A VLSI Analog Computer / Math Co-processor for a Digital Computer

Glenn Edward Russell Cowan

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2005

Abstract
A VLSI Analog Computer / Math Co-processor for a Digital Computer

Glenn Edward Russell Cowan

This dissertation investigates the utility of a programmable VLSI analog computer for the solution of differential equations. To investigate the capability of analog computing in a modern context, a large VLSI circuit (100 mm^2) was designed and fabricated in a standard mixed-signal CMOS technology. It contains 416 analog functional blocks, switches for their interconnection, and circuitry for the system's program and control. This chip is controlled and programmed by a PC via a data acquisition card. This arrangement has been used to solve differential equations with acceptable accuracy, as much as 400x faster than a modern workstation.

The utility of a VLSI analog computer has been demonstrated by solving stochastic differential equations, partial differential equations, and ordinary differential equations (ODEs). Additionally, techniques for using the digital computer to refine the solution from the analog computer are presented. Solutions from the analog computer have been used to accelerate a digital computer's solution of the periodic steady

state of an ODE by more than an order of magnitude.

An analysis has been done showing that the analog computer dissipates 0.02 % to 1 % of the energy of a digital general purpose microprocessor and about 2 % to 20 % of the energy of a digital signal processor, when solving the same differential equation.

Contents

List of Figures

List of Tables

Acknowledgments

This dissertation was made possible by my advisor, Yannis Tsividis. His unwavering enthusiasm for the project has been infectious and his steady confidence in my abilities, though frightening in the early years, lays the foundation for the research career ahead of me. He is also compelling proof that one need not choose between a successful research career and a fulfilling teaching career.

I am grateful for the contribution of the other members of my dissertation committee: Drs. David Keyes, Bob Melville, Ken Shepard, and Charles Zukowski. Your comments and time spent reading this work have been critical to its improvement. Special thanks are due to Bob Melville who has helped greatly in making contacts with mathematicians and in finding applications for the analog computer. To Ken Shepard, through the four courses he taught me, I owe my ability to "talk the talk" and "walk the walk" of digital IC design.

Professor Yannis Kevrekidis's assistance with stochastic differential equations has been invaluable. His enthusiasm and generosity with his time are greatly appreciated.

During my time in 422 Shapiro I have been fortunate to meet and work along side many great students, namely former students Shanthi Pavan, Greg Ionis, Nagi Krishnapura, Dandan Li, Yorgos Palaskas and Sanjeev Ranganathan and all of the current students. Greg, Sanjeev, Tuku Chatterjee, George Patounakis and Nebojsa Stanic have cajoled many a racalcitrant computer into behaving properly. Babak Soltanian and Yorgos have been a delight with whom to share the confines of 422G. I am grateful to Mehmet Ozgun, Tuku, and George for reading drafts of parts of this thesis.

Thanks are owed to the Department of Electrical Engineering's office staff. The late Marlene Mansfield's no-nonsense style, her effectiveness (and "affectiveness") and her commitment to her students were greatly appreciated and are missed. Betsy Arias, John Baldi, Stacey Miller, Jim Mitchell, Bilha Njuguna, Lourdes de la Paz, Elsa Sanchez, Jodi Schneider, and Azlyn Smith have kindly helped on numerous occasions to solve problems with my registration or other issues that are frequently of my own causing.

The last 5 years would not have been spent at Columbia had it not been for the influence of Professor Ajoy Opal from the University of Waterloo, my 4^{th} year project advisor. His suggestion that I apply to Columbia was the push that got the ball rolling. I am glad to have had his support and I thank Professors Andrew Heunis, George Kesidis and Tajinder Manku for their advice over the years at UW.

I am indebted to my parents for their love and support over the years. I am extremely grateful to Uncle Norman Cowan (physics teacher extraordinaire) for all the impromptu physics lessons over the years. He demonstrated the importance of skepticism, intellectual curiosity, and precise language when discussing science. Thanks are also due to my brother, Michael, an often quoted source of theories on less technical matters.

There are many others who have helped steer this research and my academic interests before coming to Columbia. It is only by accident that they are omitted from these acknowledgements.

Chapter 1

Introduction

1.1 Motivation

Analog computers of the 1960s were physically large, tedious to program and required significant user expertise. However, once programmed, they rapidly solved a variety of mathematical problems, notably differential equations, without time-domain discretization artifacts, albeit with only moderate accuracy [1]. Analog computers were superseded by digital ones long ago; the technical community has hardly considered what advantages modern VLSI techniques could impart to analog computers.

Digital computers can solve differential equations with very high accuracy. However, they may suffer from a variety of convergence problems. Further, some simulations may take a long time. This can preclude repeatedly solving the equations as part of design optimization. In real-time control applications, long simulation time may require that a simpler and possibly inferior model be simulated.

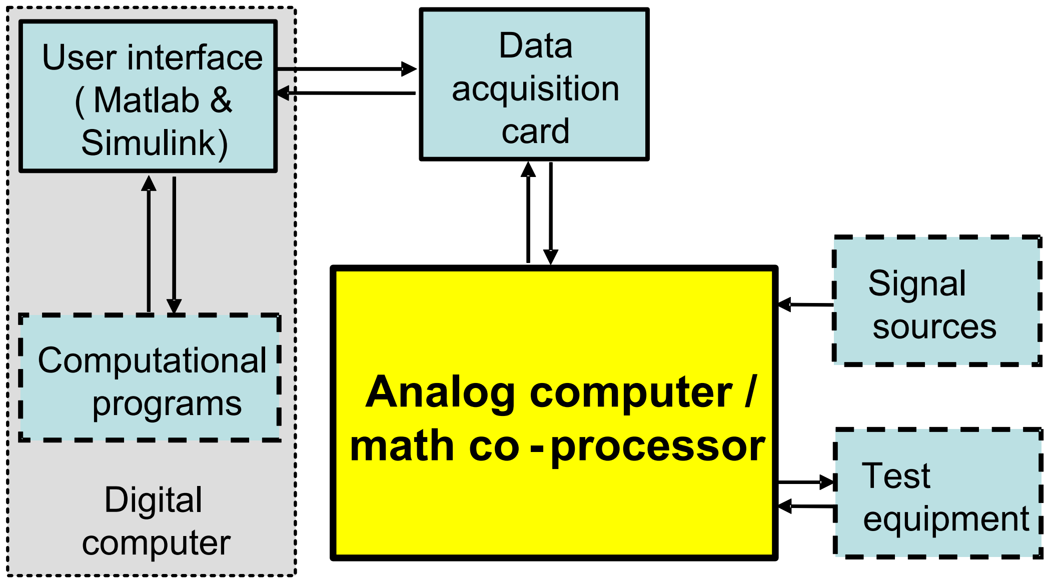

Figure 1.1: Analog computation environment.

This thesis reports on the results of a first attempt to investigate whether analog computers could be revived in a modern VLSI context [2], with significant advantages as complements to digital computers. A single-chip VLSI analog computer (AC) is presented, capable of handling large-order problems (e.g., nonlinear differential systems of up to 80 first-order equations), specifically meant to operate in a symbiotic environment with a digital computer (Fig. 1.1), complementing the latter and acting as a co-processor to it. This combination makes possible several features:

- Tight coupling to the digital computer through an A/D-D/A interface which facilitates a pre-computation calibration process.
- Digital programmability through a standard on-screen user interface.
- Real-time simulations and real-time observation of the effects of mathematical

parameter changes.

- Fast approximate solutions to difficult mathematical problems, sometimes much faster than is possible with a digital computer, and with guaranteed convergence to a physical solution.
- Capability of passing an approximate solution, as a first guess to a digital computer, thus speeding up a numerical solution.
- Interfacing to measuring instruments.

1.2 History of Analog Computation

Historically analog computers primarily solved the differential equations necessary for system simulation. Analog circuits can solve ordinary differential equations (ODEs) written in the form $\dot{x} = f(x, u, t)$, where x is a vector of state variables of length n, u is a vector of inputs of length m, f is a vector of possibly nonlinear functions of length n, and t is time. To solve this, an AC needs n integrators, m inputs and sufficient circuitry to implement f, or an adequate approximation to it. In addition, techniques exist (e.g. method of lines [3]) for converting partial differential equations (PDEs) to ODEs of the above form.

Interest in analog computers decreased in the 1960s and 1970s as digital computers became more advanced. One of the few remaining applications of analog computers today is a back-up system in the Soyuz spacecraft.

Some of the closest examples of modern (continuous-time) special-purpose analog computation are some efforts in the area of reconfigurable analog hardware [4], sometimes referred to field programmable analog arrays (FPAAs). These have gained the interest of researchers, and some products have been released, though they typically are for linear filtering applications. While implementing a filter is synonymous with implementing a differential equation, a progammable filter is less general than what is proposed here.

Large analog VLSI systems have been designed to solve many problems by emulating neural systems [5]. These are special purpose circuits capable of performing a particular function and do not solve differential equations.

For the purpose of solving differential equations, some discrete-time (DT) devices have been designed [6], [7]. These, however, may still add artifacts stemming from the discretization of time that occurs when continuous-time ODEs are mapped to the discrete-time equations that are solved by the DT computer.

While no general purpose VLSI analog computers exist today, researchers have built custom analog computers to investigate solitons [8] and nonlinear oscillators [9].

The idea of coupling an analog computer to a digital computer has been studied in detail over the years. One of the most interesting ways of pairing the two types of computers that is reported in the literature, iteratively uses an analog computer to compute a correction term in a root finding algorithm [10].

1.3 Overview of this Thesis

Chapter 2 describes how analog and digital computers each solve differential equations, and their respective strengths and weaknesses. The design of a large VLSI analog computer is outlined in Chapter 3 and the hybrid computation environment in which it is used is described in Chapter 4. The performance of the individual circuit blocks is summarized in Chapter 5. Chapter 6 gives representative examples of differential equations solved by the analog computer and Chapter 7 describes how the analog computer's solution can be used and refined by a digital computer in a way that speeds up the digital computer's solution. Chapter 8 compares this analog computer to a digital computer in terms of power consumption and computation speed. Chapter 9 gives some suggestions for future work in this field.

Chapter 2

Review of Digital and Analog Computer Techniques

2.1 Digital Computation

2.1.1 Solving Differential Equations on a Digital Computer

Solution to Initial Value Problems, Computed Point by Point

The functions that are solutions to ODEs are continuous in time; however, digital computers calculate the solution at a series of time points. The differential equations are mapped to a set of difference equations and solved. How this mapping is done has important consequences on the speed of the digital computer's solution and the required spacing of the time steps, and determines some of the stability properties of the solution.

For a more complete discussion of numerical routines, the reader is directed to [11], [12] and [13]. What follows is an overview of some of the basic methods. Consider the following first-order ODE:

$$\dot{x} = f(x, t) \tag{2.1}$$

$$y = g(x, u, t) \tag{2.2}$$

and suppose that its solution is desired over the interval of time (t_0 to t_f) beginning with initial condition $x(t_0) = x_0$. This discussion assumes that the functions g are algebraic equations whose values are easily computed once x is known. As such, the majority of the computational effort in solving for y is spent solving for x and the discussion of equations in this form will center around the solution of x. The solution to Eq. 2.1 at a time, t_1 near t_0, can be calculated from x_0. The choice of the time step from t_0 to t_1 is based on a number of error estimations, and is beyond the scope of this thesis. Based on the derivative of x, $x_1 = x(t_1)$ can be estimated by assuming that x follows a straight line from x_0 to x_1. That is:

$$x_1 = x_0 + m(t_1 - t_0) \tag{2.3}$$

Possible values for the slope m in Eq. 2.3, are $f(x_0, t_0)$, $f(x_1, t_1)$ or some combination of the two. Consider the case of the former:

$$x_1 = x_0 + f(x_0, t_0)(t_1 - t_0) \tag{2.4}$$

Since x_1 is the only unknown, and it is isolated on the left side of the equation, it can be calculated explicitly and very quickly. Routines that are explicit are referred to as predictor routines. Examples of predictor routines include the method above, known as Forward Euler and Heun's method which are one-step methods, since the solution at only one time step is used to calculate the solution at the next time step. Other predictor methods include various Runge-Kutta methods which use the solution at several time steps and are called multi-step methods.

Generally speaking, taking smaller time steps leads to higher accuracy. However, as the time step is lengthened, the degradation in accuracy may be far from graceful for a method such as Forward Euler. Taking time steps longer than the shortest time constant in the system can lead to terribly inaccurate results due to numerical instability and floating point rounding error, even if the dynamics associated with that time constant have long since decayed. That is, fast dynamics in the system require short time steps for the entire simulation. This characteristic of Forward Euler makes it inappropriate for the simulation of many types of systems, most notably, stiff systems. A system is said to be stiff if it contains dynamics that have widely differing time constants.

On the other hand, if $f(x_1, t_1)$ is used for the slope m in Eq. 2.3, larger time steps can be used. However, the problem is more difficult to solve:

$$x_1 = x_0 + f(x_1, t_1)(t_1 - t_0) \tag{2.5}$$

In the above, $f(x_1, t_1)$ depends on x_1, which is the unknown. If f is a nonlinear function, it may not be possible to solve for x_1 explicitly, and hence this type of method is said to be implicit (and a corrector method). The solution for x_1 is typically found using some sort of iterative root finding scheme such as Newton-Raphson iterations.

These iterations are not guaranteed to converge if they begin with a starting value for x_1 that is separated from the correct value of x_1 by a local extremum. The number of iterations required to reach a given level of convergence will be smaller if the initial value for x_1 is closer to its correct value. As such, numerical routines often use a combination of predictor and corrector methods. x_1 is predicted by a method such as Forward Euler, and then refined (corrected) by a method such as Backward Euler. Other examples of corrector/implicit routines are the trapezoidal method, and the gear2 method.

Each method has its own unique characteristics. The trapezoidal method can give rise to trapezoidal ringing, a phenomenon in which the state variables exhibit false oscillatory behaviour when the system is excited by a step function. For all of these methods, a continuous-time (CT) system is mapped to a discrete-time (DT) system. Ideally the stability characteristics of the CT and DT systems would be the same as one another. That is, a CT system exhibiting bounded-input bounded-output (BIBO) stability would be mapped to a DT system exhibiting BIBO stability, and a CT system that did not exhibit BIBO stability would be mapped to a DT system that also did not exhibit BIBO stability. This is very important as one of the most important reasons for doing a simulation may be to determine if the system is stable.

The above methods also apply to cases in which x and f are vectors. Little is changed in the predictor method, but the corrector methods require even more effort since Newton-Raphson iterations are somewhat more complicated when f is a vector of functions, as they may involve large-scale linear algebra.

Waveform Relaxation

In the methods discussed in the previous section, the solution for every state variable at a particular time step is calculated to a satisfactory accuracy before the solution at the next time step is calculated. The solution, up to and including the current time step, is known, and the solution at the next time step is an unknown. An alternative to this technique is called waveform relaxation [14]. In this technique, the solution over a time interval (t_0 to t_f) for one state variable (say x_i) is calculated, while the other state variables are treated as knowns, and then the solution for the next state variable (say x_{i+1}) is calculated over the same time interval, assuming the others are known. This is repeated until the solution for all state variables has been calculated, at which point the process repeats, recalculating the state variables, one at a time, over the solution interval, assuming that the other variables are known. When acceptable convergence has been reached, the simulation stops.

This approach is used in large systems with only weak coupling between state variables. In the case of stronger coupling a larger number of cycles is needed to achieve a particular level of convergence.

Periodic Steady State

Engineers are frequently interested in the steady-state response of a nonlinear system to a periodic input. The system has reached this so-called periodic steady state (PSS) if each state variable x_i satisfies:

$$x_i(t+T) = x_i(t), \text{ all } t \tag{2.6}$$

where T is the period of the state variables. T frequently is also the period of the system's input, if it has any inputs.

The condition in Eq. 2.6 allows us to consider the solution to the ODE over only one period, which is discretized into n points. The derivatives at the last point will depend on the value at the first point and vice versa, stemming from Eq. 2.6. If the system has m state variables, over the n points there are a total of $m \times n$ unknowns. Some periodic steady-state solvers perform Newton-Raphson iterations on this vector of $m \times n$ unknowns [15].

Variations of this technique, rather than considering n unknown time points, consider $n/2$ unknown complex-frequency components, which are related to the n unknown time-points by the discrete-Fourier transform. Because the frequency components are complex numbers, the frequency domain technique has the same number of unknowns as the time domain approach. Frequency domain techniques can be extended to multi-tone quasi-periodic excitations which are very useful in predicting distortion in communication systems.

2.1.2 Strengths and Weaknesses of the Digital Approach

The following is a discussion of some of the strengths and weaknesses of the digital approach.

Accuracy: Digital computers can achieve very high accuracy. However, to reap the benefits of the large number of bits with which numbers are represented, in some cases the solution must be computed at very finely-spaced time points. This can slow down the simulation.

Dynamic Range: Digital computers have very large dynamic range, owing to the floating point representation of numbers.

Reconfigurabulity: Digital computers are easily reconfigured, and new functions are easily approximated with series expansions or other techniques.

Requisite User Expertise: While users must become familiar with the software running on the computer, they can be successful with limited understanding of system theory and limited knowledge of the inner workings of the computer.

Speed: Digital computers perform individual floating point operations very quickly. However, the large number of computations needed to solve large systems of differential equations, or those equations which require computation at a large number of time-points, may result in long simulations.

Cost: Digital computers are relatively inexpensive.

Numerical Artifacts: Because continuous-time, continuous-valued problems are mapped to discrete-time, discrete-valued equations when solved on a digital computer, several problems can ensue, namely non-convergence, changes in stability properties and other artifacts stemming from the approximation of the derivatives. These problems primarily stem from the mapping to discrete-time, and not because the digital computer represents numbers with discrete values.

2.2 Analog Computation

2.2.1 Solving Differential Equations on an Analog Computer

Basics: Continuous-time ordinary differential equations (ODEs) are most easily mapped to an analog system when they are written in state-space form:

$$\dot{x} = f(x, u, t) \tag{2.7}$$

$$y = g(x, u, t) \tag{2.8}$$

where x is a vector of state variables of length n, u is a vector of inputs of length m, f is a vector of possibly nonlinear functions of length n, y is a vector of outputs of length p, and g is a vector of possibly nonlinear functions of length p. To solve this, an analog computer needs n integrators, m inputs, p outputs and sufficient circuitry to implement f and g, or adequate approximations of them. This set of differential equations is realized by denoting the output of the i^{th} integrator as the i^{th} state variable (x_i). Therefore, the input to the i^{th} integrator is $\dot{x}_i$. Circuitry necessary to implement the i^{th} function in f, (f_i), is used, the output of which is applied to the input of the i^{th} integrator. The circuits necessary to implement the i^{th} function in g, (g_i), are used to generate the outputs of the system.

