-03 -2.0
+ 3.786652E-03 8.000000E-03 -2.0 3.886895E-03 9.000000E-03 -2.0
+ 3.950314E-03 1.000000E-02 -2.0 3.981496E-03 1.100000E-02 -2.0
+ 3.988559E-03 1.200000E-02 -2.0 3.991414E-03 1.300000E-02 -2.0
+ 3.993977E-03 1.400000E-02 -2.0 3.996319E-03 1.500000E-02 -2.0
+ 3.998469E-03 1.600000E-02 -2.0 4.000467E-03 1.700000E-02 -2.0
+ 4.002335E-03 1.800000E-02 -2.0 4.004090E-03 1.900000E-02 -2.0
+ 4.005746E-03 2.000000E-02 -2.0 4.007314E-03 2.100000E-02 -2.0
+ 4.008804E-03 2.200000E-02 -2.0 4.010222E-03 2.300000E-02 -2.0
+ 4.011577E-03 2.400000E-02 -2.0 4.012874E-03 2.500000E-02 -2.0
+ 4.014117E-03 2.600000E-02 -2.0 4.015311E-03 2.700000E-02 -2.0
+ 4.016460E-03 2.800000E-02 -2.0 4.017567E-03 2.900000E-02 -2.0
+ 4.018635E-03 3.000000E-02 -2.0 4.019666E-03 3.100000E-02 -2.0
+ 4.020664E-03 3.200000E-02 -2.0 4.021630E-03 3.300000E-02 -2.0
+ 4.022566E-03 3.400000E-02 -2.0 4.023474E-03 3.500000E-02 -2.0
+ 4.024355E-03 3.600000E-02 -2.0 4.025211E-03 3.700000E-02 -2.0
+ 4.026044E-03 3.800000E-02 -2.0 4.026853E-03 3.900000E-02 -2.0
+ 4.027642E-03 4.000000E-02 -2.0 4.028411E-03 4.100000E-02 -2.0
+ 4.029159E-03 4.200000E-02 -2.0 4.029890E-03 4.300000E-02 -2.0
+ 4.030603E-03 4.400000E-02 -2.0 4.031299E-03 4.500000E-02 -2.0
+ 4.031979E-03 4.600000E-02 -2.0 4.032643E-03 4.700000E-02 -2.0
+ 4.033293E-03 4.800000E-02 -2.0 4.033928E-03 4.900000E-02 -2.0
+ 4.105956E-03 5.000000E+00 -2.0
+ -3.968550E-03 -5.000000E+00 -1.0
+ -3.902509E-03 -5.000000E-02 -1.0 -3.901965E-03 -4.900000E-02 -1.0
+ -3.901392E-03 -4.800000E-02 -1.0 -3.900805E-03 -4.700000E-02 -1.0
+ -3.900206E-03 -4.600000E-02 -1.0 -3.899593E-03 -4.500000E-02 -1.0
+ -3.898965E-03 -4.400000E-02 -1.0 -3.898323E-03 -4.300000E-02 -1.0
+ -3.897665E-03 -4.200000E-02 -1.0 -3.896990E-03 -4.100000E-02 -1.0

+ -3.896298E-03 -4.000000E-02 -1.0 -3.895588E-03 -3.900000E-02 -1.0
+ -3.894859E-03 -3.800000E-02 -1.0 -3.894110E-03 -3.700000E-02 -1.0
+ -3.893340E-03 -3.600000E-02 -1.0 -3.892548E-03 -3.500000E-02 -1.0
+ -3.891732E-03 -3.400000E-02 -1.0 -3.890891E-03 -3.300000E-02 -1.0
+ -3.890024E-03 -3.200000E-02 -1.0 -3.889128E-03 -3.100000E-02 -1.0
+ -3.888202E-03 -3.000000E-02 -1.0 -3.887244E-03 -2.900000E-02 -1.0
+ -3.886251E-03 -2.800000E-02 -1.0 -3.885221E-03 -2.700000E-02 -1.0
+ -3.884151E-03 -2.600000E-02 -1.0 -3.883038E-03 -2.500000E-02 -1.0
+ -3.881877E-03 -2.400000E-02 -1.0 -3.880665E-03 -2.300000E-02 -1.0
+ -3.879396E-03 -2.200000E-02 -1.0 -3.878065E-03 -2.100000E-02 -1.0
+ -3.876664E-03 -2.000000E-02 -1.0 -3.875187E-03 -1.900000E-02 -1.0
+ -3.873621E-03 -1.800000E-02 -1.0 -3.871957E-03 -1.700000E-02 -1.0
+ -3.870178E-03 -1.600000E-02 -1.0 -3.868266E-03 -1.500000E-02 -1.0
+ -3.866195E-03 -1.400000E-02 -1.0 -3.863934E-03 -1.300000E-02 -1.0
+ -3.861183E-03 -1.200000E-02 -1.0 -3.845789E-03 -1.100000E-02 -1.0
+ -3.787453E-03 -1.000000E-02 -1.0 -3.681360E-03 -9.000000E-03 -1.0
+ -3.526315E-03 -8.000000E-03 -1.0 -3.329159E-03 -7.000000E-03 -1.0
+ -2.999376E-03 -6.000000E-03 -1.0 -2.519394E-03 -5.000000E-03 -1.0
+ -1.995014E-03 -4.000000E-03 -1.0 -1.443237E-03 -3.000000E-03 -1.0
+ -8.747408E-04 -2.000000E-03 -1.0 -2.979095E-04 -1.000000E-03 -1.0
+ 2.797917E-04 3.469447E-17 -1.0 8.512578E-04 1.000000E-03 -1.0
+ 1.409406E-03 2.000000E-03 -1.0 1.946819E-03 3.000000E-03 -1.0
+ 2.455079E-03 4.000000E-03 -1.0 2.923174E-03 5.000000E-03 -1.0
+ 3.340265E-03 6.000000E-03 -1.0 3.627437E-03 7.000000E-03 -1.0
+ 3.762031E-03 8.000000E-03 -1.0 3.863195E-03 9.000000E-03 -1.0
+ 3.927189E-03 1.000000E-02 -1.0 3.958660E-03 1.100000E-02 -1.0
+ 3.965775E-03 1.200000E-02 -1.0 3.968603E-03 1.300000E-02 -1.0
+ 3.971143E-03 1.400000E-02 -1.0 3.973463E-03 1.500000E-02 -1.0
+ 3.975592E-03 1.600000E-02 -1.0 3.977572E-03 1.700000E-02 -1.0
+ 3.979422E-03 1.800000E-02 -1.0 3.981160E-03 1.900000E-02 -1.0
+ 3.982801E-03 2.000000E-02 -1.0 3.984354E-03 2.100000E-02 -1.0
+ 3.985829E-03 2.200000E-02 -1.0 3.987235E-03 2.300000E-02 -1.0
+ 3.988577E-03 2.400000E-02 -1.0 3.989861E-03 2.500000E-02 -1.0
+ 3.991093E-03 2.600000E-02 -1.0 3.992275E-03 2.700000E-02 -1.0
+ 3.993413E-03 2.800000E-02 -1.0 3.994510E-03 2.900000E-02 -1.0
+ 3.995567E-03 3.000000E-02 -1.0 3.996589E-03 3.100000E-02 -1.0
+ 3.997577E-03 3.200000E-02 -1.0 3.998534E-03 3.300000E-02 -1.0
+ 3.999461E-03 3.400000E-02 -1.0 4.000360E-03 3.500000E-02 -1.0
+ 4.001233E-03 3.600000E-02 -1.0 4.002081E-03 3.700000E-02 -1.0
+ 4.002905E-03 3.800000E-02 -1.0 4.003708E-03 3.900000E-02 -1.0
+ 4.004489E-03 4.000000E-02 -1.0 4.005250E-03 4.100000E-02 -1.0
+ 4.005991E-03 4.200000E-02 -1.0 4.006715E-03 4.300000E-02 -1.0
+ 4.007421E-03 4.400000E-02 -1.0 4.008111E-03 4.500000E-02 -1.0
+ 4.008784E-03 4.600000E-02 -1.0 4.009442E-03 4.700000E-02 -1.0
+ 4.010085E-03 4.800000E-02 -1.0 4.010715E-03 4.900000E-02 -1.0
+ 4.082044E-03 5.000000E+00 -1.0
+ -4.047106E-03 -5.000000E+00 0.0
+ -3.976938E-03 -5.000000E-02 0.0 -3.976358E-03 -4.900000E-02 0.0
+ -3.975745E-03 -4.800000E-02 0.0 -3.975119E-03 -4.700000E-02 0.0
+ -3.974479E-03 -4.600000E-02 0.0 -3.973824E-03 -4.500000E-02 0.0
+ -3.973153E-03 -4.400000E-02 0.0 -3.972467E-03 -4.300000E-02 0.0
+ -3.971764E-03 -4.200000E-02 0.0 -3.971044E-03 -4.100000E-02 0.0
+ -3.970305E-03 -4.000000E-02 0.0 -3.969547E-03 -3.900000E-02 0.0
+ -3.968768E-03 -3.800000E-02 0.0 -3.967969E-03 -3.700000E-02 0.0