An example of a very simple ODE is a single-input, single-output, first-order, linear time-invariant (LTI) system given by the following:

$$\dot{x} = \frac{1}{\tau}(-x + u) \tag{2.9}$$

$$y = x \tag{2.10}$$

Here, m, n and p are equal to 1. The result of the implementation procedure described above is shown in Fig. 2.1. The mapping of this equation to an analog computer requires an integrator, an amplifier that implements the gain $\frac{1}{\tau}$ and the means to sum signals. To find the solution to this equation, over the time interval from 0 to a

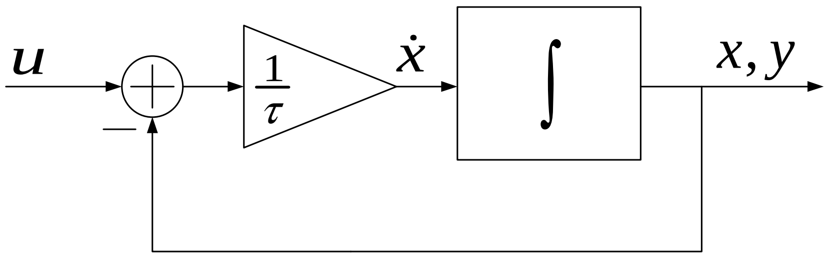

Figure 2.1: Realization of a first-order, LTI ordinary differential equation.

final time point, t_f with initial condition $x(0) = x_0$, for a particular input $u_0(t)$, the output of the integrator must be set to x_0, and $u_0(t)$ over the interval from 0 to t_f is applied to the input of the analog system. The output y, measured over the interval of time from 0 to t_f is the solution to the differential equation.

The input $u_0(t)$ can be generated by a digital computer and applied to the analog system via a digital to analog converter (DAC). Historically, the outputs were plotted directly onto paper, but in modern analog systems the outputs are typically measured by an analog-to-digital converter (ADC) and passed to a digital computer for storage and further display.

Scaling

For a system of differential equations solved on an analog computer a mapping is made between the units of the equations' variables to the units of the electrical quantities of the analog computer. This mapping, frequently called scaling, is illustrated by the following example. Assume that Eq. 2.9 and Eq. 2.10 reasonably model the temperature of a barrel of hot water placed outside on a cool day, where $x(t)$ is the average temperature of the water in the barrel at any time instant, t, x_0 is the initial temperature of the water in the barrel, $u_0(t)$ is the temperature of the outside air and τ is the time constant with which the barrel cools. For argument's sake, $x_0 = 70\ C$, $u_0(t) = 10\ C$ and $\tau = 1800\ s$. While this can be represented by the block diagram in Fig. 2.1, when it is mapped to the analog computer's circuits, some correspondence between the equation's variables (x, u) which have units [C] and the electrical quantities in the analog computer must be made, as the equation's variables will ultimately be represented by voltages or currents in the analog computer.

The temperature of the water will start at its initial temperature of 70 C and will asymptotically, with exponential decay, approach its final temperature of 10 C if $t \rightarrow \infty$. If the integrator being used is a voltage-mode circuit with a linear input and output range of ± 10 V, it would be appropriate to convert the equation's variables to the electrical quantities by scaling them by the factor $\frac{1}{10}\frac{V}{C}$. With this scaling, the output of the integrator will vary from 7 V to 1 V. Clearly, at least some prediction of the bounds of the solution needs to be made.

Most analog computers have some means of detecting if signals go out of range, allowing the user to rescale the equations, shrinking their range, and resimulate the system. This detection is important since a user may not be able to predict the extremes of variables in the differential equation before the equation is simulated. To guard against the variables going out of range, one could scale the equations so that they remain very small. However, this is unwise since the signals will be closer to the noise in the system, thereby decreasing the accuracy of the solution.

A more subtle form of scaling is needed when the variables in the equation do not change significantly relative to their average value. For example, consider the case for the same ODE, where the initial temperature is 9 C and the outside temperature is 8.9 C. There is no need to scale the variables for them to fit within the limits of the circuit. That is, the trivial ratio of $\frac{1}{1}\frac{V}{C}$ could map temperature to voltage. However, the small change in x from 9 V to 8.9 V over the simulation time might lead to inaccurate results since the noise of the circuits will be larger relative to the 0.1 V change in x than it was for the 8 V change in the earlier example.

For linear systems, the solution can easily be split into a constant part and a time-varying part, allowing the time-varying part to be expanded into a larger part of the range of the circuits. That is: $x = X_{DC} + x_{var}$ where the subscript DC denotes the constant part of the variable and var denotes the time-varying part. Likewise, $y = Y_{DC} + y_{var}$ and $u = U_{DC} + u_{var}$. We can set $Y_{DC} = X_{DC}$ and $y_{var} = x_{var}$. The differential equation becomes: $\dot{x}_{var} = \frac{1}{\tau}(-x_{var} + u_{var})$. $X_{DC} = Y_{DC} = U_{DC} = 9$ and $u_{var}(t) = -0.1$ V. Now the time-varying parts of u, x, and y range from 0 to -0.1 C. A mapping of $\frac{90}{1}\frac{V}{C}$ can be used meaning that the electrical variable representing

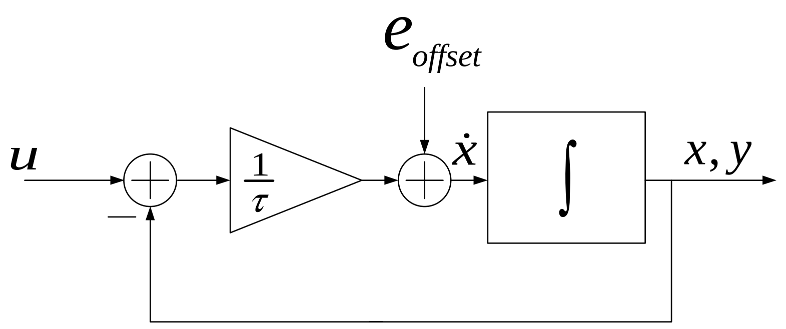

Figure 2.2: Realization of a first-order, LTI ordinary differential equation. The integrator has an input offset.

x_{var} will change by 9 V over the simulation and would be effected by noise to a much lesser degree than when no scaling is done.

While this shifting in the variables is obvious and easy to do for simple linear systems, it is much more difficult for nonlinear systems. Often, the shifting of variables in nonlinear equations modifies the equations themselves beyond simply rewriting them in terms of different variables.

In the example described by Eq. 2.9 and Eq. 2.10, when $\tau = 1800\,s$, the amplifier in Fig. 2.1 has a gain of $\frac{1}{\tau} = \frac{1}{1800} = 5.55 \times 10^{-4}$. This very low gain has several consequences:

- It may be difficult to implement such a low gain.
- The output signal of the amplifier may be small relative to the amplifier's output noise or the integrator's input noise.
- The signal applied to the input of the integrator may be very small relative to the integrator's input referred offset, leading to inaccurate results. This is because

the $y(t)$ that the analog computer computes will be the sum of the responses of the system to $u(t)$ and to $e_{offset}(t)$ where $e_{offset}(t)$ is the integrator's input-referred offset, depicted in Fig. 2.2. The DC gain from e_{offset} to y is τ, which is much larger than from u, when τ is large.

- Implementing this system with $\tau = 1800$ s means that the solution will be changing with the same, very long time constant that the actual system (barrel of water) has. The analog computer is not useful if it takes several hours to solve this simple differential equation.

To address these issues, another scaling technique is performed, frequently referred to as time scaling. Consider the system of equations in Eq. 2.7 and Eq. 2.8, with solution $y(t)$. Suppose a new set of equations, with the same set of functions for f and g is defined:

$$\dot{x}_s = Kf(x_s, u_s, t), \text{ where } u_s(t) = u(Kt) \tag{2.11}$$

$$y_s = g(x_s, u_s, t) \tag{2.12}$$

It can be shown that $y_s(t) = y(Kt)$. That is, if the input signal to every integrator in a system is multiplied by a gain of K, the solution to the new system is simply a time-scaled version of the original system, provided the inputs are time-scaled by the same factor. This property can be used to speed up a simulation, and to avoid having very small inputs to the integrators. Another aspect of time scaling stems from the fact that the integrators in analog computers usually do not implement the transfer function $\frac{1}{s}$, but implement $\frac{1}{\tau_{int}s}$ for some time constant, τ_{int}, typically much less than one second. While the term time constant normally refers to the parameter τ in a function of the form $e^{\frac{-t}{\tau}}$ it will also be used to refer to the reciprocal of the integrator's scaling factor. That is:

$$\dot{y} = \frac{1}{\tau}x \tag{2.13}$$

where y is the output of the integrator and x is the input of the integrator. τ is the time constant of the above integrator.

To time-scale a problem for simulation on an analog computer, one does the following:

- Select a K to generate a new set of equations as shown in Eqs. 2.11 and 2.12. A good choice would be one that, based on any knowledge of the system, leads to inputs to integrators of comparable size to the outputs of the integrators. For Eq. 2.10, the best choice for K, would be τ, thereby eliminating the amplifier in Fig. 2.1. Now, $y_s(t) = y(Kt)$.

- Implement the scaled equations on the analog computer. When measuring the solution from the analog computer, y_{AC}, scale its time values by the time constant of the analog computer's integrators (τ_{AC}) in the following way: $y_s(t) = y_{AC}(\frac{t}{\tau_{AC}})$. Since $y_s(t) = y(Kt)$, $y(t) = y_s(\frac{t}{K}) = y_{AC}(\frac{t}{K\tau_{AC}})$

Solving Partial Differential Equations on an Analog Computer

In many physical phenomena, functions are defined in terms of more than one variable (e.g., space and time). Frequently, rates of change are also specified for more than one variable, giving rise to partial differential equations (PDEs), which cannot immediately be solved on an analog computer in the way that ODEs can be.

In general terms, this problem can be addressed by discretizing either space or time. The latter approach, which leaves space a continuous variable, is called continuous-space, discrete-time (CSDT) while the former is called discrete-space, continuous-time (DSCT). DSCT will be described in more detail below, as it is the technique used with the analog computer described in this thesis.

Method of Lines: In the DSCT technique, also known as the method of lines, the spatial partial derivatives are approximated as finite differences, leaving a set of coupled ODEs, which can be solved in the usual fashion on the analog computer. Consider the following one-dimensional heat equation:

$$\alpha \frac{\partial^2 T}{\partial x^2} = \dot{T} \tag{2.14}$$

where $T(x,t)$ is the temperature at a point x and time t along a uniform rod oriented in the x direction. The coefficient α representing a physical property of the material is given by: $\alpha = \frac{k}{C}$, where k is the material's thermal conductivity and C is the material's specific heat capacity.

$\frac{\partial T}{\partial x}$ can be approximated by a difference in several ways, though for all, x is discretized into a set of N points, namely $x_0, x_1, x_2 ... x_{N-1}$. The spacing between discretization points may be variable or constant. We consider the constant case in which $x_i - x_{i-1} = h$. The following are three possibilities for the approximation:

$$\frac{\partial T}{\partial x} \mid_{x_i} = \frac{T_i - T_{i-1}}{h} \tag{2.15}$$

$$\frac{\partial T}{\partial x} |_{x_i} = \frac{T_{i+1} - T_i}{h} \tag{2.16}$$

$$\frac{\partial T}{\partial x} |_{x_i} = \frac{T_{i+1} - T_{i-1}}{2h} \tag{2.17}$$

Eq. 2.15 is referred to as Backward Euler, Eq. 2.16 is referred to as Forward Euler, and Eq. 2.17 is referred to as Central Differences. To approximate the second partial derivative in the heat equation, approximations must be applied twice. If Forward Euler is used to approximate the first partial derivative, Backward Euler is typically used to approximate the second partial derivative from the first, giving:

$$\frac{\partial^2 T}{\partial x^2} |_{x_i} = \frac{T_{i+1} - 2T_i + T_{i-1}}{h^2} \tag{2.18}$$

Alternatively, the second partial derivative can be approximated by applying the Central Differences approximation twice, giving:

$$\frac{\partial^2 T}{\partial x^2} |_{x_i} = \frac{T_{i+2} - 2T_i + T_{i-2}}{4h^2} \tag{2.19}$$

These two approximations have different properties that relate to the accuracy of the solution on an analog computer which will be discussed in Sect. 6.1. However, both allow for the one-dimensional heat equation to be simulated on an analog computer with $N - 2$ to N integrators, depending on the type of boundary conditions. Both techniques can also be extended to PDEs of two and three dimensions in a similar fashion. For a two-dimensional PDE with space discretized into an $N \times N$ grid, approximately N^2 integrators are needed.

The technique is called the method of lines because the problem which is defined over the (x, t) plane is solved along lines parallel to one of the axes. In the

DSCT technique, the solutions of the ODE are computed along lines in the (x, t) plane spaced equally in x and parallel to the t axis. In the CSDT technique, the solutions to the ODEs are computed along lines in the (x, t) plane equally spaced in t and parallel to the x axis.

Method of Characteristics: In this technique, known as the method of characteristics, expressions for different lines in the (x, t) plane are found which trace level curves of the solution $T(x, t)$. If the solution to the PDE at only one point in the (x, t) plane is desired, it may be possible to integrate only along one curve from an initial value, since these curves, for some PDEs, do not depend on one another and can be integrated independently.

2.2.2 Strengths and Weaknesses of the Analog Approach

For the following discussion, an attempt has been made to note which deficiencies can be, or already have been, mitigated in modern analog computers.

Accuracy: Analog computers are not very accurate. Solutions are usually accurate to no better than 0.1 % - 1 %. These errors stem from the following:

- Thermal, flicker and shot noise.
- Nonlinear transfer characteristics of blocks that should be linear.
- Other deviations from the intended input-to-output transfer characteristics of memoryless blocks. For example: imperfect multiplication of multipliers, finite output resistance of current mode circuits, or offsets of variable gain amplifiers.

- Finite DC gain of integrators.
- Input offsets of integrators.
- Finite bandwidth of memoryless blocks and non-dominant poles in the integrators.
- Granularity in the setting of coefficients such as integrator time constants or gains of variable gain amplifiers.
- Coupling between signal wires.

Dynamic Range: Dynamic range is a measure of the range of signals over which accurate processing can be guaranteed. Due to noise in the analog circuits, small signals become unduly corrupted by noise. Large signals are processed with distortion. Generally speaking, analog computers have relatively poor dynamic range, as compared to digital computers. The following VLSI techniques can be used to improve the dynamic range of an analog computer, though not to the level of a digital computer:

- Dynamic bias schemes to reduce the noise when signals are small, but also to allow for large signals.
- Class-AB operation.
- Automatic offset cancellation schemes for integrators and other blocks.

Reconfigurability: Classical analog computers were hand-wired through the plugging and unplugging of patch cords and by turning potentiometers to tune gains and time constants. This could take a significant amount of time for large systems, and required significant user expertise. This problem does not apply to VLSI analog computers, since programming operations can be controlled automatically by a digital computer that sets the states of electronic switches to control the connectivity of the blocks, and programs DACs that generate tuning signals to tune gains and time constants.

If a user wished to use a function not within a classical analog computer system, the block had to be constructed. However, in many instances, an approximation to the function is sufficient and through the use of VLSI circuits, a large amount of circuitry can be available to generate these approximations.

Requisite User Expertise: The scaling discussed in Sect. 2.2.1 normally requires user expertise to perform. However, because the interface to a modern analog computer is typically through a digital computer, much of the scaling can be automated, reducing the level of expertise that is necessary.

Speed: The speed of an analog computer is largely determined by the nominal time constant of the integrators and the interval over which a solution is desired, and not by the complexity of the system being simulated. That is, doubling the number of state variables in an ODE does not increase the simulation duration. While some digital computers make use of parallelism, every analog computer is inherently parallel.

Cost: Historically, analog computers tended to be costly owing to their large number of parts and the significant mechanical assembly needed. Also, many would be tuned at the factory, requiring the time of a technician. VLSI circuits can be inexpensive when manufactured in quantity and electronic assembly costs are low, making this no longer a drawback of analog computation.

Aliasing and Stability: There are features inherent to analog computation since the actual continuous-time, continuous-valued equations whose solution is sought are solved directly. Firstly, an analog computer will never settle to a metastable equilibrium as a digital computer may and it is less likely to suffer from non-convergence. Here, non-convergence refers to situations in which a solution exists, but the digital computer cannot find it. An example of this is the situation in which Newton-Raphson iterations fail because the starting point for the iterations was separated from the solution by local extremum. Secondly, because the system is in continuous time, there can never be aliasing, nor can there ever be artifacts introduced because time-derivatives in the differential equations are mapped to differences, as is the case with a digital computer.

The output of a modern analog computer is usually sampled and read by a digital computer. An anti-aliasing filter is needed at the input to the ADC. However, the bandwidth of the signal can be estimated by the frequencies of input signals and by the size of inputs to integrators.

Clearly, analog computation offers some advantages over digital computation and vice versa. As such, an attempt has been made to combine them in a way that best utilizes modern VLSI technology and best exploits their respective strengths. This hybrid computer system is described in Chapter 4

Chapter 3

Design of the VLSI Circuit

3.1 Chip Architecture

3.1.1 Overview

The VLSI analog computer (AC) that is the subject of this thesis is composed of 416 functional blocks, a large number of signal routing switches, memory that holds the states of the switches, memory that holds programming data for the functional blocks, and circuitry enabling the programming of the chip and the control of simulations on the chip. Computational variables are represented by differential currents (i.e. the circuits are current-in/current-out) and hence signals are added by connecting multiple signals together. Cross-coupling of a differential signal allows it to be inverted or subtracted from another signal. The chip contains the following circuits:

- 80 integrators.

- 80 variable-gain amplifiers (VGAs) / 2-input multipliers.
- 160 fanout blocks. The purpose of these blocks is noted below.
- 16 logarithms.
- 16 exponentials.
- 64 programmable nonlinear circuits.

Each programmable nonlinear circuit can be used individually to implement:

- Sign.
- Absolute value.
- Saturation.
- A programmable function that serves as a building block for generating piece-wise linear functions.

When two programmable nonlinear blocks are used together, the pair can implement:

- Minimum.
- Maximum.
- Greater than.
- Less than.

- Track.
- Track and hold.
- Sample and hold.

The chip contains 160 blocks (fanout blocks) that allow a signal to be fanned out to up to three other blocks, necessitated by the chip being current-mode.

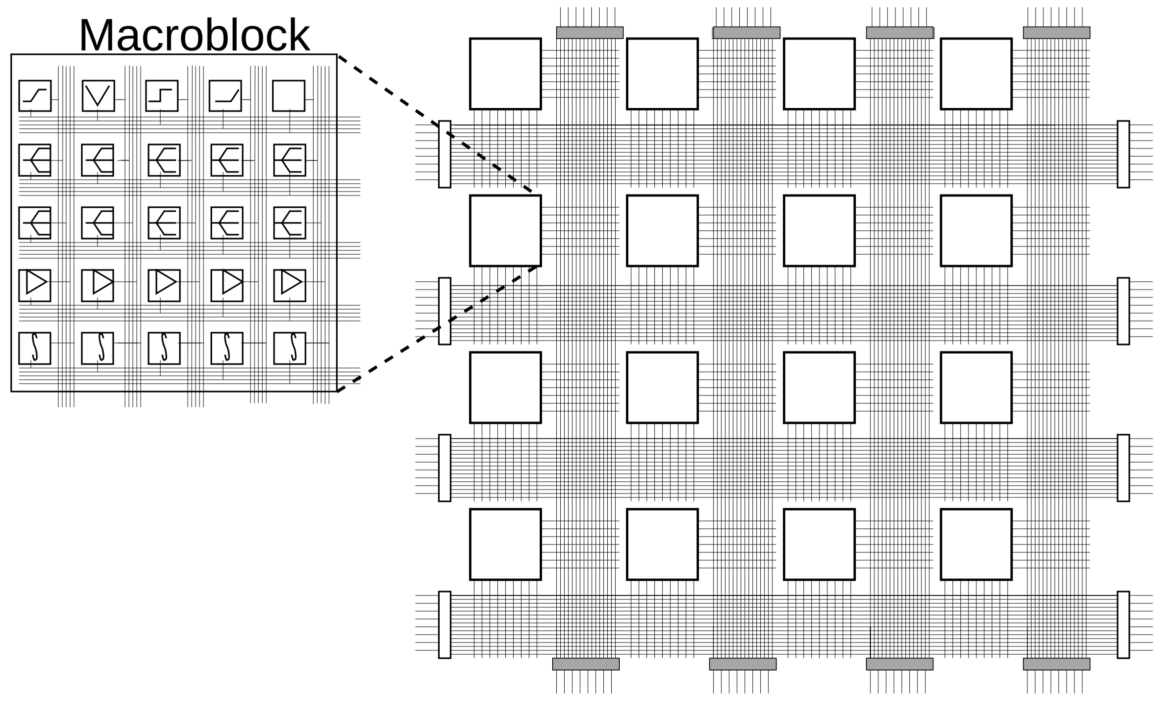

Figure 3.1: Architecture of the VLSI analog computer with expanded view of one Macroblock.

The circuits are divided into a 4 x 4 array of identical macroblocks (MBs). Fig. 3.1 shows the architecture of the chip with an expanded view of one MB.

Fig. 3.2 shows a detailed view of one Macroblock. Each block's input is connected to a wire running horizontally and each block's output is connected to a wire running vertically. These wires extend outside of the MB to allow for the connection

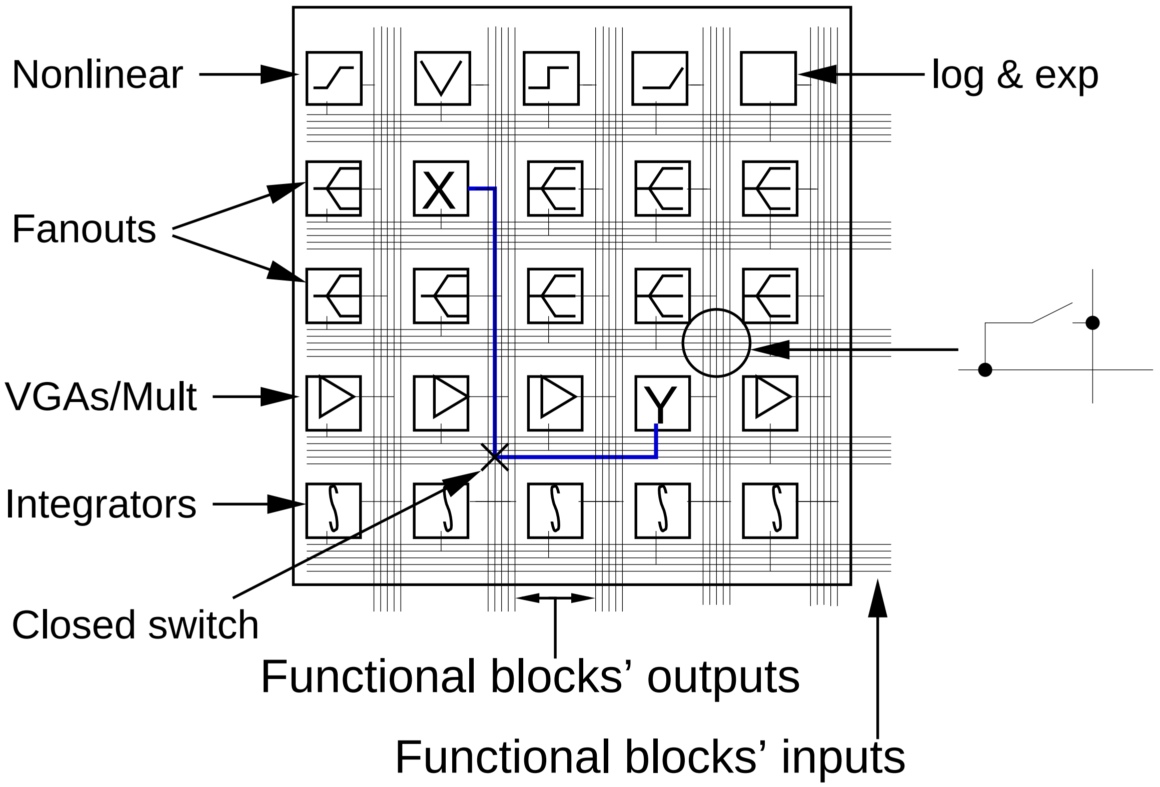

Figure 3.2: Architecture of one Macroblock within the analog computer.

between blocks in different MBs. For simplicity each block is shown with one input and one output though some blocks have more than one input or output. Each wire represents a pair of wires, carrying a differential current. There is an array of CMOS pass-transistor switches and SRAM that holds their states wherever two groups of wires cross one another. The switches can be closed to connect horizontal and vertical wires. The solid, bold line (within the expanded MB) shows how the output of block X is routed to the input of block Y.

Fig. 3.3 shows the interconnection of the Macroblocks with one another and off-chip. To route a block's output to the input of a block not in the same MB, shared, global wires are used. The dotted, bold line shows how the output of a block in MB

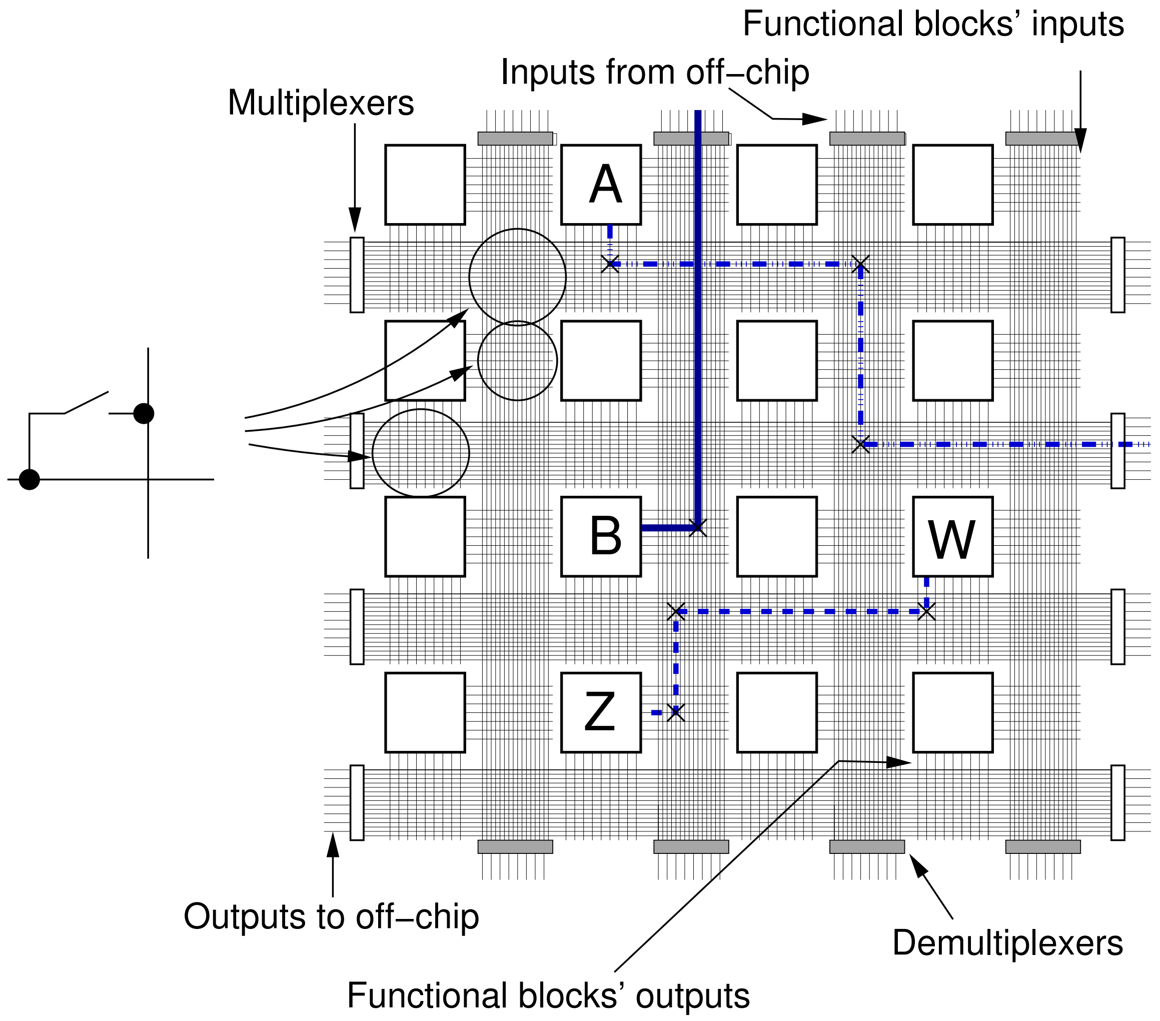

Figure 3.3: Architecture of the VLSI analog computer showing top-level connections.

W is routed to the input of a block in MB Z. Sixty-four analog signal inputs enter the interconnection network at the top and bottom of the chip through 1:2 demultiplexers depicted in Fig. 3.3. The solid line in Fig. 3.3 shows how a signal from off-chip can be applied to the input of a block within Macroblock *B*. Likewise, on-chip signals applied to the horizontal wires can be routed off-chip through multiplexers on the left and right sides of the chip, for a total of 64 outputs. The bold, dashed/dotted line connected to Macroblock *A* shows how an output signal from within Macroblock *A* is routed to off-chip.

3.2 Process Details

The chip was designed and fabricated in a 0.25 μm CMOS process from the Taiwan Semiconductor Manufacturing Corporation (TSMC). The process features thin-oxide devices for a 2.5 V V_{DD} and thick-oxide devices, capable of tolerating 3.3 V but with minimum lengths of 0.3 μm and 0.35 μm for the PMOS and NMOS devices, respectively. The process allows for polysilicon resistors, metal-insulator-metal capacitors of about 1 fF/μm^2 and has a non-epitaxial substrate. It is a single polysilicon, five-metal process with minimum metal line widths ranging from 0.32 μm for Metal-1 to 0.44 μm for Metal-5 and similar minimum line spacings.

3.3 Block Circuit Design

3.3.1 Elements Common to Most Circuits

All circuits have class-A inputs and outputs, with the exception of the logarithm circuits, which have class-AB inputs and the exponential circuits, which have class-AB output circuits. To accommodate larger dynamic range, most class-A signal ports have 100 nA, 1 μA, and 20 μA signal ranges. For some ports, the largest signal range is 10 μA.

Circuits in analog computation are used in a wide variety of configurations and as such, it is difficult to predict how the performance of one circuit will effect the overall accuracy of the system being simulated. While one could respond to this by designing circuits to meet extremely high performance standards, the usual costs would be incurred, namely increased design time, complexity, area and power consumption. Instead, some moderate performance targets were set such that all nonidealities affect the instantaneous accuracy of a block equally. For example the integrated equivalent input noise specification was set to be the same, as a percent of full-scale signal, as the nonlinear distortion. The targets are summarized below:

- Integrated equivalent input noise up to 1 MHz: 0.1 % of full scale.
- Maximum deviation from linearity for linear blocks: 0.1 % at half of full scale.
- Non-dominant pole location > 1 MHz.
- Matching of critical pairs of transistors: $\frac{\sigma I_{DS}}{I_{DS}} = 0.1\%$.
- Output resistance: > 1000 x input resistance. Recall that the blocks are current-in, current-out.
- Quiescent input voltage: 1.25 V to 1.8 V.

3.3.2 Integrator

A block diagram of the integrator is shown in Fig. 3.4. Wire labels follow the following convention: a label adjacent to a part of wire on which an arrow is superimposed refers to the current flowing in the wire. Labels not adjacent to arrows denote the voltage

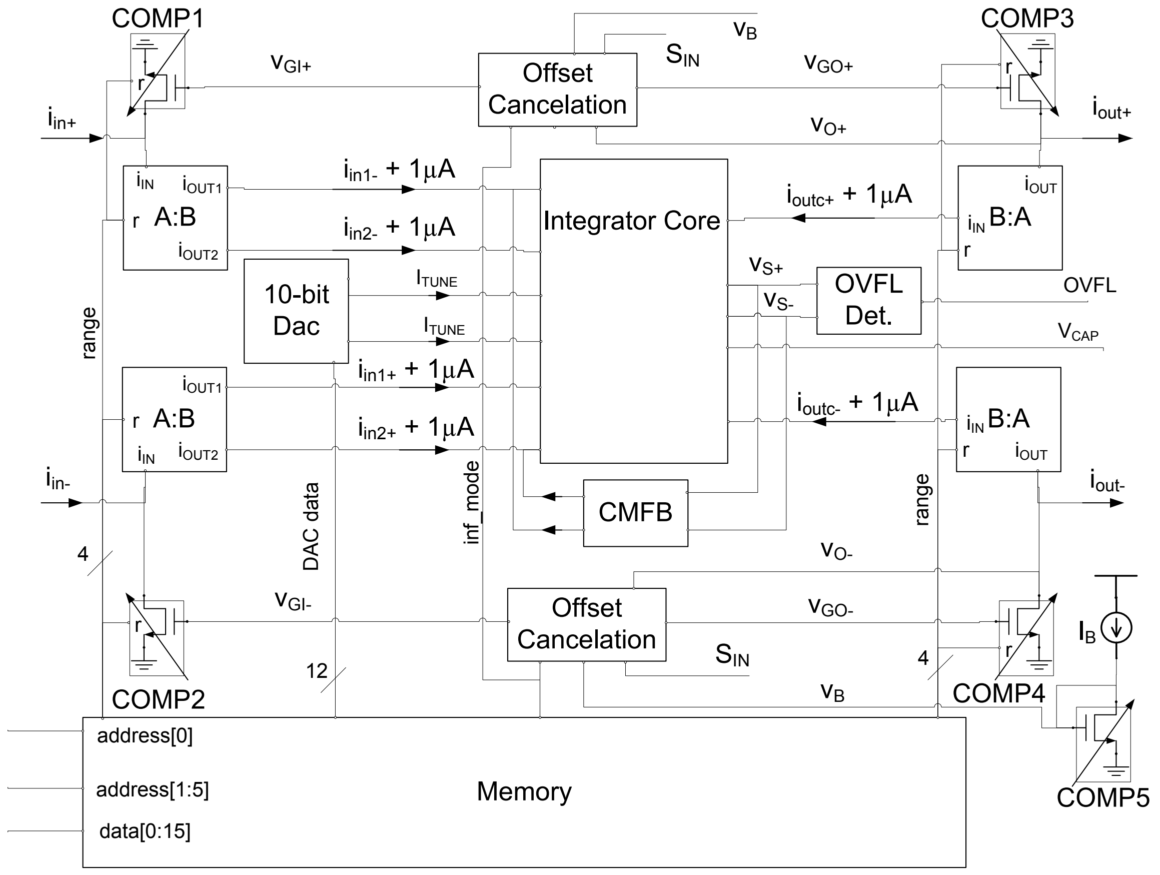

Figure 3.4: Block diagram of the integrator.

of the wire with respect to ground. Signals i_{in+} and i_{in-} form the circuit's class-A differential input. Signals i_{out+} and i_{out-} form the circuit's class-A differential output. The integration operation is performed by the block labeled "Integrator Core". The wires labeled "$i_{in1+} + 1\mu$A", "$i_{in2+} + 1\mu$A", "$i_{in1-} + 1\mu$A" and "$i_{in2-} + 1\mu$A" apply class-A analog input signals to the core, along with 1 μA biases. The integrator core, as we will derive below, has the following input-output characteristic:

$$\frac{d}{dt}\left(i_{outc+} - i_{outc-}\right) = K\left(i_{in1+} + i_{in2+} - i_{in1-} - i_{in2-}\right) \tag{3.1}$$

where K equal to half of the unity gain angular frequency of the integrator.

The blocks labeled “A:B” are single-input, dual-output current mirrors with programmable gains, having the following input-output characteristics in terms of the signals labeled in Fig. 3.4:

$$i_{in1+} = i_{in2+} = -\frac{B}{A} i_{in-} \tag{3.2}$$

and

$$i_{in1-} = i_{in2-} = -\frac{B}{A} i_{in+} \tag{3.3}$$

Along with composite devices COMP1 through COMP5, “A:B” blocks allow the integrator to have multiple input signal ranges, while always supplying each of the core’s inputs with a bias of 1 μA. The blocks labeled “B:A” are single-input, single-output current mirrors with programmable gains, having the following input-output characteristics in terms of the signals labeled in Fig. 3.4:

$$i_{out+} = \frac{A}{B} i_{outc+} \tag{3.4}$$

and

$$i_{out-} = \frac{A}{B} i_{outc-} \tag{3.5}$$

The core of the integrator, through wires “$i_{outc+} + 1\mu$A” and “$i_{outc-} + 1\mu$A” applies signals and bias to the output mirrors, which allow for three different output signal ranges. Input and output mirrors are adjusted so that the input signal limit is equal to the output signal limit. This gives rise to the following input-output characteristic for the whole integrator:

$$\frac{d}{dt}\left(i_{out+} - i_{out-}\right) = 2K\left(i_{in+} - i_{in-}\right) \tag{3.6}$$

While an open loop integrator does not, strictly speaking, have a time constant, for the purpose of this thesis, the term "time constant" will refer to the time constant that the integrator would have if it were placed in unity-gain negative feedback. If the open loop integrator has a transfer function of $H(s) = \frac{1}{\tau s}$, the closed loop system will have a time constant of τ. For open loop integrators, the term time constant will refer to this τ. It is seen that τ is the inverse of the unity-gain angular frequency of the integrator.

The time constant of the integrator in Fig. 3.4 is dependent on the two copies of the tuning current I_{TUNE} which are generated by the block labeled "10-bit DAC". The block OVFL raises the digital signal OVFL when the integrator's differential output signal is near its limit. The block labeled CMFB regulates the common mode level of the integrator's differential output current. The two blocks labeled "Offset Cancellation" perform dynamic cancellation of the integrator's input and output offsets. The block labeled Memory stores the DAC's input word, range settings, and other control data.

Control signal V_{CAP} helps reset the integrator. S_{IN} controls the offset cancellation sequence. The signals data[0:15] specify the data to be programmed to the block's memory. The signal address[0] latches the data into the block if the address lines, address[1:5] are all high. A particular block is identified by the address lines in the following way:

- Around the chip run five address signals (a[1:5]) and their complements ($\overline{\mathrm{a}[1:5]}$).