+ -3.967147E-03 -3.600000E-02 0.0 -3.966301E-03 -3.500000E-02 0.0
+ -3.965430E-03 -3.400000E-02 0.0 -3.964534E-03 -3.300000E-02 0.0
+ -3.963608E-03 -3.200000E-02 0.0 -3.962652E-03 -3.100000E-02 0.0
+ -3.961665E-03 -3.000000E-02 0.0 -3.960643E-03 -2.900000E-02 0.0
+ -3.959585E-03 -2.800000E-02 0.0 -3.958487E-03 -2.700000E-02 0.0
+ -3.957347E-03 -2.600000E-02 0.0 -3.956160E-03 -2.500000E-02 0.0
+ -3.954924E-03 -2.400000E-02 0.0 -3.953632E-03 -2.300000E-02 0.0
+ -3.952281E-03 -2.200000E-02 0.0 -3.950864E-03 -2.100000E-02 0.0
+ -3.949373E-03 -2.000000E-02 0.0 -3.947800E-03 -1.900000E-02 0.0
+ -3.946134E-03 -1.800000E-02 0.0 -3.944363E-03 -1.700000E-02 0.0
+ -3.942471E-03 -1.600000E-02 0.0 -3.940438E-03 -1.500000E-02 0.0
+ -3.938237E-03 -1.400000E-02 0.0 -3.935833E-03 -1.300000E-02 0.0
+ -3.932909E-03 -1.200000E-02 0.0 -3.917457E-03 -1.100000E-02 0.0
+ -3.859226E-03 -1.000000E-02 0.0 -3.753394E-03 -9.000000E-03 0.0
+ -3.598746E-03 -8.000000E-03 0.0 -3.401938E-03 -7.000000E-03 0.0
+ -3.066112E-03 -6.000000E-03 0.0 -2.576844E-03 -5.000000E-03 0.0
+ -2.045083E-03 -4.000000E-03 0.0 -1.487488E-03 -3.000000E-03 0.0
+ -9.144305E-04 -2.000000E-03 0.0 -3.340862E-04 -1.000000E-03 0.0
+ 2.462142E-04 3.469447E-17 0.0 8.194465E-04 1.000000E-03 0.0
+ 1.378570E-03 2.000000E-03 0.0 1.916183E-03 3.000000E-03 0.0
+ 2.423862E-03 4.000000E-03 0.0 2.890613E-03 5.000000E-03 0.0
+ 3.305849E-03 6.000000E-03 0.0 3.592285E-03 7.000000E-03 0.0
+ 3.728032E-03 8.000000E-03 0.0 3.829909E-03 9.000000E-03 0.0
+ 3.894344E-03 1.000000E-02 0.0 3.926020E-03 1.100000E-02 0.0
+ 3.933139E-03 1.200000E-02 0.0 3.935905E-03 1.300000E-02 0.0
+ 3.938389E-03 1.400000E-02 0.0 3.940659E-03 1.500000E-02 0.0
+ 3.942741E-03 1.600000E-02 0.0 3.944677E-03 1.700000E-02 0.0
+ 3.946486E-03 1.800000E-02 0.0 3.948186E-03 1.900000E-02 0.0
+ 3.949790E-03 2.000000E-02 0.0 3.951308E-03 2.100000E-02 0.0
+ 3.952751E-03 2.200000E-02 0.0 3.954125E-03 2.300000E-02 0.0
+ 3.955437E-03 2.400000E-02 0.0 3.956692E-03 2.500000E-02 0.0
+ 3.957896E-03 2.600000E-02 0.0 3.959052E-03 2.700000E-02 0.0
+ 3.960164E-03 2.800000E-02 0.0 3.961236E-03 2.900000E-02 0.0
+ 3.962270E-03 3.000000E-02 0.0 3.963269E-03 3.100000E-02 0.0
+ 3.964234E-03 3.200000E-02 0.0 3.965169E-03 3.300000E-02 0.0
+ 3.966075E-03 3.400000E-02 0.0 3.966954E-03 3.500000E-02 0.0
+ 3.967807E-03 3.600000E-02 0.0 3.968636E-03 3.700000E-02 0.0
+ 3.969441E-03 3.800000E-02 0.0 3.970225E-03 3.900000E-02 0.0
+ 3.970989E-03 4.000000E-02 0.0 3.971732E-03 4.100000E-02 0.0
+ 3.972457E-03 4.200000E-02 0.0 3.973164E-03 4.300000E-02 0.0
+ 3.973853E-03 4.400000E-02 0.0 3.974527E-03 4.500000E-02 0.0
+ 3.975185E-03 4.600000E-02 0.0 3.975828E-03 4.700000E-02 0.0
+ 3.976456E-03 4.800000E-02 0.0 3.977071E-03 4.900000E-02 0.0
+ 4.046691E-03 5.000000E+00 0.0
+ -4.091755E-03 -5.000000E+00 1.0
+ -4.020398E-03 -5.000000E-02 1.0 -4.019808E-03 -4.900000E-02 1.0
+ -4.019185E-03 -4.800000E-02 1.0 -4.018548E-03 -4.700000E-02 1.0
+ -4.017897E-03 -4.600000E-02 1.0 -4.017232E-03 -4.500000E-02 1.0
+ -4.016550E-03 -4.400000E-02 1.0 -4.015852E-03 -4.300000E-02 1.0
+ -4.015137E-03 -4.200000E-02 1.0 -4.014404E-03 -4.100000E-02 1.0
+ -4.013653E-03 -4.000000E-02 1.0 -4.012883E-03 -3.900000E-02 1.0
+ -4.012091E-03 -3.800000E-02 1.0 -4.011279E-03 -3.700000E-02 1.0
+ -4.010443E-03 -3.600000E-02 1.0 -4.009583E-03 -3.500000E-02 1.0
+ -4.008697E-03 -3.400000E-02 1.0 -4.007785E-03 -3.300000E-02 1.0

+ -4.006844E-03 -3.200000E-02 1.0 -4.005873E-03 -3.100000E-02 1.0
+ -4.004869E-03 -3.000000E-02 1.0 -4.003830E-03 -2.900000E-02 1.0
+ -4.002754E-03 -2.800000E-02 1.0 -4.001637E-03 -2.700000E-02 1.0
+ -4.000478E-03 -2.600000E-02 1.0 -3.999272E-03 -2.500000E-02 1.0
+ -3.998015E-03 -2.400000E-02 1.0 -3.996701E-03 -2.300000E-02 1.0
+ -3.995328E-03 -2.200000E-02 1.0 -3.993887E-03 -2.100000E-02 1.0
+ -3.992371E-03 -2.000000E-02 1.0 -3.990772E-03 -1.900000E-02 1.0
+ -3.989078E-03 -1.800000E-02 1.0 -3.987278E-03 -1.700000E-02 1.0
+ -3.985354E-03 -1.600000E-02 1.0 -3.983287E-03 -1.500000E-02 1.0
+ -3.981050E-03 -1.400000E-02 1.0 -3.978606E-03 -1.300000E-02 1.0
+ -3.975634E-03 -1.200000E-02 1.0 -3.960263E-03 -1.100000E-02 1.0
+ -3.902319E-03 -1.000000E-02 1.0 -3.796981E-03 -9.000000E-03 1.0
+ -3.643042E-03 -8.000000E-03 1.0 -3.447052E-03 -7.000000E-03 1.0
+ -3.110053E-03 -6.000000E-03 1.0 -2.617931E-03 -5.000000E-03 1.0
+ -2.083518E-03 -4.000000E-03 1.0 -1.523693E-03 -3.000000E-03 1.0
+ -9.489568E-04 -2.000000E-03 1.0 -3.675948E-04 -1.000000E-03 1.0
+ 2.129082E-04 3.469447E-17 1.0 7.852619E-04 1.000000E-03 1.0
+ 1.341851E-03 2.000000E-03 1.0 1.873408E-03 3.000000E-03 1.0
+ 2.360864E-03 4.000000E-03 1.0 2.793225E-03 5.000000E-03 1.0
+ 3.168165E-03 6.000000E-03 1.0 3.426571E-03 7.000000E-03 1.0
+ 3.567186E-03 8.000000E-03 1.0 3.668460E-03 9.000000E-03 1.0
+ 3.732357E-03 1.000000E-02 1.0 3.763441E-03 1.100000E-02 1.0
+ 3.769891E-03 1.200000E-02 1.0 3.772004E-03 1.300000E-02 1.0
+ 3.773898E-03 1.400000E-02 1.0 3.775452E-03 1.500000E-02 1.0
+ 3.777053E-03 1.600000E-02 1.0 3.778540E-03 1.700000E-02 1.0
+ 3.779928E-03 1.800000E-02 1.0 3.781231E-03 1.900000E-02 1.0
+ 3.782459E-03 2.000000E-02 1.0 3.783621E-03 2.100000E-02 1.0
+ 3.784724E-03 2.200000E-02 1.0 3.785774E-03 2.300000E-02 1.0
+ 3.786776E-03 2.400000E-02 1.0 3.787735E-03 2.500000E-02 1.0
+ 3.788653E-03 2.600000E-02 1.0 3.789535E-03 2.700000E-02 1.0
+ 3.790383E-03 2.800000E-02 1.0 3.791200E-03 2.900000E-02 1.0
+ 3.791987E-03 3.000000E-02 1.0 3.792748E-03 3.100000E-02 1.0
+ 3.793483E-03 3.200000E-02 1.0 3.794195E-03 3.300000E-02 1.0
+ 3.794884E-03 3.400000E-02 1.0 3.795553E-03 3.500000E-02 1.0
+ 3.796201E-03 3.600000E-02 1.0 3.796831E-03 3.700000E-02 1.0
+ 3.797444E-03 3.800000E-02 1.0 3.798039E-03 3.900000E-02 1.0
+ 3.798619E-03 4.000000E-02 1.0 3.799184E-03 4.100000E-02 1.0
+ 3.799734E-03 4.200000E-02 1.0 3.800271E-03 4.300000E-02 1.0
+ 3.800795E-03 4.400000E-02 1.0 3.801306E-03 4.500000E-02 1.0
+ 3.801805E-03 4.600000E-02 1.0 3.802293E-03 4.700000E-02 1.0
+ 3.802770E-03 4.800000E-02 1.0 3.803236E-03 4.900000E-02 1.0
+ 3.855548E-03 5.000000E+00 1.0
+ -4.127342E-03 -5.000000E+00 2.0
+ -4.055178E-03 -5.000000E-02 2.0 -4.054582E-03 -4.900000E-02 2.0
+ -4.053952E-03 -4.800000E-02 2.0 -4.053308E-03 -4.700000E-02 2.0
+ -4.052649E-03 -4.600000E-02 2.0 -4.051976E-03 -4.500000E-02 2.0
+ -4.051286E-03 -4.400000E-02 2.0 -4.050581E-03 -4.300000E-02 2.0
+ -4.049858E-03 -4.200000E-02 2.0 -4.049117E-03 -4.100000E-02 2.0
+ -4.048358E-03 -4.000000E-02 2.0 -4.047578E-03 -3.900000E-02 2.0
+ -4.046778E-03 -3.800000E-02 2.0 -4.045956E-03 -3.700000E-02 2.0
+ -4.045111E-03 -3.600000E-02 2.0 -4.044241E-03 -3.500000E-02 2.0
+ -4.043346E-03 -3.400000E-02 2.0 -4.042423E-03 -3.300000E-02 2.0
+ -4.041472E-03 -3.200000E-02 2.0 -4.040489E-03 -3.100000E-02 2.0
+ -4.039474E-03 -3.000000E-02 2.0 -4.038424E-03 -2.900000E-02 2.0