- The i^{th} address input of the Memory block is connected to either a[i] or $\overline{\text{a[i]}}$.
- The particular block is activated whenever all of its five address inputs are high.

For example, if the five address lines of a block are connected to a[1:2], $\overline{a[3:4]}$ and a[5], its memory is activated when a[1:5]=11001.

Integrator Core

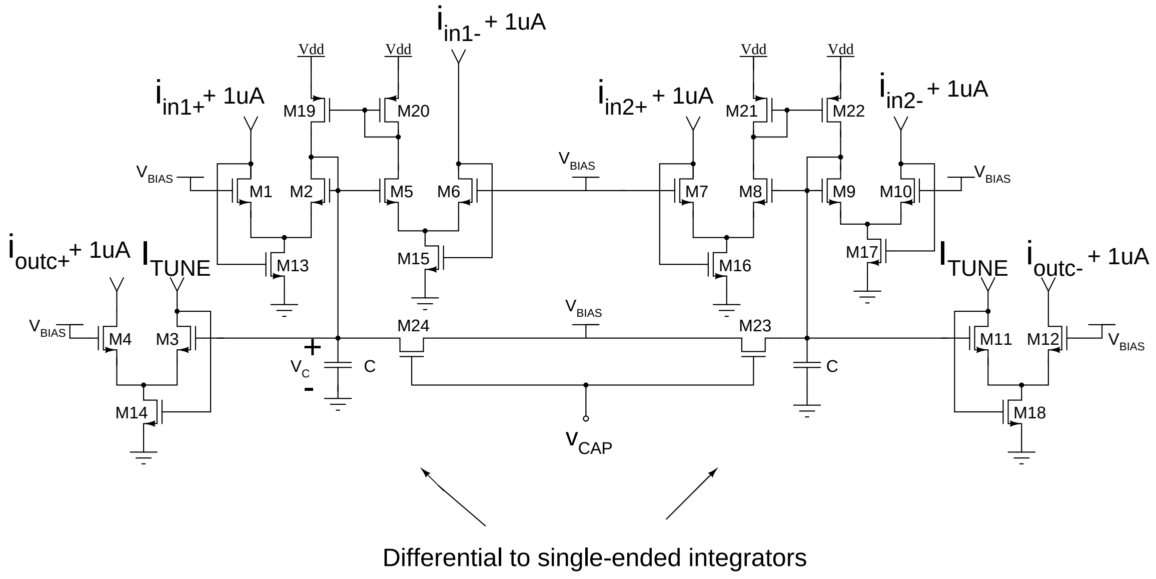

Figure 3.5: Integrator Core.

The schematic of the core of the integrator is shown in Fig. 3.5. It consists of two differential to single-ended log-domain integrators using MOSFETs, similar to those in [16]. Log-domain integrators were selected in an attempt to reuse as much of an earlier design as possible. An earlier, smaller, version of the chip used log-domain integrators without range-selecting input and output mirrors (blocks A:B and B:A), requiring that they operate over a wide range of bias currents. The externally-linear, internally-nonlinear behaviour of log-domain integrators makes them well suited to such an application. If a log-domain integrator is made with bipolar junction transistors, its usable range (linear, and not overly noisy) can be over many decades of bias current. However, when MOSFETs are used, the upper range of current must be kept small enough that the devices stay weakly inverted. Reducing the current too much leads to a poor maximum signal-to-noise ratio, since the signal range falls faster than the noise level does as the circuit's bias currents are reduced. When the integrator for the chip described in this thesis was designed, the log-domain core was kept, but the range-selectable mirrors were added. It was simpler to design them than an integrator that could handle the full signal range.

Circuit Operation: The core consists of two differential to single-ended integrators found in the right and left halves of Fig. 3.5. Transistors M1 through M12 operate in weak inversion. Transistors M13 through M18 keep the impedance low at the drain of M1, M3, M6, M7, M10 and M11, respectively, allowing the input and tuning currents to enter the circuit at a low-impedance point. The transistor pairs M19/M20 and M21/M22 form unity-gain current mirrors. We will perform an analysis of the lefthand differential to single-ended integrator consisting of transistors M1 through M6, M13 through M15, M19, and M20 and the capacitor C on the left side of the figure. This analysis assumes:

- V_{CAP} is low, shutting off M24.
- All other transistors in the integrator are in saturation.

- Output conductances of transistors are zero.
- The body effect can be ignored.
- All parasitic capacitances and device capacitances can be ignored.
- The following pairs of transistors are identical to one another: M1 and M6; M2 and M5; M19 and M20;
- Transistors M1 through M6 are weakly inverted and each drain-source current is described by [17]:

$$i_{DS} = SI_S \exp(\frac{v_{GS}}{n\phi_t}) \tag{3.7}$$

Where S is the device's aspect ratio (i.e. $\frac{W}{L}$), I_S is a constant of proportionality with units of current, v_{GS} is the device's gate to source voltage, n is the subthreshold slope factor for the device and ϕ_t is the thermal voltage (kT/q). Eq. 3.7 can be rearranged to give:

$$v_{GS} = n\phi_t \log(\frac{i_{DS}}{SI_S}) \tag{3.8}$$

A loop in a circuit composed of only gate-source voltage drops of weakly inverted MOSFETs (or BJTs) is called a translinear loop. The analysis of this circuit will proceed in a fashion very similar to that of other translinear circuits [18].

There are two translinear loops in the circuit, which are composed of: M1, M2, M3 and M4; and M6, M5, M3 and M4. Even though each of these loops starts and ends with a different element, they form electrical loops since the gates of the start and end devices are connected to the same voltage. Around each loop a Kirchoff's

Voltage Law (KVL) equation can be written:

$$v_{GS1} - v_{GS2} + v_{GS3} - v_{GS4} = 0 \tag{3.9}$$

$$v_{GS6} - v_{GS5} + v_{GS3} - v_{GS4} = 0 \tag{3.10}$$

When Eq. 3.8 is substituted for each of the gate-source voltages in Eq. 3.9 and Eq. 3.10, the KVL equations become:

$$n\phi_t \log(\frac{i_{DS1}}{S_1 I_S}) - n\phi_t \log(\frac{i_{DS2}}{S_2 I_S}) + n\phi_t \log(\frac{i_{DS3}}{S_3 I_S}) - n\phi_t \log(\frac{i_{DS4}}{S_4 I_S}) = 0 \tag{3.11}$$

$$n\phi_t \log(\frac{i_{DS6}}{S_6 I_S}) - n\phi_t \log(\frac{i_{DS5}}{S_5 I_S}) + n\phi_t \log(\frac{i_{DS3}}{S_3 I_S}) - n\phi_t \log(\frac{i_{DS4}}{S_4 I_S}) = 0 \tag{3.12}$$

Eq. 3.11 and Eq. 3.12 can be manipulated into the following form:

$$\frac{i_{DS1} i_{DS3}}{S_1 S_3} = \frac{i_{DS2} i_{DS4}}{S_2 S_4} \tag{3.13}$$

$$\frac{i_{DS6} i_{DS3}}{S_6 S_3} = \frac{i_{DS5} i_{DS4}}{S_5 S_4} \tag{3.14}$$

Eq. 3.13 and Eq. 3.14 will be used later. For now, consider the Kirchoff's Current Law (KCL) equation written at the top terminal of the capacitor:

$$i_C = -i_{DS19} - i_{DS2} \tag{3.15}$$

Since M19 is a PMOS device, its drain current is defined upward from its drain to V_{DD}. Because M19 and M20 are identical, they act as a unity-gain mirror, mirroring the drain current of M5 into the capacitor. Therefore M19 conducts the same current as M5.

$$i_{DS19} = -i_{DS5} \tag{3.16}$$

Substituting Eq. 3.16 into Eq. 3.15 gives:

$$i_C = i_{DS5} - i_{DS2} \tag{3.17}$$

The output of the integrator's core is i_{DS4}. Since the circuit is an integrator, we are interested in an expression for the time derivative of the output variable. The time derivative of Eq. 3.7 for M4 gives:

$$\dot{i}_{DS4} = S_4 I_S \exp(\frac{v_{GS4}}{n\phi_t}) \frac{\dot{V}_{G4} - \dot{v}_{S4}}{n\phi_t} \tag{3.18}$$

Where V_{G4} is the voltage at the gate of M4, with respect to ground, and, v_{S4} is the voltage at the source of M4, with respect to ground. Recognizing that the first part of the right-hand side is simply i_{DS4}, and that $\dot{V}_{G4}$ is zero, since the gate of M4 is connected to a DC voltage source, Eq. 3.18 becomes:

$$\dot{i}_{DS4} = -\frac{\dot{v}_{S4}}{n\phi_t} i_{DS4} \tag{3.19}$$

Note that v_{S4} and v_{S3} are equal to one another. Since i_{DS3} is kept constant by I_{TUNE}, V_{GS3} is also a constant. Also, because the gate of M3 is connected to one terminal of the capacitor, the rate of change of v_{G3} will be the same as the rate of change of the capacitor's voltage. These facts combine to give:

$$\dot{v}_{S4} = \dot{v}_{G3} = \dot{v}_C = \frac{1}{C} i_C \tag{3.20}$$

Substituting Eq. 3.17 into Eq. 3.20 and combining this result with Eq. 3.19 gives:

$$\dot{i}_{DS4} = \frac{1}{n\phi_t C} i_{DS4}(i_{DS2} - i_{DS5}) \tag{3.21}$$

Now we rearrange Eq. 3.13 and Eq. 3.14 and isolate i_{DS2} and i_{DS5}, respectively giving:

$$i_{DS2} = \frac{i_{DS1} I_{DS3} S_2 S_4}{i_{DS4} S_1 S_3} \tag{3.22}$$

$$i_{DS5} = \frac{i_{DS6} I_{DS3} S_5 S_4}{i_{DS4} S_6 S_3} \tag{3.23}$$

Eq. 3.22 and Eq. 3.23 can be substituted into Eq. 3.21 to give:

$$\dot{i}_{DS4} = \frac{1}{n\phi_t C} i_{DS4} \left(\frac{i_{DS1} I_{DS3} S_2 S_4}{i_{DS4} S_1 S_3} - \frac{i_{DS6} I_{DS3} S_5 S_4}{i_{DS4} S_6 S_3} \right) \tag{3.24}$$

Simplifying and noting that $S_2 = S_5$ and $S_1 = S_6$, Eq. 3.24 becomes:

$$\dot{i}_{DS4} = \frac{S_2 S_4}{S_1 S_3} \frac{I_{DS3}}{n\phi_t C} (i_{DS1} - i_{DS6}) \tag{3.25}$$

This equation describes the behaviour of the circuit in terms of total drain-source currents and not the signal quantities labeled in Fig. 3.5. Eq. 3.25 can be cast in terms of signal quantities by noting the following:

$$i_{DS1} = i_{in1+} + 1\mu\text{A} \tag{3.26}$$

$$i_{DS6} = i_{in1-} + 1\mu\text{A} \tag{3.27}$$

$$i_{DS4} = i_{outc+} + 1\mu\text{A} \tag{3.28}$$

and therefore:

$$\dot{i}_{DS4} = \dot{i}_{outc+} \tag{3.29}$$

When Eq. 3.26, Eq. 3.27, and Eq. 3.29 are substituted into Eq. 3.25, we get:

$$\dot{i}_{outc+} = \frac{S_2 S_4}{S_1 S_3} \frac{I_{TUNE}}{n\phi_t C} (i_{in1+} - i_{in1-}) \tag{3.30}$$

In Eq. 3.30 I_{TUNE} has replaced I_{DS3}. It can thus be seen that we have a differential input, single-ended output integrator whose time constant can be tuned through I_{TUNE}.

A similar analysis can be carried out for the right-hand integrator, which can be combined with Eq. 3.30 assuming that the following sets of transistors are identical to one another: M1, M6, M7 and M10; M2, M5, M8 and M9; M19 and M20; M21 and M22; M4 and M12; M3 and M11; giving:

$$\dot{i}_{outc-} = \frac{S_2 S_4}{S_1 S_3} \frac{I_{TUNE}}{n\phi_t C}(i_{in2-} - i_{in2+}) \tag{3.31}$$

Eq. 3.30 and Eq. 3.31 can be combined to give:

$$\frac{d}{dt}(i_{outc+} - i_{outc-}) = \frac{I_{TUNE} S_2 S_4}{n\phi_t C S_1 S_3}(i_{in1+} + i_{in2+} - i_{in1-} - i_{in2-}) \tag{3.32}$$

Eq. 3.32 will be related to the behaviour of the entire integrator once the operation of the blocks labeled “A:B” and “B:A” in Fig. 3.4 is described. Eq. 3.32 is not to suggest that the circuit, as described thus far, is fully differential. Rather, it is pseudo-differential.

Transistor M24 allows the capacitors to be pre-charged to V_{BIAS}. This hastens resetting the integrator, and ensures that transistors M2, M5, M19, and M20 become biased properly. Without M24 a combination of DC voltages may make it impossible for the integrator to reach a state in which the operation described above applies. For example, if $v_C = 0$ and the drain voltage of M5, v_{D5}, is at V_{DD}, transistors M2, M5, M19 and M20 are all off and regardless of the current flowing in M1 and M6, the capacitor will stay discharged. This does not contradict the analysis above, since the analysis assumed that transistors M2, M5, M19 and M20 are each on and in saturation.

The integrators have a nominal time constant of 40 μs. Each integration capac-

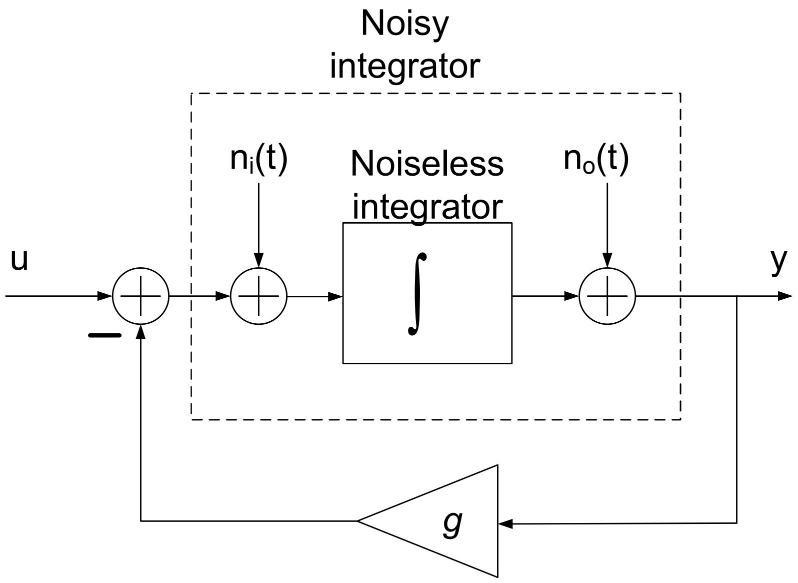

Figure 3.6: Simple system for noise analysis.

itor is implemented as an 8 by 10 array of NMOS transistors ($W = 10\mu\text{m},\ L = 10\mu\text{m}$) giving a net capacitance of 40 pF. The capacitors of an integrator occupy 25 % of the area of the integrator, despite efforts to shrink the capacitor without changing the nominal time constant. From Eq. 3.32 it is clear that reducing S_2 (and the aspect ratios of M5, M8 and M9 since M2, M5, M8 and M9 are assumed to be identical) proportionately with C keeps the capacitor area small without changing the transfer characteristics of the integrator. In this design S_2, S_5, S_8 and S_9 are 2.5 % of the aspect ratio of the other transistors in the translinear loops. This also reduces the current through M2, M5, M8 and M9 and increases their contributions to the integrator's output noise. How exactly this affects the noise of a given simulation is very dependent on the details of the system, even for very small systems. For example, consider the system shown in Fig. 3.6. Assume signals $n_i(t)$ and $n_o(t)$ are uncorrelated noise sources with flat power spectral densities, $N_i(f) = N_i$ and $N_o(f) = N_o$, respectively. Here, n_i represents all noise sources in the integrator on the input side of the integration capacitor and n_o represents all noise sources on the output side of the integration capacitor. Despite the integrator being an internally nonlinear system, the noise analysis below assumes that the system is linear, which is a valid assumption when the signal the integrator is processing (input and output) is small. The noise at the output, $N_t(f)$, will be:

$$N_t(f) = N_i|H_{LP}(f)|^2 + N_o|H_{HP}(f)|^2 \tag{3.33}$$

where

$$H_{LP}(f) = \frac{1}{j2\pi f + g} \tag{3.34}$$

$$H_{HP}(f) = \frac{j2\pi f}{j2\pi f + g} \tag{3.35}$$

$H_{LP}(f)$ is the transfer function of this system from the input of the integrator to the output of the system, whether the input is noise at the input of the integrator or an input signal to the system. $H_{HP}(f)$ is the transfer function from the output of the integrator to the output of the system. This noise analysis assumes that the gain block g is noiseless. Clearly, the relative contribution of input noise to the total noise is dependent on g, a parameter of the system being simulated. As such, the optimal allocation of the circuit's noise is dependent on the system being simulated.

The quartet of devices M2, M5, M8 and M9, the dominant source of input noise, were sized such that they contributed approximately half of the core's noise when the integrator was in the configuration discussed here, with $g = 1$, and the noise was integrated up to 1 MHz.

Settable-Gain Input and Output Mirrors

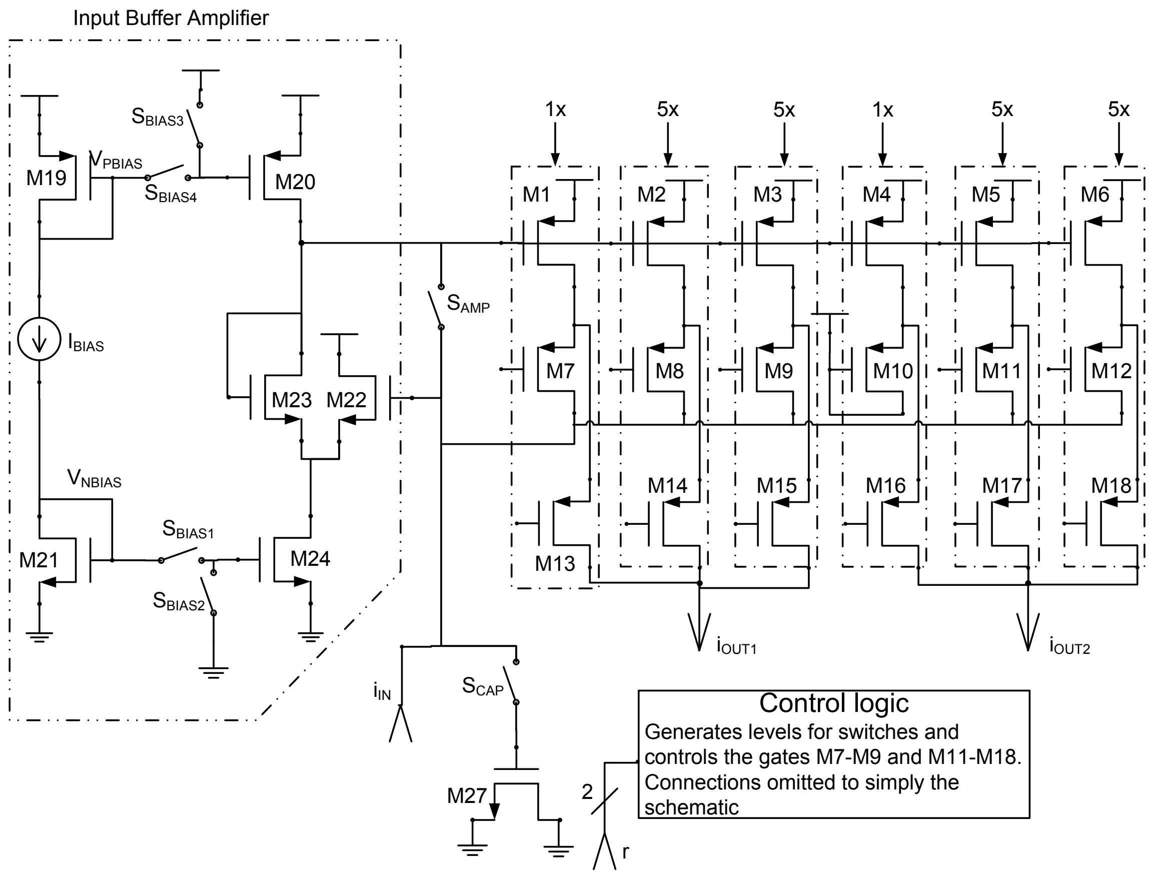

Figure 3.7: Input variable gain current mirror.

The core of the integrator uses weakly inverted MOSFETs whose relationship between gate to source voltage (V_{GS}) and drain current (I_{DS}) is exponential. It is this characteristic that makes the core externally linear. However, for larger drain currents, the devices become moderately inverted and their current-voltage characteristics are no longer exponential. One could make the devices wider, extending the current up to which the exponential relationship is maintained. However, this would be at the penalty of circuit area, since all capacitances would increase. Alternatively, the length of the active devices could be decreased as their width is increased, maintaining a fixed area. This would reduce their output resistance and their exponential characteristics would be limited by short channel effects. Instead, to allow for a larger range of input and output current, settable-gain input and output current mirrors were used (labeled A:B and B:A, respectively, in Fig. 3.4). Fig. 3.7 shows a simplified schematic of the *input* mirror. The numbers above the dashed boxes in the figure indicate the number of unit devices of which each transistor is composed.

Each input mirror has one analog input current and two equal output currents so that each polarity of the integrator's input can be applied to each of the core's two differential to single-ended halves (Fig. 3.4). The input mirror consists of mirroring devices M1 to M6, input current-steering devices M7-M12, output current-steering devices M13-M18, a unity-gain buffer amplifier (M19-M24 and S_{BIAS1} through S_{BIAS4}), a by-pass to the amplifier (S_{AMP}), some devices for compensation (S_{CAP} and M27), and some control logic. By appropriately controlling the gates of M7-M18 (unit size is $W = 1\,\mu\mathrm{m}$, $L = 0.3\,\mu\mathrm{m}$), the circuit achieves mirroring ratios of 20:1, 1:1, 1:10 from input to each of its two outputs. Device M10, since its gate is connected to V_{DD}, never conducts. It is included so that the capacitive loading at the drain of M4 is the same as the loading at the drain of M1. The input bias to the mirror is adjusted (20

μA, 1 μA, 100 nA) so that the output bias is always 1 μA. Table 3.1 details how the current steering transistors M7-M18 are controlled.

Mirror Ratio	M1	M2	M3	M4	M5	M6
20:1	M13	M8	M9	M16	M11	M12
1:1	–	M8	M15	–	M17	–
1:10	M7	M14	M15	–	M17	M18

Table 3.1: Control for current steering switches in the integrator's variable-gain input current mirror. Rows two through four indicate the conducting device connected to the device in row one.

Each device listed in the first row of the table above is connected to two current steering devices. The devices listed in rows 2 through 4 indicate which of the two current steering devices associated with the device in the first row is on. The entry "–" denotes that neither current steering device is on. What precisely is meant by "ON" is explained below. For the moment, it can be assumed that the current-steering devices act as switches. The scenario in which the circuit implements a current mirroring ratio of 20:1 is depicted in Fig. 3.8. Transistors draw with a bold line are conducting while the others are not. M13 is on thereby connecting M1 to the output terminal i_{OUT1}. M2, M3, M5 and M6 are connected to the input i_{IN} for a total of 20 unit devices, while M4 is connected to i_{OUT2}. This means that there is one unit device supplying current to each output. M1 through M6 have the same gate-source voltage. Assuming that they are in saturation, the connections described above will lead to a mirroring ratio of 20:1.

The block labeled "Control Logic" takes a two-bit signal, r, as its input and

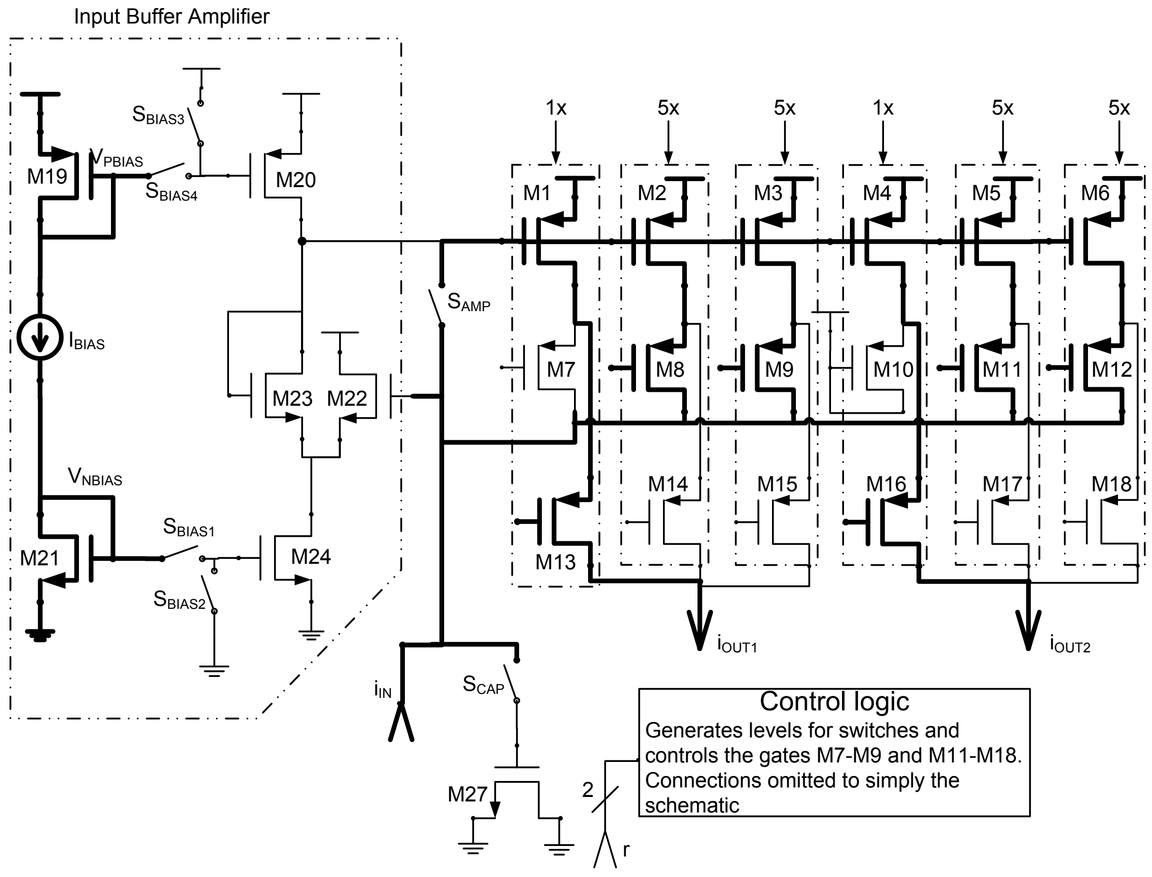

Figure 3.8: Input variable gain current mirror implementing 20:1 mirror ration. Devices drawn in bold are on.

generates the necessary control signals for the switches $\mathrm{S_{BIAS1}}$ through $\mathrm{S_{BIAS4}}$, $\mathrm{S_{CAP}}$ and $\mathrm{S_{AMP}}$, and the gates of the current steering transistors.

The simplest way to operate the mirror would have been to directly connect the input (i_{IN}) to the gates and drains of M1-M6. This, however, would load the input with a large capacitance ($\sim$ 11 pF), since the unit transistor of M1-M6 is large ($W = 10\,\mu$m, $L = 10\,\mu$m). When put in parallel with the circuit's input resistance, the circuit's frequency response would suffer. The input resistance of the circuit is $1/g_m$ where g_m is the transconductance of the subset of transistors M1 to M6 that is connected to the input. For the mirroring ratio of 1:10, the input current, and g_m, is the smallest and the input resistance is largest. This combination of input capacitance and input resistance would result in a pole in the mirror's frequency response near 40 kHz.

To prevent the gate capacitance of the mirror's large devices (M1-M6) from limiting the bandwidth of the input mirror when the input resistance is high, the input is not connected directly to the gates of M1-M6. For the 100 nA and 1 μA ranges, switches $\mathrm{S_{BIAS1}}$ and $\mathrm{S_{BIAS4}}$ are on, switches $\mathrm{S_{AMP}}$, $\mathrm{S_{BIAS2}}$ and $\mathrm{S_{BIAS3}}$ are off, and M19-M24 form a unity-gain buffer from the voltage at i_{IN} to the gates of M1-M6. Mismatch between M22 and M23 will cause the buffer to have an input offset and affect the input voltage of the mirror, but will not change the mirroring ratio. Since matching between M22 and M23 is relatively unimportant these devices can be made much smaller than M1-M6 and therefore do not load the input. When the input is shielded from M1-M6, the input is still loaded with a capacitor. This comes from the wire that connects the input of the block to the switching grids, typically ~2 pF. The feedback loop, from the input, through M22, M23, M2, and M8 can be unstable on the 1 μA range, unless $\mathrm{S_{CAP}}$ connects a small compensation capacitor (M27, 0.4 pF) to the input. For the largest input range, $\mathrm{S_{BIAS1}}$ and $\mathrm{S_{BIAS4}}$ are off, switches $\mathrm{S_{AMP}}$, $\mathrm{S_{BIAS2}}$ and $\mathrm{S_{BIAS3}}$ are on. This switches off the buffer, creating a simple current mirror. $\mathrm{S_{CAP}}$ is off.

The unit device is large enough so as to ensure good matching. For devices that are weakly inverted, the relative standard deviation in current for two equally biased devices, considering only threshold voltage mismatch is given by [19]:

$$\frac{\Delta I_{DS}}{I_{DS}} = \frac{A_{V_T}}{n\phi_t\sqrt{WL}} \tag{3.36}$$

where A_{V_T} is a process dependent constant, usually quoted in [mV/μm] and W and L are the dimensions of the transistor in [μm]. For the TSMC025 process $A_{V_T} \simeq$ 5mV/μm and $n\phi_t = 40$ mV, meaning that devices that are 100 μm^2 in gate area match to about 1 %. When set to the 20:1 range, the mirror's matching of i_{OUT1} and i_{OUT2} to one another relies on the matching of single unit devices, which are 100 μm^2 in area.

The gate voltages of M7, M8, M9, M11 and M12 are raised to V_{DD} to shut the devices off or lowered to *gnd* to allow them to connect the mirroring devices to the input. Similarly, the gate voltages of M13-M18 are raised to V_{DD} to turn them off, but when they are connecting the mirroring devices to the output, the gates are lowered to only $V_{DD}/2$. This creates cascode pairs of devices and increases the output resistance of the circuit.

The *output* mirror circuits labeled "B:A" are similar to that in Fig. 3.7 with the following differences:

- The output mirrors implement ratios 1:20, 1:1, 10:1.
- The output mirrors have a fixed input bias of 1 μA.
- Each output mirror has only one output.

For convenience, the equation describing the input-output behaviour of the integrator's core is repeated below:

$$\frac{d}{dt}\left(i_{outc+} - i_{outc-}\right) = \frac{I_{TUNE}S_2S_4}{n\phi_t CS_1S_3}\left(i_{in1+} + i_{in2+} - i_{in1-} - i_{in2-}\right) \tag{3.37}$$

Recall that the input and output mirrors are adjusted so that their mirroring ratios are the reciprocals of one another. For example, when the input mirrors are set to have the ratio of 20:1, the output mirrors have the ratio 1:20. If "A:B" is the mirroring ratio of the input mirrors, $i_{in1+} = i_{in2+} = -\frac{B}{A}i_{in-}$ and $i_{in1-} = i_{in2-} = -\frac{B}{A}i_{in+}$. If "B:A" is the mirroring ratio of the output mirrors, $i_{out+} = \frac{A}{B}i_{outc+}$ and $i_{out-} = \frac{A}{B}i_{outc-}$. When these relationships are substituted into Eq. 3.37, the input-output behaviour of the integrator is found to be:

$$\frac{d}{dt}\left(i_{out+} - i_{out-}\right) = 2\frac{I_{TUNE}S_2S_4}{n\phi_t CS_1S_3}\left(i_{in+} - i_{in-}\right) \tag{3.38}$$

Composite Devices

Composite devices COMP1 through COMP5 in Fig. 3.4 each have nine long channel devices ($W = 1\,\mu$m, $L = 20\,\mu$m) and several short channel devices used as switches. Switches connect the long devices in a 1 by 9, a 3 by 3, or a 9 by 1 array of transistors depending on the levels of digital control signals. Fig. 3.9 shows the three configurations without switches. Fig. 3.10 shows a detailed schematic of the composite device. The circuit's short channel devices are depicted by switches. Transistors M1 through M9 are long channel devices. The label adjacent to each switch indicates the signal that controls the switch. When the signal is high, the switch is closed. Table 3.2 shows the switch control signals that correspond to each configuration.

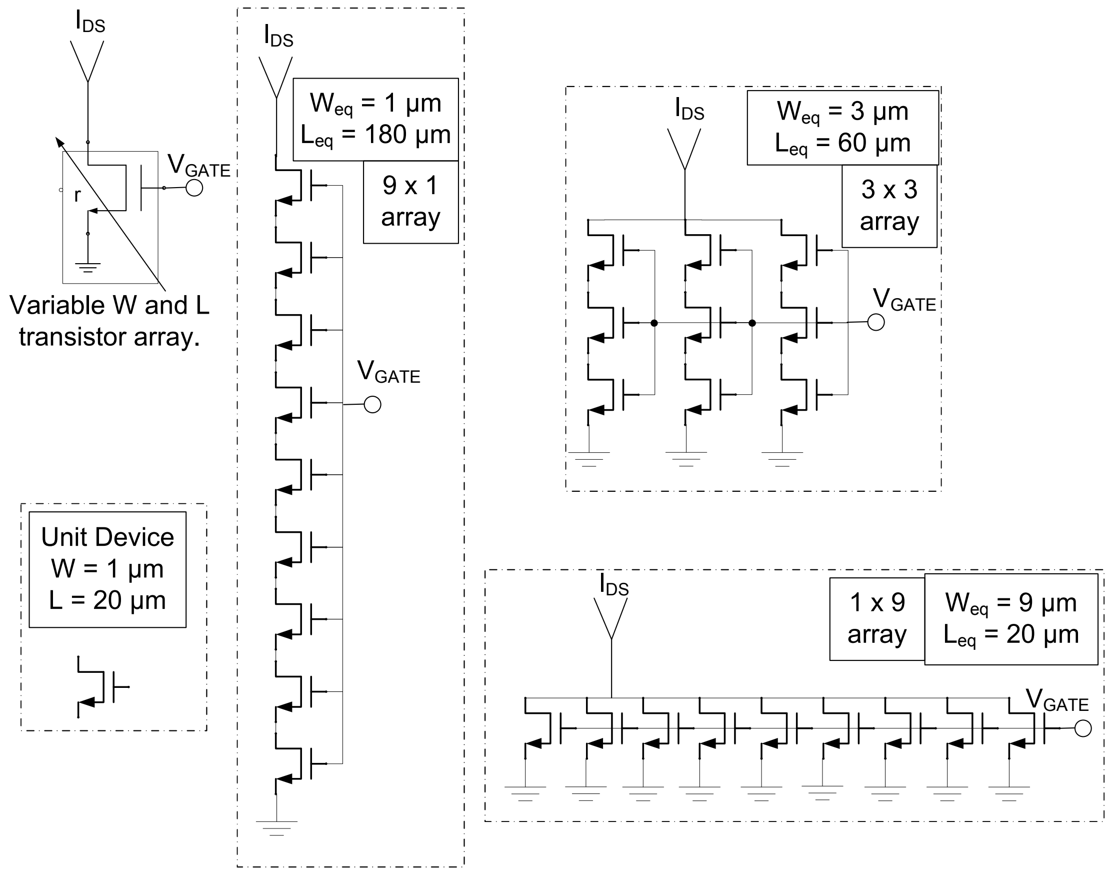

Figure 3.9: Composite device. The Composite device on the left can implement the three configurations of nine devices shown in the figure.

This scheme yields devices of equivalent size (W = 1 μm, L = 180 μm), (W = 3 μm, L = 60 μm) and (W = 9 μm, L = 20 μm), used to carry currents of 100 nA, 1 μA and 20 μA, respectively. While the currents are not exactly proportional to the aspect ratio of the composite device, the level of inversion of the equivalent device changes by only 2.5 while the current changes by 200. This scheme allows for the following:

- Constant WL product, leading to constant ΔV_T mismatch between arrays.