+ -4.037336E-03 -2.800000E-02 2.0 -4.036207E-03 -2.700000E-02 2.0
 + -4.035034E-03 -2.600000E-02 2.0 -4.033815E-03 -2.500000E-02 2.0
 + -4.032543E-03 -2.400000E-02 2.0 -4.031215E-03 -2.300000E-02 2.0
 + -4.029826E-03 -2.200000E-02 2.0 -4.028369E-03 -2.100000E-02 2.0
 + -4.026836E-03 -2.000000E-02 2.0 -4.025219E-03 -1.900000E-02 2.0
 + -4.023507E-03 -1.800000E-02 2.0 -4.021687E-03 -1.700000E-02 2.0
 + -4.019741E-03 -1.600000E-02 2.0 -4.017651E-03 -1.500000E-02 2.0
 + -4.015388E-03 -1.400000E-02 2.0 -4.012918E-03 -1.300000E-02 2.0
 + -4.009914E-03 -1.200000E-02 2.0 -3.994658E-03 -1.100000E-02 2.0
 + -3.937069E-03 -1.000000E-02 2.0 -3.832337E-03 -9.000000E-03 2.0
 + -3.679280E-03 -8.000000E-03 2.0 -3.484361E-03 -7.000000E-03 2.0
 + -3.147170E-03 -6.000000E-03 2.0 -2.653817E-03 -5.000000E-03 2.0
 + -2.118503E-03 -4.000000E-03 2.0 -1.558464E-03 -3.000000E-03 2.0
 + -9.847577E-04 -2.000000E-03 2.0 -4.075109E-04 -1.000000E-03 2.0
 + 1.502090E-04 3.469447E-17 2.0 6.731062E-04 1.000000E-03 2.0
 + 1.159781E-03 2.000000E-03 2.0 1.608446E-03 3.000000E-03 2.0
 + 2.016640E-03 4.000000E-03 2.0 2.380417E-03 5.000000E-03 2.0
 + 2.700065E-03 6.000000E-03 2.0 2.932565E-03 7.000000E-03 2.0
 + 3.072467E-03 8.000000E-03 2.0 3.173029E-03 9.000000E-03 2.0
 + 3.236292E-03 1.000000E-02 2.0 3.266718E-03 1.100000E-02 2.0
 + 3.272456E-03 1.200000E-02 2.0 3.273887E-03 1.300000E-02 2.0
 + 3.275165E-03 1.400000E-02 2.0 3.276189E-03 1.500000E-02 2.0
 + 3.277272E-03 1.600000E-02 2.0 3.278278E-03 1.700000E-02 2.0
 + 3.279216E-03 1.800000E-02 2.0 3.280097E-03 1.900000E-02 2.0
 + 3.280927E-03 2.000000E-02 2.0 3.281713E-03 2.100000E-02 2.0
 + 3.282459E-03 2.200000E-02 2.0 3.283168E-03 2.300000E-02 2.0
 + 3.283845E-03 2.400000E-02 2.0 3.284493E-03 2.500000E-02 2.0
 + 3.285114E-03 2.600000E-02 2.0 3.285710E-03 2.700000E-02 2.0
 + 3.286283E-03 2.800000E-02 2.0 3.286835E-03 2.900000E-02 2.0
 + 3.287367E-03 3.000000E-02 2.0 3.287881E-03 3.100000E-02 2.0
 + 3.288378E-03 3.200000E-02 2.0 3.288859E-03 3.300000E-02 2.0
 + 3.289325E-03 3.400000E-02 2.0 3.289776E-03 3.500000E-02 2.0
 + 3.290215E-03 3.600000E-02 2.0 3.290640E-03 3.700000E-02 2.0
 + 3.291054E-03 3.800000E-02 2.0 3.291457E-03 3.900000E-02 2.0
 + 3.291848E-03 4.000000E-02 2.0 3.292230E-03 4.100000E-02 2.0
 + 3.292602E-03 4.200000E-02 2.0 3.292965E-03 4.300000E-02 2.0
 + 3.293318E-03 4.400000E-02 2.0 3.293664E-03 4.500000E-02 2.0
 + 3.294001E-03 4.600000E-02 2.0 3.294331E-03 4.700000E-02 2.0
 + 3.294653E-03 4.800000E-02 2.0 3.294968E-03 4.900000E-02 2.0
 + 3.330317E-03 5.000000E+00 2.0
 + -4.158073E-03 -5.000000E+00 3.0
 + -4.085263E-03 -5.000000E-02 3.0 -4.084662E-03 -4.900000E-02 3.0
 + -4.084026E-03 -4.800000E-02 3.0 -4.083376E-03 -4.700000E-02 3.0
 + -4.082712E-03 -4.600000E-02 3.0 -4.082032E-03 -4.500000E-02 3.0
 + -4.081337E-03 -4.400000E-02 3.0 -4.080625E-03 -4.300000E-02 3.0
 + -4.079896E-03 -4.200000E-02 3.0 -4.079149E-03 -4.100000E-02 3.0
 + -4.078382E-03 -4.000000E-02 3.0 -4.077596E-03 -3.900000E-02 3.0
 + -4.076789E-03 -3.800000E-02 3.0 -4.075959E-03 -3.700000E-02 3.0
 + -4.075106E-03 -3.600000E-02 3.0 -4.074229E-03 -3.500000E-02 3.0
 + -4.073326E-03 -3.400000E-02 3.0 -4.072395E-03 -3.300000E-02 3.0
 + -4.071435E-03 -3.200000E-02 3.0 -4.070444E-03 -3.100000E-02 3.0
 + -4.069420E-03 -3.000000E-02 3.0 -4.068360E-03 -2.900000E-02 3.0
 + -4.067262E-03 -2.800000E-02 3.0 -4.066123E-03 -2.700000E-02 3.0
 + -4.064940E-03 -2.600000E-02 3.0 -4.063710E-03 -2.500000E-02 3.0

+ -4.062427E-03 -2.400000E-02 3.0 -4.061088E-03 -2.300000E-02 3.0
+ -4.059686E-03 -2.200000E-02 3.0 -4.058216E-03 -2.100000E-02 3.0
+ -4.056670E-03 -2.000000E-02 3.0 -4.055038E-03 -1.900000E-02 3.0
+ -4.053310E-03 -1.800000E-02 3.0 -4.051474E-03 -1.700000E-02 3.0
+ -4.049512E-03 -1.600000E-02 3.0 -4.047403E-03 -1.500000E-02 3.0
+ -4.045120E-03 -1.400000E-02 3.0 -4.042628E-03 -1.300000E-02 3.0
+ -4.039599E-03 -1.200000E-02 3.0 -4.024493E-03 -1.100000E-02 3.0
+ -3.967348E-03 -1.000000E-02 3.0 -3.863388E-03 -9.000000E-03 3.0
+ -3.711506E-03 -8.000000E-03 3.0 -3.518150E-03 -7.000000E-03 3.0
+ -3.181890E-03 -6.000000E-03 3.0 -2.689627E-03 -5.000000E-03 3.0
+ -2.164278E-03 -4.000000E-03 3.0 -1.648767E-03 -3.000000E-03 3.0
+ -1.151683E-03 -2.000000E-03 3.0 -6.748417E-04 -1.000000E-03 3.0
+ -2.200524E-04 3.469447E-17 3.0 2.106400E-04 1.000000E-03 3.0
+ 6.148980E-04 2.000000E-03 3.0 9.900645E-04 3.000000E-03 3.0
+ 1.333025E-03 4.000000E-03 3.0 1.640163E-03 5.000000E-03 3.0
+ 1.914033E-03 6.000000E-03 3.0 2.127281E-03 7.000000E-03 3.0
+ 2.264469E-03 8.000000E-03 3.0 2.364518E-03 9.000000E-03 3.0
+ 2.427295E-03 1.000000E-02 3.0 2.457198E-03 1.100000E-02 3.0
+ 2.462348E-03 1.200000E-02 3.0 2.463210E-03 1.300000E-02 3.0
+ 2.463976E-03 1.400000E-02 3.0 2.464574E-03 1.500000E-02 3.0
+ 2.465224E-03 1.600000E-02 3.0 2.465827E-03 1.700000E-02 3.0
+ 2.466390E-03 1.800000E-02 3.0 2.466919E-03 1.900000E-02 3.0
+ 2.467417E-03 2.000000E-02 3.0 2.467888E-03 2.100000E-02 3.0
+ 2.468335E-03 2.200000E-02 3.0 2.468761E-03 2.300000E-02 3.0
+ 2.469167E-03 2.400000E-02 3.0 2.469556E-03 2.500000E-02 3.0
+ 2.469928E-03 2.600000E-02 3.0 2.470286E-03 2.700000E-02 3.0
+ 2.470629E-03 2.800000E-02 3.0 2.470960E-03 2.900000E-02 3.0
+ 2.471279E-03 3.000000E-02 3.0 2.471588E-03 3.100000E-02 3.0
+ 2.471886E-03 3.200000E-02 3.0 2.472174E-03 3.300000E-02 3.0
+ 2.472453E-03 3.400000E-02 3.0 2.472724E-03 3.500000E-02 3.0
+ 2.472987E-03 3.600000E-02 3.0 2.473242E-03 3.700000E-02 3.0
+ 2.473490E-03 3.800000E-02 3.0 2.473732E-03 3.900000E-02 3.0
+ 2.473967E-03 4.000000E-02 3.0 2.474195E-03 4.100000E-02 3.0
+ 2.474418E-03 4.200000E-02 3.0 2.474636E-03 4.300000E-02 3.0
+ 2.474848E-03 4.400000E-02 3.0 2.475055E-03 4.500000E-02 3.0
+ 2.475257E-03 4.600000E-02 3.0 2.475455E-03 4.700000E-02 3.0
+ 2.475648E-03 4.800000E-02 3.0 2.475837E-03 4.900000E-02 3.0
+ 2.497038E-03 5.000000E+00 3.0
+ -4.185645E-03 -5.000000E+00 4.0
+ -4.112282E-03 -5.000000E-02 4.0 -4.111677E-03 -4.900000E-02 4.0
+ -4.111036E-03 -4.800000E-02 4.0 -4.110381E-03 -4.700000E-02 4.0
+ -4.109712E-03 -4.600000E-02 4.0 -4.109027E-03 -4.500000E-02 4.0
+ -4.108327E-03 -4.400000E-02 4.0 -4.107609E-03 -4.300000E-02 4.0
+ -4.106875E-03 -4.200000E-02 4.0 -4.106122E-03 -4.100000E-02 4.0
+ -4.105349E-03 -4.000000E-02 4.0 -4.104557E-03 -3.900000E-02 4.0
+ -4.103743E-03 -3.800000E-02 4.0 -4.102908E-03 -3.700000E-02 4.0
+ -4.102048E-03 -3.600000E-02 4.0 -4.101165E-03 -3.500000E-02 4.0
+ -4.100255E-03 -3.400000E-02 4.0 -4.099317E-03 -3.300000E-02 4.0
+ -4.098350E-03 -3.200000E-02 4.0 -4.097351E-03 -3.100000E-02 4.0
+ -4.096319E-03 -3.000000E-02 4.0 -4.095251E-03 -2.900000E-02 4.0
+ -4.094145E-03 -2.800000E-02 4.0 -4.092997E-03 -2.700000E-02 4.0
+ -4.091806E-03 -2.600000E-02 4.0 -4.090566E-03 -2.500000E-02 4.0
+ -4.089274E-03 -2.400000E-02 4.0 -4.087924E-03 -2.300000E-02 4.0
+ -4.086512E-03 -2.200000E-02 4.0 -4.085030E-03 -2.100000E-02 4.0