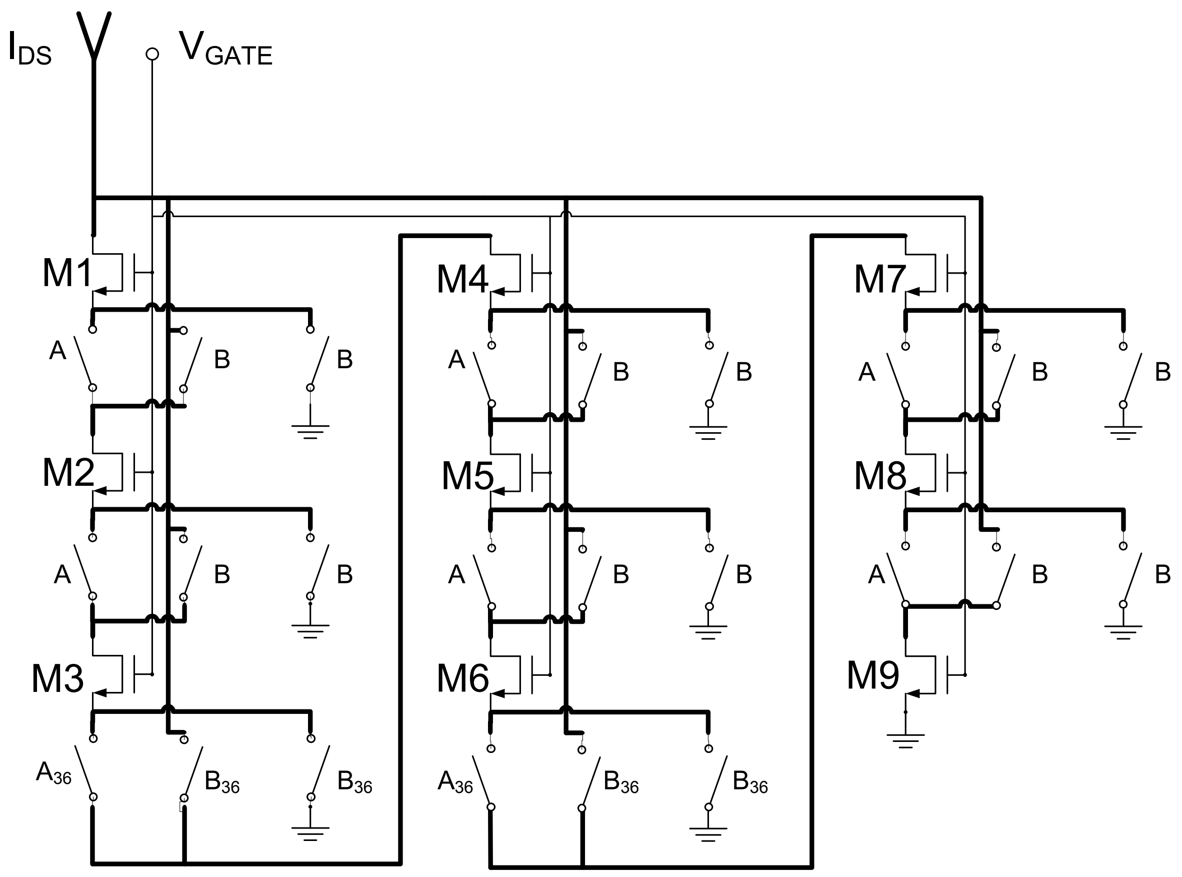

Figure 3.10: Composite device. Switches are drawn in the place of MOSFETs.

- Efficient use of area, since every device is being used at all times.
- Nearly constant V_{DSSAT}. V_{DSSAT} would be constant if the aspect ratio changed proportionally with current.

The net device is large ($WL = 180\ \mu\text{m}^2$) so that one composite device matches a nearby one well.

Equivalent size	A	A_{36}	B	B_{36}
$W = 1\ \mu$m. $L = 180\ \mu$m	high	high	low	low
$W = 3\ \mu$m. $L = 60\ \mu$m	high	low	low	high
$W = 9\ \mu$m. $L = 20\ \mu$m	low	low	high	high

Table 3.2: Control signal levels for each configuration of a composite device.

Common-mode Feedback

An integrator with differential outputs needs common-mode feedback, because its inputs only affect the differential output of the circuit. Without common-mode feedback the integrator, regardless of the differential-mode feedback around it, operates in common-mode open loop. Due to the unavoidable offsets of an integrator, its common-mode output will saturate causing it to no longer process differential signals correctly and it will saturate the input of the circuit to which it is connected.

The common-mode feedback system (enclosed in a dashed line) along with the core of the integrator (enclosed in the box drawn with a dotted line) is shown in Fig. 3.11. The common-mode feedback system operates as follows:

- M4-cmfb, M-12-cmfb. These devices copy the output current of each side of the core of the integrator, assuming they are in saturation, since the gate and source voltages of M4-cmfb and M12-cmfb are the same as M4 and M12, respectively.

- MCM1, MCM2, MCM3. These devices compute the common output, and subtract it from the input to the core. The drains of M4-cmfb and M12-cmfb are connected together, thereby summing their drain currents. This sum is twice

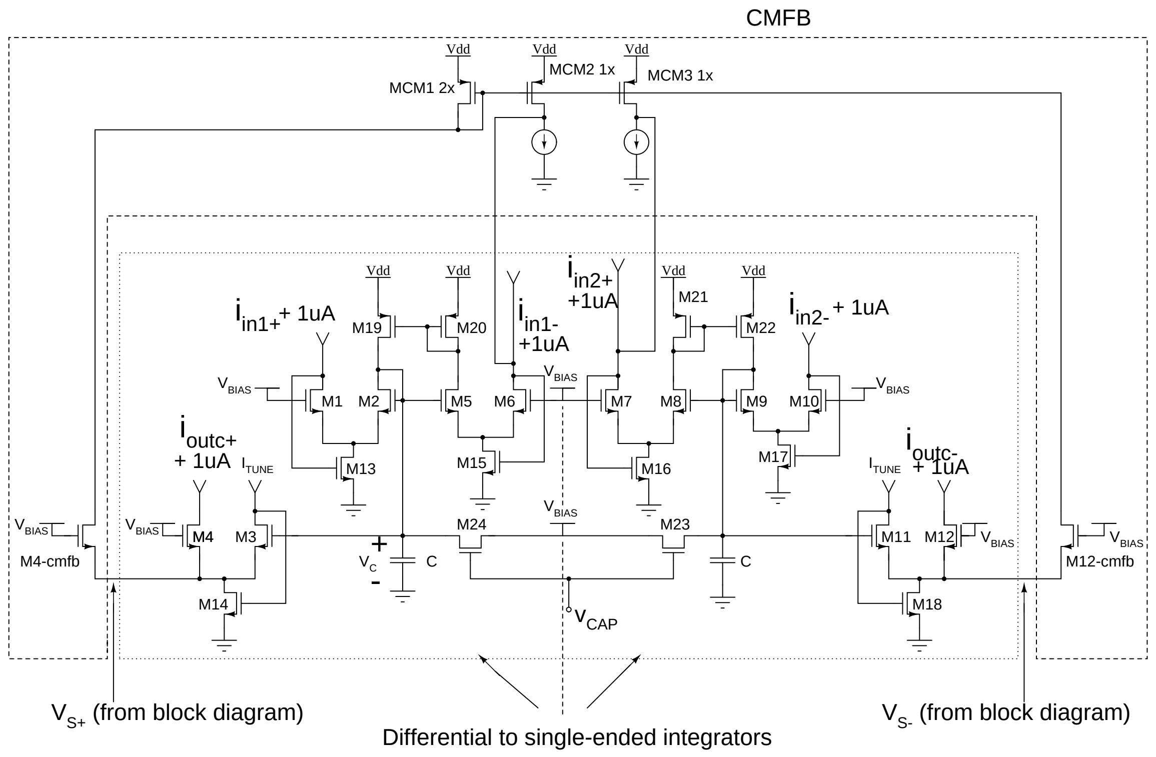

Figure 3.11: Common-mode feedback scheme.

the common mode output of the integrator. Diode connected MCM1 mirrors this sum to MCM2 and MCM3 scaled by a factor of $\frac{1}{2}$. The difference between the mirrored currents and the current sources is applied to two of the core's four inputs in such a way as to cause the common-mode output to approach the current of the current sources, which each conduct 1 μA.

Transistors M4-cmfb and M12-cmfb do not alter the operation of the core of the integrator as derived earlier. The operation was derived by writing a series of KVL equations and KCL equations, none of which assumed that the drain-to-source currents of M3, M4 and M14 summed to zero. The current through output device M4 is determined by its source voltage, since its gain is connected to a fixed voltage.

The source voltage is determined by the tuning current I_{TUNE} flowing through M3 and the voltage across the capacitor. Connecting M4-cmfb does not alter the M4's source voltage.

This common-mode feedback scheme puts the common-mode output in unity-gain, negative feedback. If the integrator had infinite DC gain (i.e., it is an ideal integrator) the error between the actual common-mode output and the desired common-mode output would be driven to zero in steady state by the integration operation. However, since the real integrators have finite DC gain, the error between the actual common-mode output of the integrator and the desired common-mode output (1 μA) will not be driven to zero.

It is imperative that common-mode feedback is implemented as above rather than by generating two copies of each output (by using two devices similar to M4-cmfb for each output), generating two sums, and mirroring one sum to M6 and the other to M7. The unavoidable mismatches between the two feedback paths will result in common mode instability, and an ineffective common-mode feedback scheme.

Offset Cancellation

The integrator has two modes of operation. In one, its input offset is dynamically cancelled before a simulation is run while in the other, no such cancellation takes place.

Fig. 3.12 shows a simplified block diagram of the integrator with an expanded view of the offset cancellation circuitry. In the mode in which no offset cancellation

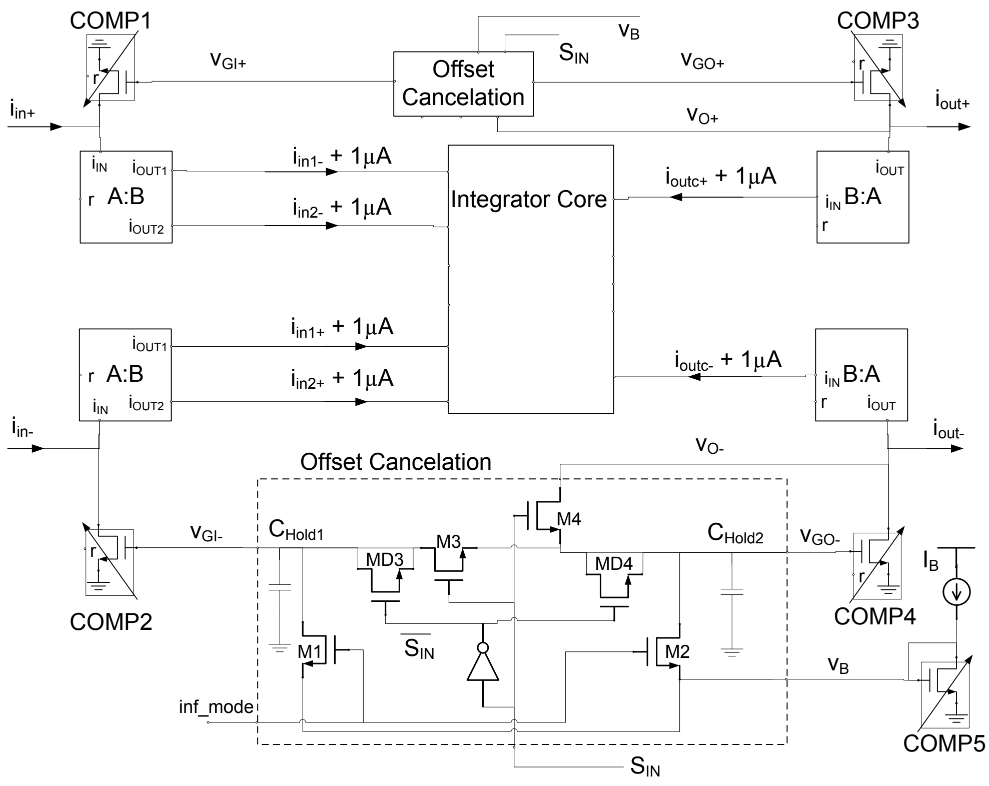

Figure 3.12: Block diagram of the integrator highlighting offset cancellation circuitry.

takes place, signal *inf_mode* is high (V_{DD}) and signal S_{IN} is low (*gnd*). The signal *inf_mode* is short for "infinity mode", indicating that the integrator could operate in this mode indefinitely, while in the other mode, due to the dynamic nature of the offset cancellation scheme, the integrator can operate properly for a finite time. Raising *inf_mode* and lowering S_{IN} connects the gates of composite devices COMP1 through COMP4 to the bias voltage (V_B) generated by the diode-connected COMP5. There is no limit to the duration over which integrators in this mode can be operated, however, due to mismatch, the integrator will have an input offset.

To cancel the offset dynamically, the output of the integrator is not connected to any other circuits, *inf_mode* is lowered to ground and S_{IN} is raised to V_{DD} (see

Fig. 3.12). This connects the integrator in unity-gain, negative feedback. To see how this configuration puts the integrator in negative feedback, consider the following argument: Assume that with i_{in+} and i_{in-} equal to zero, the integrator has reached equilibrium. This assumption requires that the integrator is not in positive feedback. We will see that in fact the integrator is in negative feedback, making this assumption valid. Assume that i_{in-} decreases by some Δi. That is, more current is pulled from the lower A:B block. Therefore, i_{in1+} and i_{in2+} increase by $\frac{B}{A}\Delta i$. This decreases i_{outc-}, as predicted by Eq. 3.37. Since the output of the integrator is not connected to another circuit during offset cancellation, the current flowing into COMP4 decreases, thereby decreasing v_{GO-}. Since the gate of COMP2 is connected to the gate of COMP4 through M3, v_{GI-} also decreases. This reduces the current COMP2 conducts, reducing the current that is pulled from block A:B, and hence reducing the effect of the disturbance at the input. Because the system responds to reduce the effect of an input disturbance, the system is in negative feedback.

The above sequence ignored what happened to i_{outc+} and the feedback through the Offset Cancellation block in the upper portion of Fig. 3.12. Imagine that the decrease in i_{in-} is accompanied by an equal increase in i_{in+}, thereby making the input disturbance differential. Similar reasoning shows that the upper feedback network will tend to compensate for the increase in i_{in+}. Also, the integrator's response to the increase in i_{in+} will reduce the effect of the decrease in i_{in-} and vice versa. In fact, the offset cancellation scheme only responds to the differential component of input disturbances. The common mode component of input disturbances is rejected by the differential structure of the integrator cores.

The discussion above of the system's response to an input when the system is connected in feedback is relevant to the discussion of the system's response to an input referred offset because an integrator with an offset is accurately modeled as an ideal system with an input. When the system reaches steady-state, the necessary input to cancel the offset will be applied by COMP1 and COMP2. To store this input, S_{IN} is lowered, and the voltage needed to apply this input is held on C_{Hold1}, C_{Hold2} and the capacitors inside the upper offset cancellation block, ignoring some nonideal behavior discussed below. This procedure also cancels the integrators's output offset. That is, when the process is finished, the outputs (i_{out+} and i_{out-} in Fig. 3.12) of the integrator will be zero.

While this procedure should exactly cancel the integrator's offsets, it does not because the charge on C_{Hold1} and C_{Hold2} is changed as S_{IN} is lowered by charge injection from M3's and M4's channel charge and by capacitance division between C_{gd} of M3 and M4 and the hold capacitors. To alleviate these problems dummy switches (MD3 and MD4), connected to an inverted version of S_{IN}, are connected to the nodes v_{GI-} and v_{GO-} in Fig. 3.12.

If this procedure is done when other blocks' outputs (except for other integrators) are applied to the input of the integrator, the output offsets of those blocks are also canceled. For example, suppose that the output of an amplifier is connected to the input of the integrator. If the system had no offset cancellation ability, the output offset of the amplifier would degrade the simulation in the same way, and to the same extent that the input offset of the integrator would. However, if the output of the amplifier is connected to the input of the integrator when the cancellation scheme is executed, the output offset of the amplifier is nulled in the same fashion that any other disturbance at the input of the integrator is.

Because charge is stored dynamically, charge will leak, changing the voltage on C_{Hold1} and C_{Hold2} and the performance of the integrator will deteriorate. However, since the leakage from C_{Hold1} and C_{Hold2} in one offset cancellation block should happen at a similar rate to that in the other offset cancellation block, the common-mode feedback of the circuit can maintain adequate performance of the integrators in this dynamic mode for longer than if the circuit were single ended.

The dynamic scheme can reduce the circuit's output resistance since a capacitive feedback path from the output back to the gate of COMP4 exists, through the C_{gd} of the composite device. Even if the composite device has infinite output resistance, in this dynamic mode of operation, the output resistance becomes:

$$R_o = \frac{C_{Hold2} + C_{gd}}{g_m C_{gd}} \tag{3.39}$$

where g_m is the transconductance of the composite device.

Care was taken in the layout of transistors M1 through M4, capacitors C_{Hold1} and C_{Hold2} and the wires carrying S_{IN} and $\overline{S_{IN}}$ to ensure that the coupling capacitances between $S_{IN}/\overline{S_{IN}}$ and nodes v_{GI-} and v_{GO-} were minimized. A guard ring consisting of densely spaced vias from the substrate up to Metal-5 surrounds C_{Hold1} and C_{Hold2}. A grounded layer of Metal-3 separates the wires carrying $S_{IN}/\overline{S_{IN}}$ from the connections to the dynamic nodes.

Overflow Detection

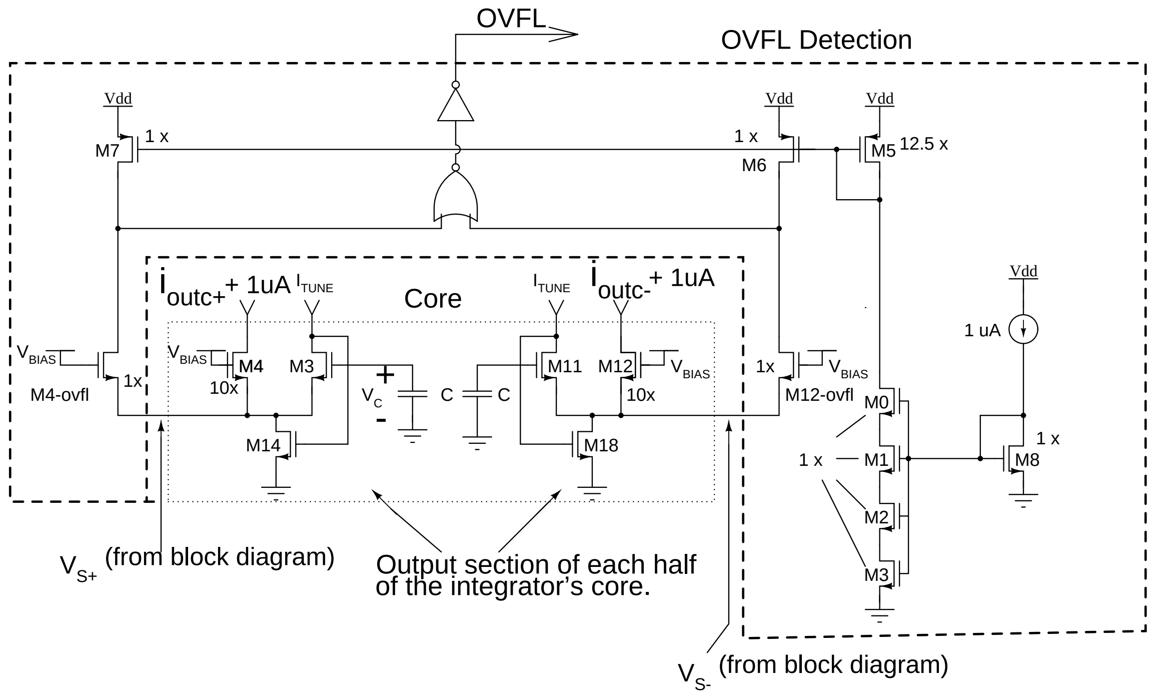

Figure 3.13: Overflow detection circuitry.

An analog computer will give erroneous results if any of its signals exceed the range over which an individual block can accurately process them. For every circuit except the integrator, the size of the output is uniquely determined by the size of the input. By ensuring that the input is limited, it can be guaranteed that the output will not saturate.

On the other hand, there is not a 1-1 correspondence between the input signal level and the output signal level for the integrator, owing to the integration operation. Regardless of how judiciously one limits the size of the input, the output will still saturate if a small input is applied for a long enough time. Therefore, the block that is in greatest need of circuitry to detect saturation (or overflow) at its output is the integrator.

The circuit that does this is shown in Fig. 3.13. Enclosed in the dotted box is the part of the integrator's core that is relevant to overflow detection. Enclosed in the dashed line is the circuitry that detects the integrator's output saturation. This is labeled "OVFL Detection" in Fig. 3.4. The connection between the core and the overflow detection circuitry comes through the wires labeled v_{S+} and v_{S-} in both Figs. 3.4 and 3.13.

The gate of transistors M0 through M3 (in the block labeled OVFL Det.) are at a DC voltage generated by the diode-connected transistor M8. M0-M3 and M8 have dimensions $W = 1\,\mu$m and $L = 20\,\mu$m). Assuming M0 is in saturation, it will conduct about 250 nA because M0 through M3 form a device of $W = 1\,\mu\text{m},\ L = 80\,\mu\text{m}$. This current flows out of the diode-connected PMOS transistor M5. The current mirror consisting of M5-M7 would mirror this current, divided by 12.5, if M6 and M7 are in saturation. The mirroring ratio of $\frac{1}{12.5}$ occurs because the aspect ratio of M5 is 12.5 times that of M6 and M7. M4-ovfl has the same gate and source voltages as M4 in the core of the integrator. If M4-ovfl is in saturation, it conducts $1/10^{th}$ the current that M4 does, since the former is $1/10^{th}$ the width of the latter. For the purpose of this discussion, the term "saturation current" refers to the approximate current a given transistor would conduct if it were in saturation for its present v_G, v_S, and v_B. The signal $OVFL$ goes high if either (or both) of the drains of M6 and M7 are at a voltage near V_{DD}. This will occur if the saturation current of M4-ovfl or M12-ovfl is smaller than the saturation current of M6 or M7 (250 nA/12.5 = 20 nA). This occurs when the saturation current of either M4-ovfl or M12-ovfl is less than 20 nA which corresponds to either M4 or M12 conducting less than 200 nA, or 20 % of its full scale range. M4 or M12 conducting this little current means that the signal has reached 80 % of its full scale range, since the bias current for M4 and M12 is 1 μA. When M4 or M12 is conducting only 200 nA, the other device is conducting 1800 nA, and therefore the overflow circuitry detects when the output is nearing saturation in both the positive and negative direction. The currents processed by this circuit are small, and the $OVFL$ flag may not toggle precisely at 80% of full scale, however, it is somewhat unimportant when exactly the flag is raised, so long as it gets raised before the block saturates.

The digital output $OVFL$ is latched into a scan chain that can be read from the chip after a simulation has finished. A signal indicating whether an overflow has occurred can be monitored during a simulation.

Dac for Tuning

As noted in the discussion of the core of the integrator, the integrator's time constant is inversely proportional to a current, I_{TUNE}. Two equal currents are sourced by the block labeled DAC in Fig. 3.4 which supply the core's two copies of I_{TUNE}. The DAC takes as its reference a current of about 1 μA and a 10-bit digital word. From these, it generates two currents ranging from near 0 to 3 μA, which are ideally equal to one another.

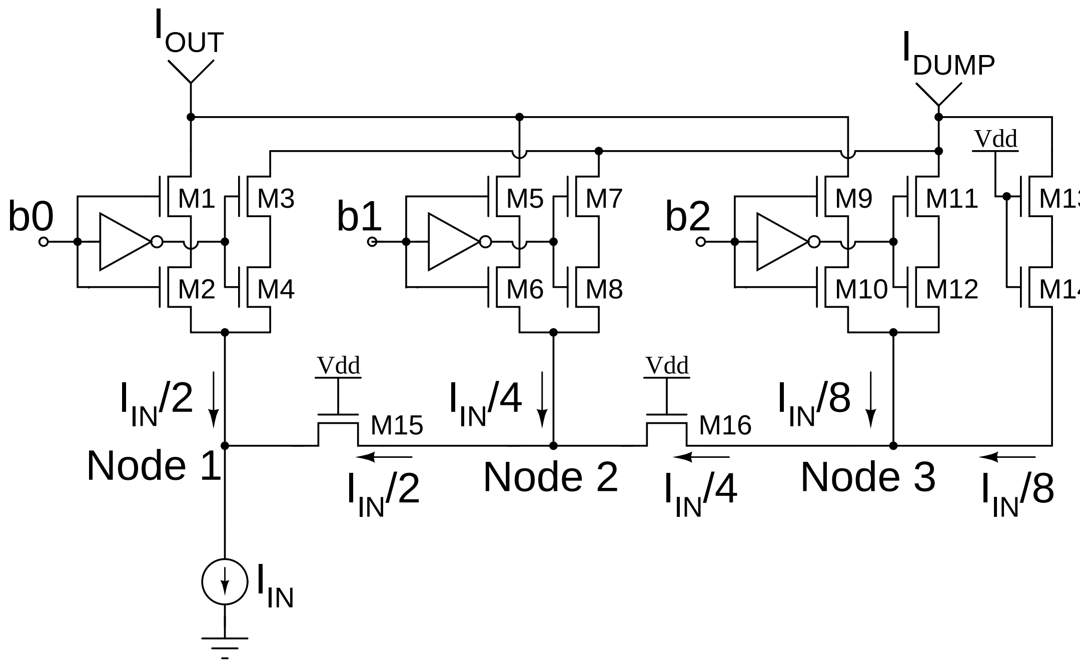

Figure 3.14: Digital to analog converter used to generate tuning currents for the integrator.

The operation of the DAC is much like an R-2-R ladder, in that at each stage the current is divided in two, with half coming from the next stage and the other half either coming from the output or from a dump node. However, here the elements are transistors. Fig. 3.14 shows three bits of the structure. Signals I_{OUT} and I_{IN} refer to the output and input, respectively, of the DAC and should not be confused with similarly named signals in Fig. 3.4. The arrow has been omitted from the symbol for the NMOS transistors used in Fig. 3.14. All transistors in the figure are NMOS devices. For the time being, assume that each NMOS transistor is the same size (W/L), and that nodes I_{OUT} and I_{DUMP} are at voltages high enough to keep all transistors connected to them in saturation. The right-most pair of transistors (M13 and M14), with gates connected to V_{DD} form an equivalent device of $W/2L$. The digital signal b2 selects which series combination of devices (M9/M10 or M11/M12) is on. The pair that is on has the same gate and source voltage as M13/M14. Since it is assumed that nodes I_{DUMP} and I_{OUT} are at a voltage high enough to keep all devices connected to them in saturation, the current through the pairs of devices is determined almost exclusively by their gate and source voltages. Therefore, from the point of view of the current flowing into Node 3, the b2-controlled pair and M13/14 *act* like two devices in parallel, forming an equivalent device of $2W/2L$ which will, in this application, behave like a device of W/L. Now this equivalent device is in series with M16 (W/L) forming a device equivalent to $W/2L$. This analysis continues until we see that the pairs controlled by b0 will form a device of $W/2L$ in "parallel" with a collection of transistors to the right of node 1, which, regardless of the state of signals b1 and b2, form a device of $W/2L$. Hence, I_{IN} is split in two, with half flowing from the right and half flowing from either I_{OUT} or I_{DUMP}. Therefore, the state of b0 determines if $I_{IN}/2$ flows from I_{OUT} or I_{DUMP}. This splitting occurs for each successive bit. Bit b1 determines if $I_{IN}/4$ flows from I_{OUT} or I_{DUMP} and b2 determines if $I_{IN}/8$ flows from I_{OUT} or I_{DUMP}.

I_{OUT} is applied to a simple two-output PMOS current mirror (with a gain of 3), whose two outputs are applied to the integrator core as the two copies of I_{TUNE}. The actual DAC used differs somewhat from that described above in that the series devices (those whose gates are always connected to V_{DD}; M15 and M16 in Fig. 3.14) are slightly shorter than the shunt devices (M1-M14). This has the effect of skewing

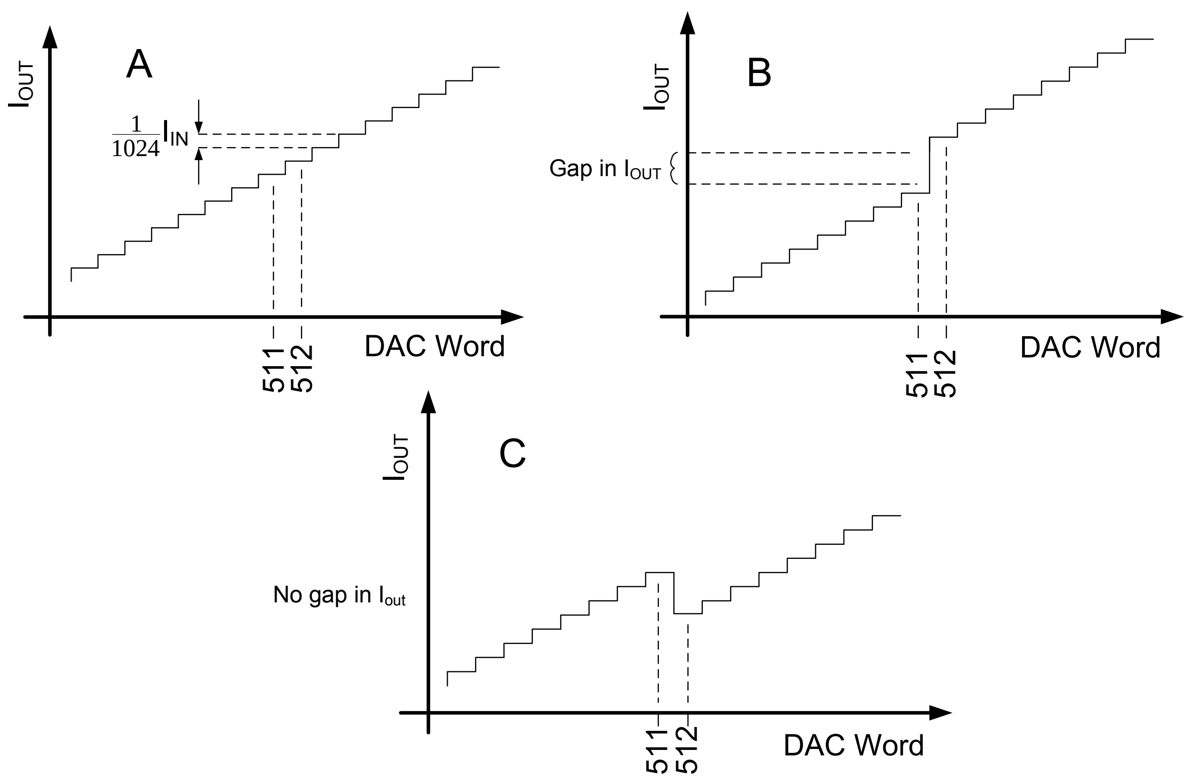

Figure 3.15: I_{out} Vs. DAC word. Ideal and two nonideal characteristics.

the DAC's I_{OUT} vs. DAC word transfer characteristic to be non-monotonic. When the input of a 10-bit R-2-R DAC is 511, b0, the most significant bit is low, and the 9 less significant bits are high. That is, the b0 bit is directing its current ($I_{IN}/2$) from the "dump" node, while the others are directing their currents from the "out" node. When the input is incremented to 512, b0 is high, and b1-b9 are low. Now, b0 directs its current from the "out" node, while the others direct their currents from the "dump" node. In the absence of mismatch, the output current for an input of 512 is one step ($\frac{1}{1024}I_{IN}$) larger than the current for an input of 511. Fig. 3.15 A shows this ideal case. However, if M15 is longer (less conductive) than the other transistors, the characteristic in Fig. 3.15 B results. As shown in the figure, this larger step in the I_{OUT} Vs. DAC word characteristic means that a range of outputs cannot be generated. In the context of the integrator this would mean that the integrator could not be tuned to a range of time constants. On the other hand, if M15 is shorter than the other transistors the characteristic in Fig. 3.15 C results, which is non-monotonic. While non-monotonicity is undesirable in some applications, here it is not, since the calibration scheme for the integrator measures the time constant vs. DAC word characteristic and stores the results in a look-up table. When a particular time constant for the integrator is desired, the DAC word that gives the time constant closest to the desire one is selected. This scheme does not rely on the measured time constants being in any particular order. Note that no range of unrealizable values of I_{OUT} or time constants results from M15 being shorter than the rest. Since there will inevitably be mismatch between transistors, but mismatch in one direction is more troublesome than in the other, the length of the series devices (M15 and M16) was chosen to be shorter than the length of the shunt devices, so that in the presence of mismatch, no gaps in the I_{OUT} characteristic could occur.

The DAC used in the integrator is the 10-bit version of the 3-bit DAC in Fig. 3.14.

3.3.3 VGA / 2-Input Multipliers

An overview of the VGA/2-input multiplier is shown in Fig. 3.16. A control signal (MULT), stored in the circuit's memory, determines whether the circuit behaves as a

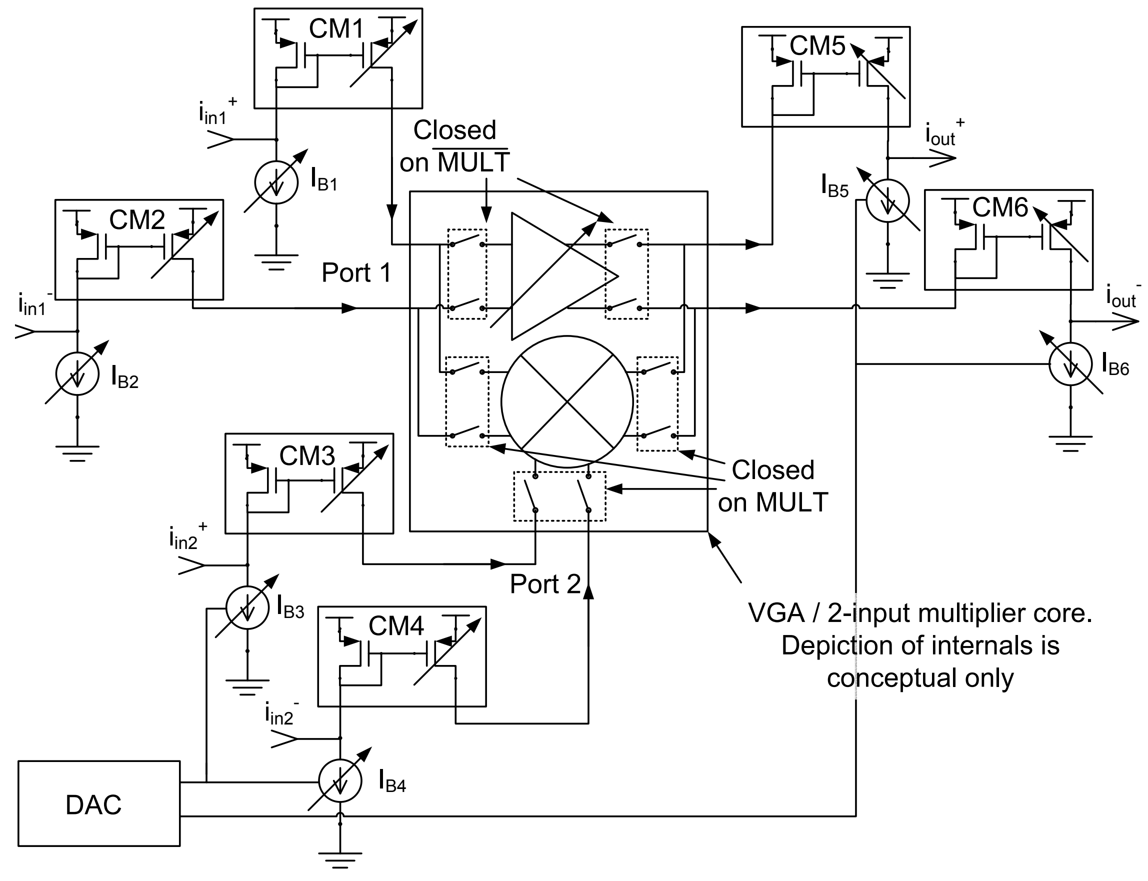

Figure 3.16: Top level diagram of the VGA / 2-input multiplier circuit.

2-input multiplier or a variable gain amplifier. The depiction of the internals of the core are for conceptual purposes only; it doesn't have separate circuitry for the VGA and the 2-input multiplier.

The input signals of the circuit are first processed by range-selectable current mirrors, which have the same gain settings as those in the input of the integrator. I_{B1} and I_{B2} are set to 20 μA when CM1/2 have gains of 20:1, 1 μA when CM1/2 have gains of 1:1 and 100 nA when CM1/2 have gains of 1:10. Hence, the bias component of the signal applied to Port 1 is 1 μA.

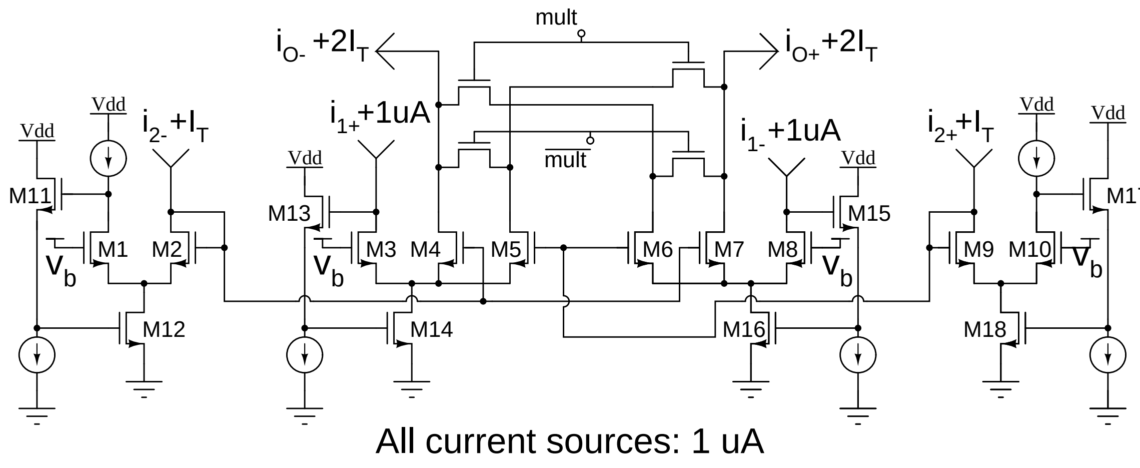

Figure 3.17: Core of the VGA / 2-input multiplier.