+ -4.083472E-03 -2.000000E-02 4.0 -4.081829E-03 -1.900000E-02 4.0
+ -4.080088E-03 -1.800000E-02 4.0 -4.078237E-03 -1.700000E-02 4.0
+ -4.076260E-03 -1.600000E-02 4.0 -4.074135E-03 -1.500000E-02 4.0
+ -4.071835E-03 -1.400000E-02 4.0 -4.069325E-03 -1.300000E-02 4.0
+ -4.066275E-03 -1.200000E-02 4.0 -4.051374E-03 -1.100000E-02 4.0
+ -3.994849E-03 -1.000000E-02 4.0 -3.892092E-03 -9.000000E-03 4.0
+ -3.742740E-03 -8.000000E-03 4.0 -3.576491E-03 -7.000000E-03 4.0
+ -3.310323E-03 -6.000000E-03 4.0 -2.922299E-03 -5.000000E-03 4.0
+ -2.521804E-03 -4.000000E-03 4.0 -2.120647E-03 -3.000000E-03 4.0
+ -1.724743E-03 -2.000000E-03 4.0 -1.338148E-03 -1.000000E-03 4.0
+ -9.643097E-04 3.469447E-17 4.0 -6.064875E-04 1.000000E-03 4.0
+ -2.679003E-04 2.000000E-03 4.0 4.820974E-05 3.000000E-03 4.0
+ 3.386082E-04 4.000000E-03 4.0 5.995746E-04 5.000000E-03 4.0
+ 8.360584E-04 6.000000E-03 4.0 1.030550E-03 7.000000E-03 4.0
+ 1.168672E-03 8.000000E-03 4.0 1.268355E-03 9.000000E-03 4.0
+ 1.330782E-03 1.000000E-02 4.0 1.360278E-03 1.100000E-02 4.0
+ 1.364951E-03 1.200000E-02 4.0 1.365345E-03 1.300000E-02 4.0
+ 1.365689E-03 1.400000E-02 4.0 1.365949E-03 1.500000E-02 4.0
+ 1.366242E-03 1.600000E-02 4.0 1.366513E-03 1.700000E-02 4.0
+ 1.366766E-03 1.800000E-02 4.0 1.367004E-03 1.900000E-02 4.0
+ 1.367228E-03 2.000000E-02 4.0 1.367439E-03 2.100000E-02 4.0
+ 1.367640E-03 2.200000E-02 4.0 1.367831E-03 2.300000E-02 4.0
+ 1.368014E-03 2.400000E-02 4.0 1.368189E-03 2.500000E-02 4.0
+ 1.368356E-03 2.600000E-02 4.0 1.368516E-03 2.700000E-02 4.0
+ 1.368671E-03 2.800000E-02 4.0 1.368819E-03 2.900000E-02 4.0
+ 1.368963E-03 3.000000E-02 4.0 1.369101E-03 3.100000E-02 4.0
+ 1.369235E-03 3.200000E-02 4.0 1.369365E-03 3.300000E-02 4.0
+ 1.369490E-03 3.400000E-02 4.0 1.369612E-03 3.500000E-02 4.0
+ 1.369730E-03 3.600000E-02 4.0 1.369844E-03 3.700000E-02 4.0
+ 1.369956E-03 3.800000E-02 4.0 1.370064E-03 3.900000E-02 4.0
+ 1.370170E-03 4.000000E-02 4.0 1.370272E-03 4.100000E-02 4.0
+ 1.370373E-03 4.200000E-02 4.0 1.370470E-03 4.300000E-02 4.0
+ 1.370566E-03 4.400000E-02 4.0 1.370659E-03 4.500000E-02 4.0
+ 1.370749E-03 4.600000E-02 4.0 1.370838E-03 4.700000E-02 4.0
+ 1.370925E-03 4.800000E-02 4.0 1.371010E-03 4.900000E-02 4.0
+ 1.379877E-03 5.000000E+00 4.0
+ -4.210950E-03 -5.000000E+00 5.0
+ -4.137094E-03 -5.000000E-02 5.0 -4.136485E-03 -4.900000E-02 5.0
+ -4.135840E-03 -4.800000E-02 5.0 -4.135180E-03 -4.700000E-02 5.0
+ -4.134506E-03 -4.600000E-02 5.0 -4.133817E-03 -4.500000E-02 5.0
+ -4.133112E-03 -4.400000E-02 5.0 -4.132390E-03 -4.300000E-02 5.0
+ -4.131650E-03 -4.200000E-02 5.0 -4.130892E-03 -4.100000E-02 5.0
+ -4.130114E-03 -4.000000E-02 5.0 -4.129317E-03 -3.900000E-02 5.0
+ -4.128498E-03 -3.800000E-02 5.0 -4.127657E-03 -3.700000E-02 5.0
+ -4.126792E-03 -3.600000E-02 5.0 -4.125902E-03 -3.500000E-02 5.0
+ -4.124986E-03 -3.400000E-02 5.0 -4.124042E-03 -3.300000E-02 5.0
+ -4.123068E-03 -3.200000E-02 5.0 -4.122063E-03 -3.100000E-02 5.0
+ -4.121024E-03 -3.000000E-02 5.0 -4.119949E-03 -2.900000E-02 5.0
+ -4.118835E-03 -2.800000E-02 5.0 -4.117680E-03 -2.700000E-02 5.0
+ -4.116480E-03 -2.600000E-02 5.0 -4.115232E-03 -2.500000E-02 5.0
+ -4.113931E-03 -2.400000E-02 5.0 -4.112572E-03 -2.300000E-02 5.0
+ -4.111151E-03 -2.200000E-02 5.0 -4.109660E-03 -2.100000E-02 5.0
+ -4.108091E-03 -2.000000E-02 5.0 -4.106436E-03 -1.900000E-02 5.0
+ -4.104684E-03 -1.800000E-02 5.0 -4.102821E-03 -1.700000E-02 5.0

+ -4.100831E-03 -1.600000E-02 5.0 -4.098692E-03 -1.500000E-02 5.0
+ -4.096377E-03 -1.400000E-02 5.0 -4.093849E-03 -1.300000E-02 5.0
+ -4.090828E-03 -1.200000E-02 5.0 -4.087637E-03 -1.100000E-02 5.0
+ -4.083953E-03 -1.000000E-02 5.0 -4.079531E-03 -9.000000E-03 5.0
+ -4.073819E-03 -8.000000E-03 5.0 -4.064737E-03 -7.000000E-03 5.0
+ -3.917042E-03 -6.000000E-03 5.0 -3.632121E-03 -5.000000E-03 5.0
+ -3.323215E-03 -4.000000E-03 5.0 -3.004271E-03 -3.000000E-03 5.0
+ -2.682691E-03 -2.000000E-03 5.0 -2.363612E-03 -1.000000E-03 5.0
+ -2.051295E-03 3.469447E-17 5.0 -1.749606E-03 1.000000E-03 5.0
+ -1.462203E-03 2.000000E-03 5.0 -1.192593E-03 3.000000E-03 5.0
+ -9.441245E-04 4.000000E-03 5.0 -7.199399E-04 5.000000E-03 5.0
+ -5.135435E-04 6.000000E-03 5.0 -3.342268E-04 7.000000E-03 5.0
+ -1.951575E-04 8.000000E-03 5.0 -9.569127E-05 9.000000E-03 5.0
+ -3.349815E-05 1.000000E-02 5.0 -4.304090E-06 1.100000E-02 5.0
+ -1.434517E-08 1.200000E-02 5.0 -1.039336E-09 1.300000E-02 5.0
+ -1.592208E-10 1.400000E-02 5.0 -1.550494E-10 1.500000E-02 5.0
+ -1.231544E-10 1.600000E-02 5.0 -1.036630E-10 1.700000E-02 5.0
+ -8.911986E-11 1.800000E-02 5.0 -7.801443E-11 1.900000E-02 5.0
+ -6.927459E-11 2.000000E-02 5.0 -6.222390E-11 2.100000E-02 5.0
+ -5.642128E-11 2.200000E-02 5.0 -5.156792E-11 2.300000E-02 5.0
+ -4.745366E-11 2.400000E-02 5.0 -4.392649E-11 2.500000E-02 5.0
+ -4.087340E-11 2.600000E-02 5.0 -3.820870E-11 2.700000E-02 5.0
+ -3.586593E-11 2.800000E-02 5.0 -3.379302E-11 2.900000E-02 5.0
+ -3.194835E-11 3.000000E-02 5.0 -3.029839E-11 3.100000E-02 5.0
+ -2.881573E-11 3.200000E-02 5.0 -2.747785E-11 3.300000E-02 5.0
+ -2.626595E-11 3.400000E-02 5.0 -2.516434E-11 3.500000E-02 5.0
+ -2.415967E-11 3.600000E-02 5.0 -2.324074E-11 3.700000E-02 5.0
+ -2.239790E-11 3.800000E-02 5.0 -2.162284E-11 3.900000E-02 5.0
+ -2.090834E-11 4.000000E-02 5.0 -2.024822E-11 4.100000E-02 5.0
+ -1.963705E-11 4.200000E-02 5.0 -1.907013E-11 4.300000E-02 5.0
+ -1.854317E-11 4.400000E-02 5.0 -1.805250E-11 4.500000E-02 5.0
+ -1.759489E-11 4.600000E-02 5.0 -1.716748E-11 4.700000E-02 5.0
+ -1.676757E-11 4.800000E-02 5.0 -1.639293E-11 4.900000E-02 5.0
+ 4.151303E-08 5.000000E+00 5.0

APPENDIX F

EXAMPLE SPICE OUTPUT DECK FOR THE
SIMPLIFIED MACRO MODEL


```

SSSS  UU  UU PPPPPP PPPPPP LL  EEEEEEEE 3333
SS SS UU  UU PP PP PP PP LL  EE 33 33
SS  UU  UU PP PP PP PP LL  EE 33
SS  UU  UU PP PP PP PP LL  EE 33
SS  UU  UU PPPPPP PPPPPP LL  EEEEEEEE 333
SS UU  UU PP  PP  LL  EE 33
SS UU  UU PP  PP  LL  EE 33
SS UU  UU PP  PP  LL  EE 33
SS SS UU  UU PP  PP  LL  EE 33 33
SSSS  UUUU PP  PP  LLLLLLLL EEEEEEEE 3333

```


* SUPPLE3 version 0.03 - October 18, 1991 *

SUPPLE3 IS A GENERAL PURPOSE CIRCUIT ANALYSIS PROGRAM
MAINTAINED AT TEXAS INSTRUMENTS BY THE
DESIGN AUTOMATION DIVISION IN DALLAS
PROPERTY OF TEXAS INSTRUMENTS -- FOR UNRESTRICTED
INTERNAL
USE ONLY -- UNAUTHORIZED REPRODUCTION AND/OR
DISTRIBUTION
IS STRICTLY PROHIBITED. THIS PRODUCT IS PROTECTED UNDER
COPYRIGHT LAW AND TRADE SECRET LAW AS AN UNPUBLISHED
WORK.
CREATED 1991, (C) COPYRIGHT 1991, TEXAS INSTRUMENTS. ALL
RIGHTS RESERVED.

THESE COMMODITIES ARE UNDER U. S. GOVERNMENT
DISTRIBUTION
LICENSE CONTROL. AS SUCH, THEY ARE NOT TO BE RE-EXPORTED
WITHOUT PRIOR APPROVAL OF THE U.S. DEPARTMENT OF
COMMERCE.

* INPUT LISTING *
* TEMPERATURE = 27.000 *
* RUN DATE = 12/17/91 RUN NUMBER 1 RUN TIME = 13:52:13 *
*

* Comparator design
* begun on computer 6/25/91
*

```

*****
* Analysis Section
*****
.WIDTH IN = 80 OUT = 80
*.DC VPLUS -.1 .1 .25M
*.PUNCH DC I(VOUT)
*.PRINT DC I(VOUT)
*.AC DEC 10 100 1E14
*.PUNCH AC VDB(4) VP(4)
*.PZ 6 0 4 0 VOL
.FORCE NODE 4 5.0 XCOMP1.4 0 XCOMP2.4 0 XCOMP3.4 0 XCOMP6.4 0
XCOMP5.4 0
+ XCOMP1.3 0 XCOMP2.3 0 XCOMP3.3 0 XCOMP6.3 0 XCOMP5.3 0
*.FORCE 4 -5.0
.TRAN .01N 1M
.PUNCH TR V(4)
*.INITIAL TR NODE 4 -5.0 7 -5.0
*.TF V(4) VPLUS
*.OP
.options itl1 = 100000 itl2 = 100000 ITL4 = 2000 RMIN = 1E-14 ITLPZ = 10000
+ PZMAXR = 5E11
*****
* Power Rails and Inputs
*****
* -.425M offset voltage
*.VPLUS 6 0 DC 0 AC 1
*.VMINUS 7 4 DC 0
*.XCOMP 6 7 4 COMP
*.VOUT 4 0 0
*.Rf 7 4 .1

*XZERON 6 5 NZERO ZEROVAL = -6.28E + 05
*XPOLE1 5 3 POLE POLEVAL = 6.28E + 03
*XPOLE2 3 4 POLE POLEVAL = 6.28E + 07

VP1 1 0 -2
VP2 2 0 -2
VP3 3 0 -2
VP5 5 0 -2
VP6 6 0 -1
VP7 7 0 2
VP8 8 0 2.3
VP9 9 0 3.1
VP10 10 0 4
XCOMP1 1 4 4 COMP
XCOMP2 2 4 4 COMP
XCOMP3 3 4 4 COMP
XCOMP5 5 4 4 COMP
XCOMP6 6 4 4 COMP
XCOMP7 7 4 4 COMP
XCOMP8 8 4 4 COMP
XCOMP9 9 4 4 COMP
XCOMP10 10 4 4 COMP

```

* macro modeling section

```
.SUBCKT POLE 1 2 POLEVAL = 1E3
CPOLE 2 0 1
EPOLE 3 0 1 0 1
RPOLE 2 3 #1/POLEVAL#
.ENDS POLE
```

```
.SUBCKT PZERO 4 3 ZEROVAL = 1E3
EIN 1 0 0 4 #1/ZEROVAL#
VIN 1 5 0
CIN 5 0 1
HIN 2 0 VIN -1
EOUT 2 3 4 0 1
COUT 3 0 1
.ENDS PZERO
```

```
.SUBCKT NZERO 4 3 ZEROVAL = 1E3
EIN 1 0 0 4 #1/ZEROVAL#
VIN 1 5 0
CIN 5 0 1
HIN 2 0 VIN 1
EOUT 2 3 4 0 -1
COUT 3 0 1
.ENDS NZERO
```

```
.SUBCKT PATHAC 1 26
CIN 1 0 1P
XP1 1 2 POLE POLEVAL = 1.03067E + 04
XP3 2 4 POLE POLEVAL = 2.90000E + 09
EGAIN 26 0 4 0 1
ROUT 26 0 1MEG
.ENDS PATHAC
```

```
.SUBCKT PATHDC 1 2
GOUT 0 2 PWL(2) 1 0 2 0 IOUT
.LIBRARY DSN = /user/jwagnon/thesis/spice/input/compout.pwl NOPRINT
.ENDS PATHDC
```

* input capacitance is 30picoFarads

```
.SUBCKT COMP 1 2 5
CINPLUS 1 0 26.2P
CINMINUS 2 0 3.02P
EIN 3 0 1 2 1
XAC 3 4 PATHAC
XDC 4 5 PATHDC
COUT 5 0 2.827P
ROUT 5 0 100G
.ENDS COMP
```


.END

```

*****
*****
*
*           CAPACITOR MODEL PARAMETERS
*           TEMPERATURE = 27.000
* RUN DATE = 12/17/91      RUN NUMBER 1      RUN TIME = 13:52:13
*
*****
*****

```

```

NAME  C$MOD$
CJ    0.000
CJSW  0.000
DEFW  1.00E-06
NARROW 0.000

```

```

*****
*****
*
*           RESISTOR MODEL PARAMETERS
*           TEMPERATURE = 27.000
* RUN DATE = 12/17/91      RUN NUMBER 1      RUN TIME = 13:52:13
*
*****
*****

```

```

NAME  R$MOD$
TC1   0.000
TC2   0.000
RSH   0.000
DEFW  1.00E-06
NARROW 0.000

```

```

*****
*****
*
*           PWL MODEL PARAMETERS
*           TEMPERATURE = 27.000
* RUN DATE = 12/17/91      RUN NUMBER 1      RUN TIME = 13:52:13
*
*****
*****

```

NAME IOUT

```

          FIRST   SECOND
DEPENDENT INDEPENDENT INDEPENDENT
VARIABLE  VARIABLE  VARIABLE

```

*OMITTED SEE APPENDIX E

```

=====
=====
=====
=
PARAMETERS
=

```

```

=====
=====
=====

```

NAME	VALUE	TC1	TC2
PL	1.00E+00	0.00E+00	0.00E+00
PC	1.00E+00	0.00E+00	0.00E+00
PR	1.00E+00	0.00E+00	0.00E+00

```

=====
=====
=====

```

INDEPENDENT VOLTAGE SOURCES

```

=====
=====
=====

```

NAME	N+	N-	DC VALUE	AC VALUE	AC PHASE
TRANSIENT					
VP10	10	0	4.000	0.000	0.000
VP9	9	0	3.100	0.000	0.000
VP8	8	0	2.300	0.000	0.000
VP7	7	0	2.000	0.000	0.000
VP6	6	0	-1.000	0.000	0.000
VP5	5	0	-2.000	0.000	0.000
VP3	3	0	-2.000	0.000	0.000
VP2	2	0	-2.000	0.000	0.000
VP1	1	0	-2.000	0.000	0.000

```

=====
=====
=====

```

SUBCIRCUIT CALLS

```

=====
=====
=====

```

NAME	SUBCIRCUIT	EXTERNAL NODES
XCOMP10	COMP	10 4 4
XCOMP9	COMP	9 4 4
XCOMP8	COMP	8 4 4
XCOMP7	COMP	7 4 4
XCOMP6	COMP	6 4 4

XCOMP5 COMP 5 4 4
 XCOMP3 COMP 3 4 4
 XCOMP2 COMP 2 4 4
 XCOMP1 COMP 1 4 4

 * ELEMENT NODE TABLE *
 * TEMPERATURE = 27.000 *
 * RUN DATE = 12/17/91 RUN NUMBER 1 RUN TIME = 13:52:13 *
 *

XCOMP9.X XCOMP9.X XCOMP9.X
 AC.XP3.3 AC.XP3.E AC.XP3.R
 POLE POLE

0 VP1 VP10 VP2 VP3 VP5 VP6

VP7 VP8 VP9 XCOMP1.C XCOMP1.C XCOMP1.C
 INMINUS INPLUS OUT

XCOMP1.E XCOMP1.R XCOMP1.X XCOMP1.X XCOMP1.X XCOMP1.X

IN OUT AC.CIN AC.EGAIN AC.ROUT AC.XP1.C
 POLE

XCOMP1.X XCOMP1.X XCOMP1.X XCOMP1.X XCOMP10. XCOMP10.
 AC.XP1.E AC.XP3.C AC.XP3.E DC.GOUT CINMINUS CINPLUS
 POLE POLE POLE

XCOMP10. XCOMP10. XCOMP10. XCOMP10. XCOMP10. XCOMP10.
 COUT EIN ROUT XAC.CIN XAC.EGAI XAC.ROUT
 N

XCOMP10. XCOMP10. XCOMP10. XCOMP10. XCOMP10. XCOMP2.C
 XAC.XP1. XAC.XP1. XAC.XP3. XAC.XP3. XDC.GOUT INMINUS
 CPOLE EPOLE CPOLE EPOLE

XCOMP2.C XCOMP2.C XCOMP2.E XCOMP2.R XCOMP2.X XCOMP2.X

INPLUS OUT IN OUT AC.CIN AC.EGAIN

XCOMP2.X XCOMP2.X XCOMP2.X XCOMP2.X XCOMP2.X XCOMP2.X
 AC.ROUT AC.XP1.C AC.XP1.E AC.XP3.C AC.XP3.E DC.GOUT
 POLE POLE POLE POLE

XCOMP3.C XCOMP3.C XCOMP3.C XCOMP3.E XCOMP3.R
 XCOMP3.X
 INMINUS INPLUS OUT IN OUT AC.CIN

XCOMP3.X XCOMP3.X XCOMP3.X XCOMP3.X XCOMP3.X XCOMP3.X
 AC.EGAIN AC.ROUT AC.XP1.C AC.XP1.E AC.XP3.C AC.XP3.E
 POLE POLE POLE POLE

XCOMP3.X XCOMP5.C XCOMP5.C XCOMP5.C XCOMP5.E
 XCOMP5.R
 DC.GOUT INMINUS INPLUS OUT IN OUT

 XCOMP5.X XCOMP5.X XCOMP5.X XCOMP5.X XCOMP5.X XCOMP5.X
 AC.CIN AC.EGAIN AC.ROUT AC.XP1.C AC.XP1.E AC.XP3.C
 POLE POLE POLE

 XCOMP5.X XCOMP5.X XCOMP6.C XCOMP6.C XCOMP6.C
 XCOMP6.E
 AC.XP3.E DC.GOUT INMINUS INPLUS OUT IN
 POLE

 XCOMP6.R XCOMP6.X XCOMP6.X XCOMP6.X XCOMP6.X XCOMP6.X
 OUT AC.CIN AC.EGAIN AC.ROUT AC.XP1.C AC.XP1.E
 POLE POLE

 XCOMP6.X XCOMP6.X XCOMP6.X XCOMP7.C XCOMP7.C XCOMP7.C

 AC.XP3.C AC.XP3.E DC.GOUT INMINUS INPLUS OUT
 POLE POLE

 XCOMP7.E XCOMP7.R XCOMP7.X XCOMP7.X XCOMP7.X XCOMP7.X

 IN OUT AC.CIN AC.EGAIN AC.ROUT AC.XP1.C
 POLE

 XCOMP7.X XCOMP7.X XCOMP7.X XCOMP7.X XCOMP8.C XCOMP8.C

 AC.XP1.E AC.XP3.C AC.XP3.E DC.GOUT INMINUS INPLUS
 POLE POLE POLE

 XCOMP8.C XCOMP8.E XCOMP8.R XCOMP8.X XCOMP8.X XCOMP8.X

 OUT IN OUT AC.CIN AC.EGAIN AC.ROUT

 XCOMP8.X XCOMP8.X XCOMP8.X XCOMP8.X XCOMP8.X XCOMP9.C

 AC.XP1.C AC.XP1.E AC.XP3.C AC.XP3.E DC.GOUT INMINUS
 POLE POLE POLE POLE

 XCOMP9.C XCOMP9.C XCOMP9.E XCOMP9.R XCOMP9.X XCOMP9.X

 INPLUS OUT IN OUT AC.CIN AC.EGAIN

 XCOMP9.X XCOMP9.X XCOMP9.X XCOMP9.X XCOMP9.X XCOMP9.X
 AC.ROUT AC.XP1.C AC.XP1.E AC.XP3.C AC.XP3.E DC.GOUT
 POLE POLE POLE POLE

 1 VP1 XCOMP1.C
 INPLUS

 2 VP2 XCOMP2.C
 INPLUS

 3 VP3 XCOMP3.C
 INPLUS

4 XCOMP1.C XCOMP1.C XCOMP1.R XCOMP1.X XCOMP10.
 XCOMP10.
 INMINUS OUT OUT DC.GOUT CINMINUS COUT
 XCOMP10. XCOMP10. XCOMP2.C XCOMP2.C XCOMP2.R XCOMP2.X
 ROUT XDC.GOUT INMINUS OUT OUT DC.GOUT
 XCOMP3.C XCOMP3.C XCOMP3.R XCOMP3.X XCOMP5.C
 XCOMP5.C
 INMINUS OUT OUT DC.GOUT INMINUS OUT
 XCOMP5.R XCOMP5.X XCOMP6.C XCOMP6.C XCOMP6.R XCOMP6.X
 OUT DC.GOUT INMINUS OUT OUT DC.GOUT
 XCOMP7.C XCOMP7.C XCOMP7.R XCOMP7.X XCOMP8.C
 XCOMP8.C
 INMINUS OUT OUT DC.GOUT INMINUS OUT
 XCOMP8.R XCOMP8.X XCOMP9.C XCOMP9.C XCOMP9.R XCOMP9.X
 OUT DC.GOUT INMINUS OUT OUT DC.GOUT

5 VP5 XCOMP5.C
 INPLUS

6 VP6 XCOMP6.C
 INPLUS

7 VP7 XCOMP7.C
 INPLUS

8 VP8 XCOMP8.C
 INPLUS

9 VP9 XCOMP9.C
 INPLUS

XCOMP6.X XCOMP6.X XCOMP6.X
 AC.2 AC.XP1.C AC.XP1.R
 POLE POLE

XCOMP6.X XCOMP6.X XCOMP6.X
 AC.4 AC.XP3.C AC.XP3.R
 POLE POLE

XCOMP5.X XCOMP5.X XCOMP5.X
 AC.XP1.3 AC.XP1.E AC.XP1.R
 POLE POLE

XCOMP7.3 XCOMP7.E XCOMP7.X
 IN AC.CIN

XCOMP7.4 XCOMP7.X XCOMP7.X
 AC.EGAIN AC.ROUT

XCOMP5.X XCOMP5.X XCOMP5.X
 AC.2 AC.XP1.C AC.XP1.R
 POLE POLE

XCOMP5.X XCOMP5.X XCOMP5.X
 AC.4 AC.XP3.C AC.XP3.R
 POLE POLE

XCOMP3.X XCOMP3.X XCOMP3.X
 AC.XP3.3 AC.XP3.E AC.XP3.R
 POLE POLE

XCOMP8.X XCOMP8.X XCOMP8.X
 AC.XP3.3 AC.XP3.E AC.XP3.R
 POLE POLE

XCOMP2.3 XCOMP2.E XCOMP2.X
 IN AC.CIN

XCOMP2.4 XCOMP2.X XCOMP2.X
 AC.EGAIN AC.ROUT

XCOMP9.X XCOMP9.X XCOMP9.X
 AC.XP1.3 AC.XP1.E AC.XP1.R
 POLE POLE

XCOMP10. XCOMP10. XCOMP10.
 3 EIN XAC.CIN

XCOMP10. XCOMP10. XCOMP10.
 4 XAC.EGAI XAC.ROUT
 N

XCOMP5.3 XCOMP5.E XCOMP5.X
 IN AC.CIN

XCOMP5.4 XCOMP5.X XCOMP5.X
 AC.EGAIN AC.ROUT

XCOMP2.X XCOMP2.X XCOMP2.X
 AC.XP3.3 AC.XP3.E AC.XP3.R
 POLE POLE

XCOMP7.X XCOMP7.X XCOMP7.X
 AC.XP3.3 AC.XP3.E AC.XP3.R
 POLE POLE

XCOMP3.X XCOMP3.X XCOMP3.X
 AC.2 AC.XP1.C AC.XP1.R
 POLE POLE

XCOMP3.X XCOMP3.X XCOMP3.X
 AC.4 AC.XP3.C AC.XP3.R
 POLE POLE

XCOMP8.3 XCOMP8.E XCOMP8.X
 IN AC.CIN

XCOMP8.4 XCOMP8.X XCOMP8.X
AC.EGAIN AC.ROUT

XCOMP10. XCOMP10. XCOMP10.
XAC.XP3. XAC.XP3. XAC.XP3.
3 EPOLE RPOLE

XCOMP3.X XCOMP3.X XCOMP3.X
AC.XP1.3 AC.XP1.E AC.XP1.R
POLE POLE

XCOMP8.X XCOMP8.X XCOMP8.X
AC.XP1.3 AC.XP1.E AC.XP1.R
POLE POLE

XCOMP2.X XCOMP2.X XCOMP2.X
AC.2 AC.XP1.C AC.XP1.R
POLE POLE

XCOMP2.X XCOMP2.X XCOMP2.X
AC.4 AC.XP3.C AC.XP3.R
POLE POLE

XCOMP1.X XCOMP1.X XCOMP1.X
AC.XP3.3 AC.XP3.E AC.XP3.R
POLE POLE

XCOMP6.X XCOMP6.X XCOMP6.X
AC.XP3.3 AC.XP3.E AC.XP3.R
POLE POLE

XCOMP10. XCOMP10. XCOMP10.
XAC.2 XAC.XP1. XAC.XP1.
CPOLE RPOLE

XCOMP10. XCOMP10. XCOMP10.
XAC.4 XAC.XP3. XAC.XP3.
CPOLE RPOLE

XCOMP3.3 XCOMP3.E XCOMP3.X
IN AC.CIN

XCOMP3.4 XCOMP3.X XCOMP3.X
AC.EGAIN AC.ROUT

XCOMP1.X XCOMP1.X XCOMP1.X
AC.2 AC.XP1.C AC.XP1.R
POLE POLE

XCOMP1.X XCOMP1.X XCOMP1.X
AC.4 AC.XP3.C AC.XP3.R
POLE POLE

XCOMP2.X XCOMP2.X XCOMP2.X
AC.XP1.3 AC.XP1.E AC.XP1.R
POLE POLE

XCOMP7.X XCOMP7.X XCOMP7.X
 AC.XP1.3 AC.XP1.E AC.XP1.R
 POLE POLE

XCOMP9.X XCOMP9.X XCOMP9.X
 AC.2 AC.XP1.C AC.XP1.R
 POLE POLE

XCOMP9.X XCOMP9.X XCOMP9.X
 AC.4 AC.XP3.C AC.XP3.R
 POLE POLE

XCOMP6.3 XCOMP6.E XCOMP6.X
 IN AC.CIN

XCOMP6.4 XCOMP6.X XCOMP6.X
 AC.EGAIN AC.ROUT

XCOMP10. XCOMP10. XCOMP10.
 XAC.XP1. XAC.XP1. XAC.XP1.
 3 EPOLE RPOLE

XCOMP5.X XCOMP5.X XCOMP5.X
 AC.XP3.3 AC.XP3.E AC.XP3.R
 POLE POLE

XCOMP9.3 XCOMP9.E XCOMP9.X
 IN AC.CIN

XCOMP9.4 XCOMP9.X XCOMP9.X
 AC.EGAIN AC.ROUT

XCOMP8.X XCOMP8.X XCOMP8.X
 AC.2 AC.XP1.C AC.XP1.R
 POLE POLE

XCOMP8.X XCOMP8.X XCOMP8.X
 AC.4 AC.XP3.C AC.XP3.R
 POLE POLE

XCOMP1.X XCOMP1.X XCOMP1.X
 AC.XP1.3 AC.XP1.E AC.XP1.R
 POLE POLE

XCOMP6.X XCOMP6.X XCOMP6.X
 AC.XP1.3 AC.XP1.E AC.XP1.R
 POLE POLE

XCOMP1.3 XCOMP1.E XCOMP1.X
 IN AC.CIN

XCOMP1.4 XCOMP1.X XCOMP1.X
 AC.EGAIN AC.ROUT

XCOMP7.X XCOMP7.X XCOMP7.X
 AC.2 AC.XP1.C AC.XP1.R
 POLE POLE

XCOMP7.X XCOMP7.X XCOMP7.X
 AC.4 AC.XP3.C AC.XP3.R
 POLE POLE

10 VP10 XCOMP10.
 CINPLUS

 * OPTION SUMMARY *
 * TEMPERATURE = 27.000 *
 * RUN DATE = 12/17/91 RUN NUMBER 1 RUN TIME = 13:52:13
 *

DC ANALYSIS -

GMIN = 1.00E-12
 TOL = 1.00E-05
 ABSTOL = 1.00E-12
 LVLCOD = 1
 ITL1 = 100000
 ITL2 = 100000
 DCRTOL = 1.00E-06
 DCATOL = 1.00E-12
 DCVTOL = 1.00E-06
 CKDC = 2.50E+00
 CKBJT = 2.50E+01
 ITLBJT = 300

TRANSIENT ANALYSIS -

METHOD = GEAR
 MAXORD = 2
 TSATOL = 1.00E-03
 LVLTIM = 4
 ITL3 = 4
 ITL4 = 2000
 ITL5 = 50000
 TRATOL = 1.00E-12

MISCELLANEOUS -

LIMPTS = 10001
 LIMTIM = 5.00E+00
 ANTIME = 0.00E+00
 NUMDGT = 4
 TNOM = 2.70E+01
 CAPMAX = 1.00E-03
 MINOUT = 50
 RMIN = 1.00E-14
 LVLSUB = 6

MACRO -

VDIODE = -1.00E+00
 VMAX = 5.00E+00
 VBB = 0.00E+00

```

*****
*****
*           INITIAL TRANSIENT SOLUTION           *
*           TEMPERATURE = 27.000                *
* RUN DATE = 12/17/91      RUN NUMBER 1      RUN TIME = 13:52:13
*
*****
*****

```

```

NODE  VOLTAGE  NODE  VOLTAGE  NODE  VOLTAGE  NODE
VOLTAGE
XCOMP9.X -1.900 1      -2.000 2      -2.000 3      -2.000
AC.XP3.3

4      5.000 5      -2.000 6      -1.000 7      2.000

8      2.300 9      3.100 XCOMP6.X -5.99E-03 XCOMP6.X -5.99E-03
      AC.2      AC.4

XCOMP5.X -6.99E-03 XCOMP7.3 -3.000 XCOMP7.4 -3.000 XCOMP5.X
-6.99E-03
AC.XP1.3      AC.2

XCOMP5.X -6.99E-03 XCOMP3.X -6.99E-03 XCOMP8.X -2.700 XCOMP2.3
-6.99E-03
AC.4      AC.XP3.3      AC.XP3.3

XCOMP2.4 -6.99E-06 XCOMP9.X -1.900 XCOMP10. -1.000 XCOMP10.
-1.000
      AC.XP1.3      3      4

XCOMP5.3 -6.99E-03 XCOMP5.4 -6.99E-06 XCOMP2.X -6.99E-03
XCOMP7.X -3.000
      AC.XP3.3      AC.XP3.3

XCOMP3.X -6.99E-03 XCOMP3.X -6.99E-03 XCOMP8.3 -2.700 XCOMP8.4
-2.700
AC.2      AC.4

XCOMP10. -1.000 XCOMP3.X -6.99E-03 XCOMP8.X -2.700 XCOMP2.X
-6.99E-03
XAC.XP3.      AC.XP1.3      AC.XP1.3      AC.2
3

XCOMP2.X -6.99E-03 XCOMP1.X -6.99E-03 XCOMP6.X -5.99E-03
XCOMP10. -1.000
AC.4      AC.XP3.3      AC.XP3.3      XAC.2

XCOMP10. -1.000 XCOMP3.3 -6.99E-03 XCOMP3.4 -6.99E-06 XCOMP1.X
-6.99E-03
XAC.4      AC.2

XCOMP1.X -6.99E-03 XCOMP2.X -6.99E-03 XCOMP7.X -3.000 XCOMP9.X
-1.900
AC.4      AC.XP1.3      AC.XP1.3      AC.2

```

XCOMP9.X -1.900 XCOMP6.3 -5.99E-03 XCOMP6.4 -5.99E-06 XCOMP10.
 -1.000
 AC.4 XAC.XP1.

3

XCOMP5.X -6.99E-03 XCOMP9.3 -1.900 XCOMP9.4 -1.900 XCOMP8.X
 -2.700
 AC.XP3.3 AC.2

XCOMP8.X -2.700 XCOMP1.X -6.99E-03 XCOMP6.X -5.99E-03 XCOMP1.3
 -6.99E-03
 AC.4 AC.XP1.3 AC.XP1.3

XCOMP1.4 -6.99E-06 XCOMP7.X -3.000 XCOMP7.X -3.000 10
 4.000
 AC.2 AC.4

VOLTAGE SOURCE CURRENTS

NAME	CURRENT
VP10	0.000
VP9	0.000
VP8	0.000
VP7	0.000
VP6	0.000
VP5	0.000
VP3	0.000
VP2	0.000
VP1	0.000

TOTAL POWER DISSIPATION 0.000 WATTS

=====
 =====
 =====
 = VOLTAGE-CONTROLLED CURRENT SOURCES =
 =====
 =====
 =====

NAME	XCOMP1.X	XCOMP2.X	XCOMP3.X	XCOMP5.X	XCOMP6.X	XCOMP7.X	XCOMP8.X
DC.GOUT	DC.GOUT	DC.GOUT	DC.GOUT	DC.GOUT	DC.GOUT	DC.GOUT	DC.GOUT
I-SOURCE	-2.05E-03	-2.05E-03	-2.05E-03	-2.05E-03	-2.05E-03	-4.18E-03	-4.18E-03

NAME	XCOMP9.X	XCOMP10.
DC.GOUT	XDC.GOUT	
I-SOURCE	-4.16E-03	-4.15E-03

```

*****
*****
*
*           NODE CAPACITANCE TABLE           *
*           TEMPERATURE = 27.000             *
* RUN DATE = 12/17/91      RUN NUMBER 1      RUN TIME = 13:52:13
*
*****
*****

```

NODE	CAPACITANCE	NODE VOLTAGE
0	18.000	0.000
XCOMP9.X	0.000	-1.900
AC.XP3.3		
1	2.62E-11	-2.000
2	2.62E-11	-2.000
3	2.62E-11	-2.000
4	5.26E-11	5.000
5	2.62E-11	-2.000
6	2.62E-11	-1.000
7	2.62E-11	2.000
8	2.62E-11	2.300
9	2.62E-11	3.100
XCOMP6.X	1.000	-5.99E-03
AC.2		
XCOMP6.X	1.000	-5.99E-03
AC.4		
XCOMP5.X	0.000	-6.99E-03
AC.XP1.3		
XCOMP7.3	1.00E-12	-3.000
XCOMP7.4	0.000	-3.000
XCOMP5.X	1.000	-6.99E-03
AC.2		
XCOMP5.X	1.000	-6.99E-03
AC.4		
XCOMP3.X	0.000	-6.99E-03
AC.XP3.3		
XCOMP8.X	0.000	-2.700
AC.XP3.3		
XCOMP2.3	1.00E-12	-6.99E-03
XCOMP2.4	0.000	-6.99E-06
XCOMP9.X	0.000	-1.900
AC.XP1.3		
XCOMP10.	1.00E-12	-1.000
3		
XCOMP10.	0.000	-1.000
4		
XCOMP5.3	1.00E-12	-6.99E-03
XCOMP5.4	0.000	-6.99E-06
XCOMP2.X	0.000	-6.99E-03
AC.XP3.3		
XCOMP7.X	0.000	-3.000
AC.XP3.3		
XCOMP3.X	1.000	-6.99E-03
AC.2		

XCOMP3.X	1.000	-6.99E-03
AC.4		
XCOMP8.3	1.00E-12	-2.700
XCOMP8.4	0.000	-2.700
XCOMP10.	0.000	-1.000
XAC.XP3.		
3		
XCOMP3.X	0.000	-6.99E-03
AC.XP1.3		
XCOMP8.X	0.000	-2.700
AC.XP1.3		
XCOMP2.X	1.000	-6.99E-03
AC.2		
XCOMP2.X	1.000	-6.99E-03
AC.4		
XCOMP1.X	0.000	-6.99E-03
AC.XP3.3		
XCOMP6.X	0.000	-5.99E-03
AC.XP3.3		
XCOMP10.	1.000	-1.000
XAC.2		
XCOMP10.	1.000	-1.000
XAC.4		
XCOMP3.3	1.00E-12	-6.99E-03
XCOMP3.4	0.000	-6.99E-06
XCOMP1.X	1.000	-6.99E-03
AC.2		
XCOMP1.X	1.000	-6.99E-03
AC.4		
XCOMP2.X	0.000	-6.99E-03
AC.XP1.3		
XCOMP7.X	0.000	-3.000
AC.XP1.3		
XCOMP9.X	1.000	-1.900
AC.2		
XCOMP9.X	1.000	-1.900
AC.4		
XCOMP6.3	1.00E-12	-5.99E-03
XCOMP6.4	0.000	-5.99E-06
XCOMP10.	0.000	-1.000
XAC.XP1.		
3		
XCOMP5.X	0.000	-6.99E-03
AC.XP3.3		
XCOMP9.3	1.00E-12	-1.900
XCOMP9.4	0.000	-1.900
XCOMP8.X	1.000	-2.700
AC.2		
XCOMP8.X	1.000	-2.700
AC.4		
XCOMP1.X	0.000	-6.99E-03
AC.XP1.3		
XCOMP6.X	0.000	-5.99E-03
AC.XP1.3		

XCOMP1.3 1.00E-12 -6.99E-03
 XCOMP1.4 0.000 -6.99E-06
 XCOMP7.X 1.000 -3.000
 AC.2
 XCOMP7.X 1.000 -3.000
 AC.4
 10 2.62E-11 4.000

 * JOB STATISTICS SUMMARY *

Global Signals = 0
 Subckt Signals = 0
 Total Signals = 0

Number Equations = 110
 Number Non-Zeros (Before) = 245
 Number Non-Zeros (After) = 333
 Percent Zeros = 97.25
 Number Matrix Operations = 544
 LVL COD = 1

RERUNS = 0
 TEMPERATURES = 1

Number Resistors = 36
 Number Capacitors = 54
 Number Inductors = 0
 Number Voltage Sources = 9
 Number Current Sources = 0
 Number Dependent Sources = 45
 Number Diodes = 0
 Number BJTs = 0
 Number JFETs = 0
 Number MOSFETs = 0
 Total number of elements = 144

Number Diode models = 0
 Number BJT models = 0
 Number JFET models = 0
 Number MOS models = 0

DC SOLUTION ANALYSIS:

Number of Iterations = 8
 Load Time = 0.00 SEC
 Solve Time = 0.00 SEC
 Analysis Time = 0.37 SEC

TRANSIENT ANALYSIS:

Number of Transient time points = 10000001
 Number of Calculated time points = 109
 Number of Rejected time points = 10
 Number of Iterations = 10324

Load Time = 0.00 SEC
Solve Time = 0.00 SEC
Truncation Time = 0.00 SEC
Analysis Time = 156.80 SEC
METHOD = 0

MEMORY USAGE:

Memory used for Devices = 0.05 MB
Memory used for Matrix = 0.03 MB
Memory used for RHS = 0.02 MB
Total Memory used in ANALYSIS = 0.15 MB
Total Memory used in OUTPUT = 0.03 MB

RUN (CPU) TIMES: user + sys
Topology Check = 0.00 SEC
Matrix Setup = 0.03 SEC
Re-order = 0.12 SEC
READIN = 9.28 + 0.43 = 9.72 SEC
SETUP = 0.43 + 0.40 = 0.83 SEC
ANALYSIS = 157.40 + 18.60 = 176.00 SEC
OUTPUT = 0.12 + 0.10 = 0.22 SEC

TOTAL = 167.33 + 19.87 = 187.20 SEC

0 Error(s).
0 Warning(s).

JOB CONCLUDED

APPENDIX G

SOURCE CODE LISTING OF C PROGRAM FOR
RUNGE-KUTTA SIMULATION


```

/*****
Program History:
    Started 03/18/91 under name RK3 for Runge Kutta simulation of a
        third order system.
    03/20/91:  --reorganized to more functional form including
                comments.
                --added nonlinear function at the output of three
                pole block.
    03/27/91:  --changed to matrix based code
                --generalized RK section for an array of units
                --added median net connectivity-oscillates.
    03/29/91:  --rewrote RK section to correct simulation inaccuracy
    03/30/91:  --found conditions for oscillation with pure
                integration on the output (that is, the units have
                infinite output impedance)
    04/01/91:  --added finite output impedance, confirmed that the
                the system will work for a 5 sample window with an
                eight bit resolution (simulation gave 15 bits of
                accuracy)
    11/15/91:  --moved to apollo system hoping for quicker simulations
*****/
#include <stdio.h>
#include "cadlib.h"
#include <strings.h>
#include <math.h>
#define POLE1 (double) 3E+03 /* negative real axis coordinate of pole #1 */
#define POLE2 (double) 9000 /* negative real axis coordinate of pole #2 */
#define POLE3 (double) 10000 /* negative real axis coordinate of pole #3 */
#define AV (double) 1E-04 /* gain of processing element (act. transcond)*/
#define RAIL (double) 1e-05 /* limiting rail of processing element */
#define CAP (double) 1.8E-06 /* value of capacitance to be charged */
#define ROUT (double) 90E+06 /* output impedance of units */
#define IERR (double) 1E-06 /* output current error */

#define TIMESTEP (double) .00001 /* simulation timestep */
#define PIXELS 9
#define ORDER 3
#define OLD 0
#define DOT 1
#define DOT_TEMP 2
#define NEW 3
#define NEW_TEMP 4
#define DELTA 5
/* function declarations */
void runge_kutta(); /* 2nd order RK simulation update */
void init(); /* initialize simulation */
void get_inputs(); /* get the input pixels */
void main();
/* global variable declarations */
double pole[ORDER], t, t100;
double x[PIXELS], z[PIXELS][ORDER][5], x_old[PIXELS];
double y[PIXELS], y_limit, pix_out[6];
int i,j;
void main()
{
    init(); /* initialize the simulation */
    get_inputs(); /* get the input pixels for the filter */
    do{ /* simulate the system */
        runge_kutta(); /* evaluate the units at the next time step */
        t++; /* increment the time variable */
        j++; /* increment the divide-by-ten variable */
        if (j == 100){
            printf(" + %f\t%f\n",pix_out[NEW],t*TIMESTEP); /* print the current output value */
            j=0;
        } while (t*TIMESTEP < 300);
    }
}

```

```

        /* continue the main loop until the minimum number of iterations is
           exceeded or the derivative of the output falls below .001
           whichever comes last */
    }
void runge_kutta()
{
    pix_out[DOT] = 0;
    pix_out[DOT_TEMP] = 0;
    for (i = 0; i < PIXELS; i++){
        /* simulate the third order processing units */
        z[i][0][DOT] = z[i][1][OLD];
        z[i][1][DOT] = z[i][2][OLD];
        z[i][2][DOT] = -(pole[2]*z[i][2][OLD]) -(pole[1]*z[i][1][OLD])
            -(pole[0]*z[i][0][OLD]) + pole[0]*(x_old[i]-pix_out[OLD]);
        z[i][0][NEW_TEMP] = z[i][0][OLD] + TIMESTEP*z[i][0][DOT];
        z[i][1][NEW_TEMP] = z[i][1][OLD] + TIMESTEP*z[i][1][DOT];
        z[i][2][NEW_TEMP] = z[i][2][OLD] + TIMESTEP*z[i][2][DOT];
        y[i] = z[i][0][NEW_TEMP];
        if (fabs(y[i]) < y_limit) pix_out[DOT_TEMP] += AV*y[i];
    }
    /* if the processing element
       is in the linear region,
       output an amplified signal */
    else pix_out[DOT_TEMP] += (y[i]/fabs(y[i]))*RAIL;
    /* otherwise, return a signal
       clamped at a rail. */

    pix_out[DOT_TEMP] -= pix_out[OLD] / ROUT;
    pix_out[DOT_TEMP] += IERR / PIXELS;
    pix_out[NEW_TEMP] = pix_out[OLD] + TIMESTEP*(pix_out[DOT_TEMP]/CAP);
    /* above integrates the outputs of the comparators */
    for (i = 0; i < PIXELS; i++){
        z[i][0][DOT_TEMP] = z[i][1][NEW_TEMP];
        z[i][1][DOT_TEMP] = z[i][2][NEW_TEMP];
        z[i][2][DOT_TEMP] = -(pole[2]*z[i][2][NEW_TEMP])
            -(pole[1]*z[i][1][NEW_TEMP])
            -(pole[0]*z[i][0][NEW_TEMP])
            + pole[0]*(x[i]-pix_out[NEW_TEMP]);
        z[i][0][NEW] = z[i][0][OLD] + (TIMESTEP/2)*(z[i][0][DOT] + z[i][0][DOT_TEMP]);
        z[i][1][NEW] = z[i][1][OLD] + (TIMESTEP/2)*(z[i][1][DOT] + z[i][1][DOT_TEMP]);
        z[i][2][NEW] = z[i][2][OLD] + (TIMESTEP/2)*(z[i][2][DOT] + z[i][2][DOT_TEMP]);
        y[i] = z[i][0][NEW];
        z[i][0][OLD] = z[i][0][NEW];
        z[i][1][OLD] = z[i][1][NEW];
        z[i][2][OLD] = z[i][2][NEW];
        x_old[i] = x[i];
        if (fabs(y[i]) < y_limit) pix_out[DOT] += AV*y[i];
    }
    /* if the processing element
       is in the linear region,
       output an amplified signal */
    else pix_out[DOT] += (y[i]/fabs(y[i]))*RAIL;
    /* otherwise, return a signal
       clamped at a rail. */

    pix_out[DOT] -= pix_out[NEW_TEMP] / ROUT;
    pix_out[DOT] += IERR/PIXELS;
    }
    pix_out[DELTA] = (TIMESTEP/2)*((pix_out[DOT_TEMP] + pix_out[DOT])/CAP);
    pix_out[NEW] += pix_out[DELTA];
    pix_out[OLD] = pix_out[NEW];
} /* end of function runge_kutta */

```

```
void init()
{
t = 0;
for ( i = 0; i < PIXELS; i++ ){
y[i] = 0;
z[i][0][OLD] = y[i];
z[i][1][OLD] = 0;
z[i][2][OLD] = 0;
z[i][0][DOT_TEMP] = 0;
z[i][1][DOT_TEMP] = 0;
z[i][2][DOT_TEMP] = 0;
x[i] = 5;
x_old[i] = 0;
}
pole[0] = POLE1*POLE2*POLE3;
pole[1] = POLE1*POLE2 + POLE2*POLE3 + POLE1*POLE3;
pole[2] = POLE1 + POLE2 + POLE3;
printf("%f\n%f\n%f\n",pole[0],pole[1],pole[2]);
pix_out[DOT] = 0;
y_limit = ((double) RAIL)/((double) AV);
j = 0;
} /* end of function init() */
void get_inputs()
{
    for ( i = 0; i < PIXELS; i++ ){
        scanf("%f",&(x[i]));
        x_old[i] = x[i];
    }
} /* end of function get_inputs */
```

VITA

John P. Wagnon

Candidate for the Degree of

Master of Science

**Thesis: A PROPOSAL FOR AN ANALOG CMOS MEDIAN FILTER SYSTEM
BASED ON NEURAL NETWORK ARCHITECTURAL PRINCIPLES**

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Modesto, California, April 13, 1966, the son of Paul H. and Ramona G. Wagnon. Married Andrea Helen Brune, July 7, 1990, Wichita Mountains of Oklahoma.

Education: Graduated from Duncan Senior High School, Duncan, Oklahoma, in May, 1984; received Bachelor of Science Degree in Electrical Engineering from Oklahoma State University in December, 1988; received Master of Science Degree in Electrical Engineering from Oklahoma State University in May, 1992.

Professional Experience: Teaching Assistant, Department of Electrical and Computer Engineering, Oklahoma State University, January 1988 to January 1991; Summer Graduate Fellow, Rome Air Development Center, Rome, New York, Summer 1989; CAD Specialist in Electrical Design, Texas Instruments Semiconductor, Sugar Land, Texas, since February 1991.