Port 2 operates in a somewhat different fashion. CM3/4 assume the same gains as CM1/2 but I_{B3} and I_{B4} are set to $\frac{I_T}{1\mu\text{A}}20\,\mu\text{A}$, $\frac{I_T}{1\mu A}1\,\mu\text{A}$ and $\frac{I_T}{1\mu A}100$ nA where I_T is a current generated by the DAC. For all settings, the bias component of the signal applied to the core of the circuit at Port 2 is I_T. This scheme, and that used for Port 1, allows for the circuit to process signals over a wide range while keeping small the range of currents over which the devices in the core of the circuit must operate. This is desirable since the core has devices that must remain weakly inverted, and as discussed in Sect. 3.3.2 this range is limited. The core of the VGA/two-input multiplier circuit [20] (Fig. 3.17) uses weakly-inverted MOSFETs (M1 though M10) in translinear loops in a fashion similar to that of the integrator. When the signal MULT is low, the drains of M4 and M5 are connected together, and so are M6 and M7; the circuit acts as a VGA with a gain of $2I_T/1\,\mu A$ from i_1 to i_o assuming i_2 is zero. This is depicted in Fig. 3.18. When MULT is high, the circuit becomes a

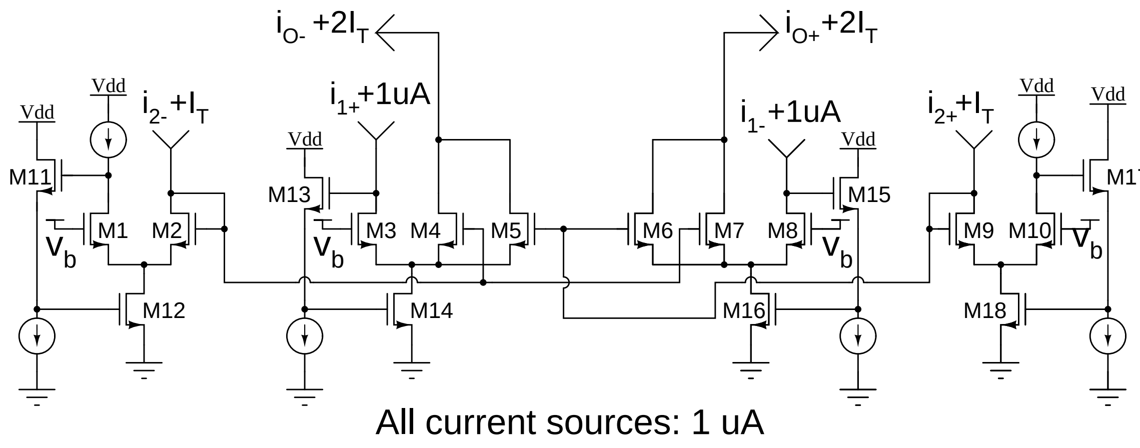

Figure 3.18: Core of the VGA / 2-input multiplier in variable-gain mode.

four-quadrant multiplier, as depicted in Fig. 3.19 with [20]:

$$i_o^+ - i_o^- = \frac{(i_1^+ - i_1^-)(i_2^+ - i_2^-)}{1\,\mu A} \tag{3.40}$$

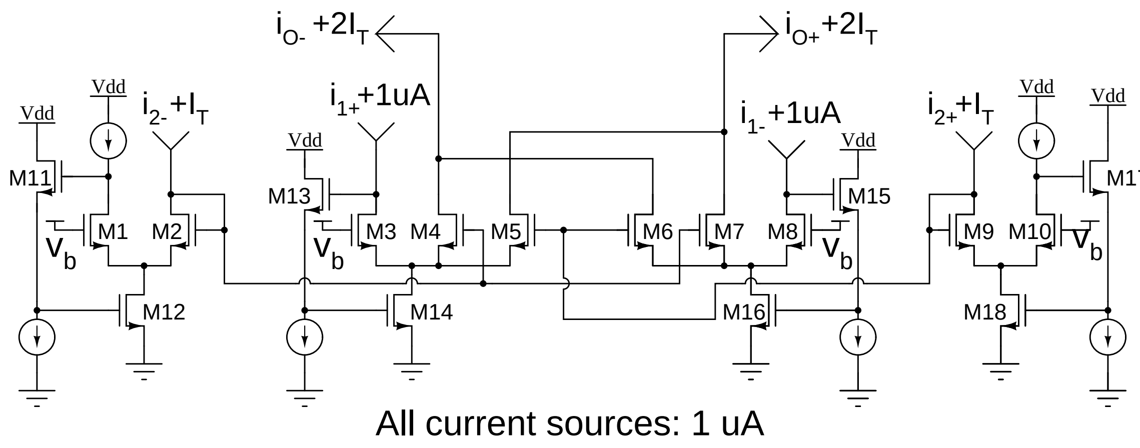

Figure 3.19: Core of the VGA / 2-input multiplier in multiplier mode.

The output current mirrors can be set to gains of 10:1, 1:1 and 1:20. $I_{B5/6}$ are set to $\frac{I_T}{1\mu A}100\,nA$, $\frac{I_T}{1\mu A}1\,\mu A$ and $\frac{I_T}{1\mu A}20\,\mu A$ for the three ranges, respectively.

The gain of the VGA is set through a combination of tuning I_T, for fine adjustments, and stepping the gains of the input and output mirrors, for coarse adjustment, allowing for finely spaced gains from 0.01 to 100.

3.3.4 Fanout Blocks

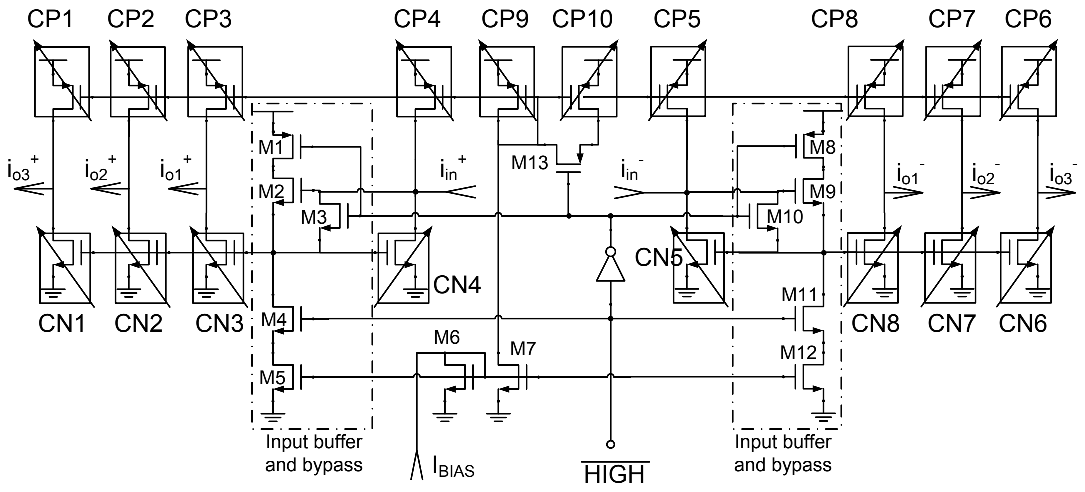

Figure 3.20: Schematic of the fanout circuit.

Fanout blocks are needed to apply copies of one signal to the input of several blocks. The schematic of one is shown in Fig. 3.20. Device names beginning with the letter "C" denote composite devices similar to that in Fig. 3.10. When the signal $\overline{HIGH}$ is low, the circuit's input range is 18 μA, transistors M1, M4, M8, M11 and M13 are off, M3 and M10 are on, and the circuit behaves like a simple current mirror. Fig. 3.21 shows a simplified schematic of the fanout block in this mode. Raising signal $\overline{HIGH}$, turns on M1, M4, M8, M11 and M13, and turns off M3 and M10. This activates the source-follower stages (M2 and M9). Fig. 3.22 shows a simplified schematic of the fanout block in this mode. These stages are used to shield the input of the circuit from the large gate capacitance of the four composite devices on each

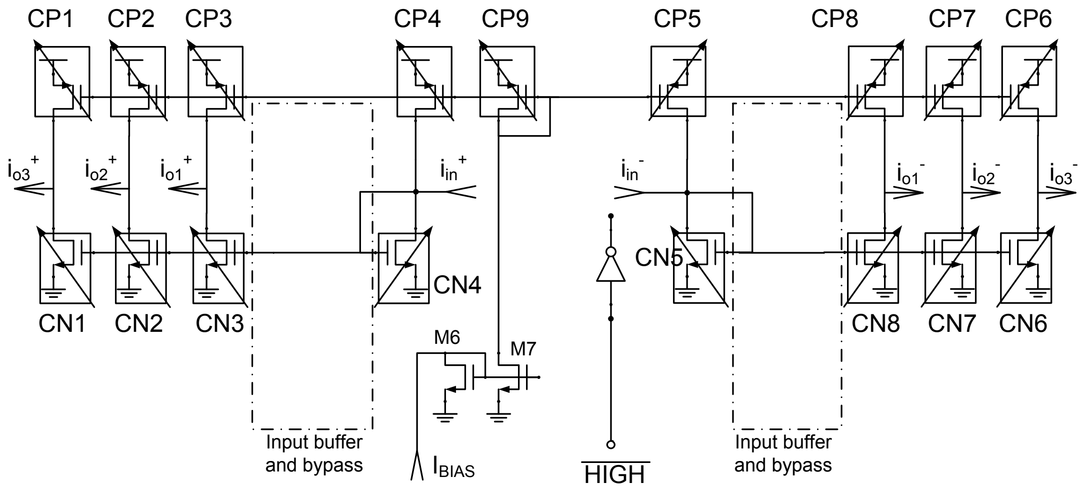

Figure 3.21: Schematic of the fanout circuit in its largest signal range.

side of the circuit. M2 and M9 are much smaller in gate area than the composite devices. When the input range is not its highest, the input resistance of the circuit increases such that putting it in parallel with the large gate capacitance would slow the circuit unduly.

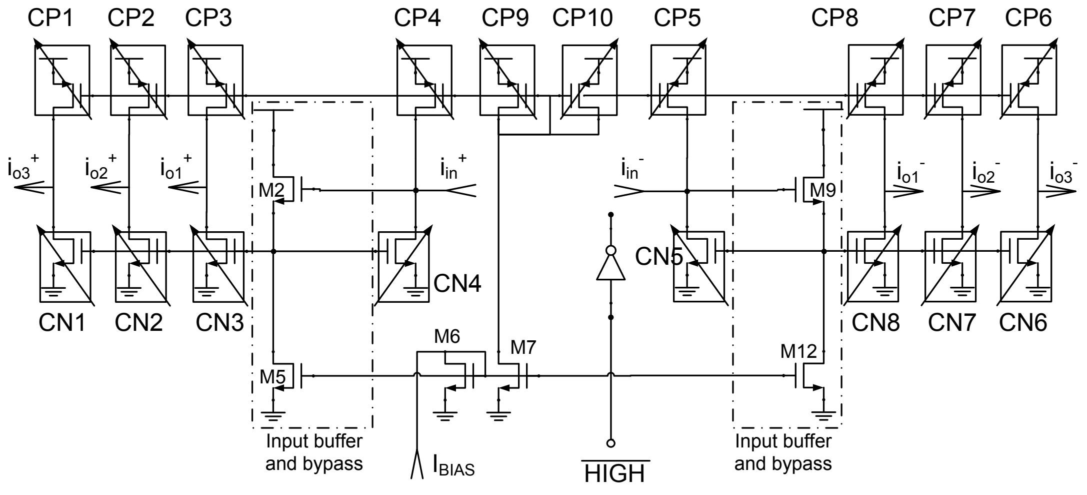

Figure 3.22: Schematic of the fanout circuit in is smallest and middle signal range.

If composite devices CN1 to CN8 and CP1 to CP8 are all in the same configuration as one another, the circuit has a gain from input to each output of 1. By configuring input composite devices differently from output composite devices non-unity gain can be achieved. The composite devices are controlled in groups of four. These groups are: CN4, CP4, CN5, and CP5; CN1, CP1, CN6, and CP6; CN2, CP2, CN7, and CP7; and CN3, CP3, CN8, and CP8. Within each quartet, all composite devices have the same configuration, however, each quartet may have a different configuration than the others. Gain is possible since the gate-source voltage of the input composite device (CN4) is the same as each of the output devices (CN1, CN2 and CN3), but their W/L ratios may be different and assuming the voltage at the outputs is high enough to keep the devices in saturation, current is proportional to the aspect ratio of the composite device. The possible range and gain settings are summarized in Table 3.3, with "l" denoting the lowest signal range, "m" denoting the medium signal range, and "h" denoting the highest signal range.

input range	output range	gain
l, 111 nA	l, 111 nA	1
l, 111 nA	m, 1 μA	9
l, 111 nA	h, 9 μA	81
m, 1 μA	l, 111 nA	0.11
m, 1 μA	m, 1 μA	1
m, 1 μA	h, 9 μA	9
h, 18 μA	l, 222 nA	0.012
h, 18 μA	m, 2 μA	0.11
h, 18 μA	h, 18 μA	1

Table 3.3: Fanout block range settings and limits.

3.3.5 Exponential

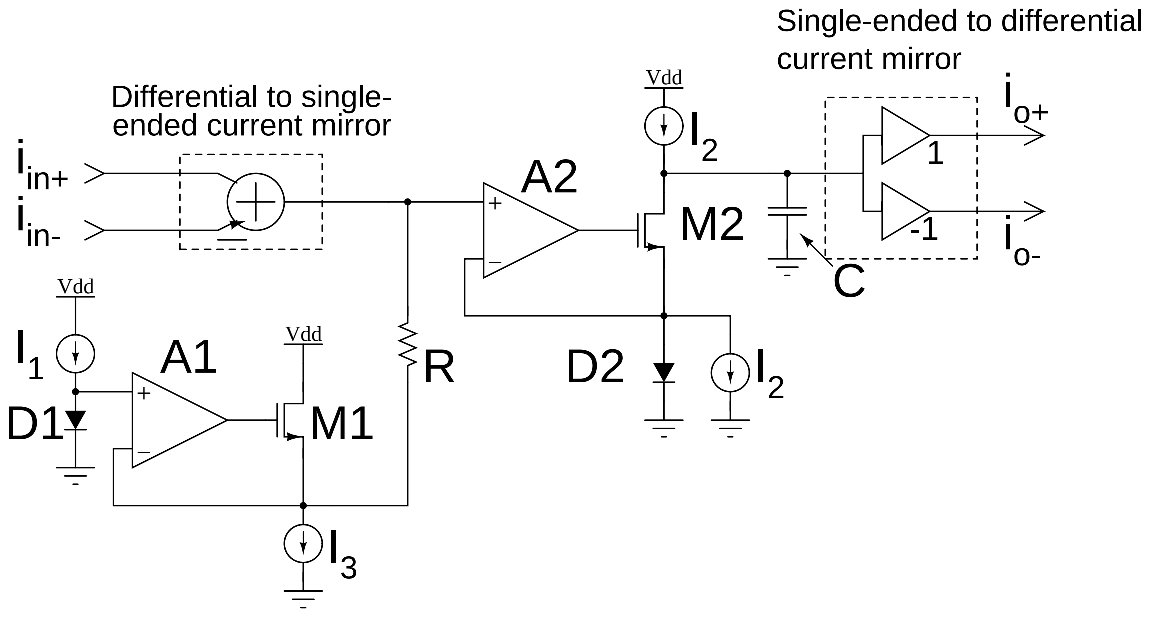
Single-ended to differential current mirror
Differential to single-ended current mirror
Vdd
I2
io+
io-
iin+
iin-
A2
M2
1
-1
C
Vdd
I1
A1
Vdd
M1
R
D2
I2
D1
I3

Figure 3.23: Exponential circuit.

A simplified block-diagram/schematic of the exponential circuit is shown in Fig. 5.12. The input to output behavior of the circuit is given by:

$$i_o^+ - i_o^- = 2I_1 \exp\left(R\frac{i_{in}^+ - i_{in}^-}{n\phi_t}\right) \tag{3.41}$$

where ϕ_t is the thermal voltage and n is the diodes' slope factor, assuming that the diodes match, and that the two current sources carrying I_2, which keep a minimum current flowing through M2, exactly cancel. The diodes are formed by the source/drain to well junction of a PMOS transistor. Amplifier offsets scale the output by a multiplicative factor only and their effect can be canceled through adjusting the gains of the blocks connected to the exponential block's input.

The generation of the exponential must be done on a single-ended signal, since $e^{(i_{in}^+ - i_{in}^-)} \neq e^{i_{in}^+} - e^{-i_{in}^-}$. The output single-ended to differential current mirror is a class-

AB circuit. Class-AB mirrors have the following advantages over a class-A mirror, assuming that the class-A circuit's bias has been selected to handle the largest signal expected.

- lower power dissipation, both when no signal is applied, and when the largest signal is applied.
- lower output current noise for small signals, since the current through the active devices is smaller.
- graceful degradation should the input exceed the maximum expected signal.
- smaller offset errors due to mismatch.

Most discussion of class-A versus class-AB circuits centers around the first three items. The last claim is explained in more detail below.

Consider a class-A current mirror designed to handle a maximum current of I_{MAX}. To do this, the devices are biased with that current, at least. Assume that the input and output devices' aspect ratio match one another's to 1 % and the input and output bias sources match exactly. Assume also that the only source of drain current mismatch is mismatch in their aspect ratios. When no input signal is present, the output offset of this mirror will be $0.01 \times I_{MAX}$. Now, consider a class-AB mirror intended to handle the same maximum current of I_{MAX}, and assume that input and output devices match in the same fashion. In this case, when there is no input present, the devices are conducting a much smaller current, I_{STAND}. Now the output offset will be $0.01 \times I_{STAND}$, which is smaller than for the class-A mirror since $I_{STAND} \ll I_{MAX}$.

Similar arguments can be made for the way in which output currents of a mirror will match one another in the case of a mirror with multiple outputs. Offsets due to mismatches between the threshold voltages of devices are also smaller for class-AB circuits than for class-A circuits, by similar reasoning.

Two disadvantages of class-AB currents mirrors are their increased complexity and their potential for instability. The former stems from the more complicated biasing network needed to establish the minimum current and the latter stems from a combination of there being more stages in the feedback loop and the wider range of loop gain over which stability must be guaranteed. Even a simple, class-A mirror using a diode-connected device is a feedback system. The input small-signal current develops a small-signal voltage across the r_o of the diode-connected device. This voltage is applied across the gate and source terminals of the device and induces a current to flow from drain to source. However, this loop is stable for any impedance connected to the diode node, with positive real part, assuming quasi-static operation of the devices. This assumption is valid for all but the highest frequencies of operation and seldom impacts the stability of a simple current mirror.

On the other hand, the greater complexity of a class-AB circuit can lead to instability. The schematic of the class-AB mirror used at the output of the exponential circuit is shown in Fig 3.24. It is based on a class-AB output stage, but with multiple outputs. To use it as a current mirror, the high-impedance node v_{IN} is connected to the current output i_{outfb} and whatever current flows in or out of i_{outfb} is copied to i_{out1} and copied, with an inversion, to i_{out2}.

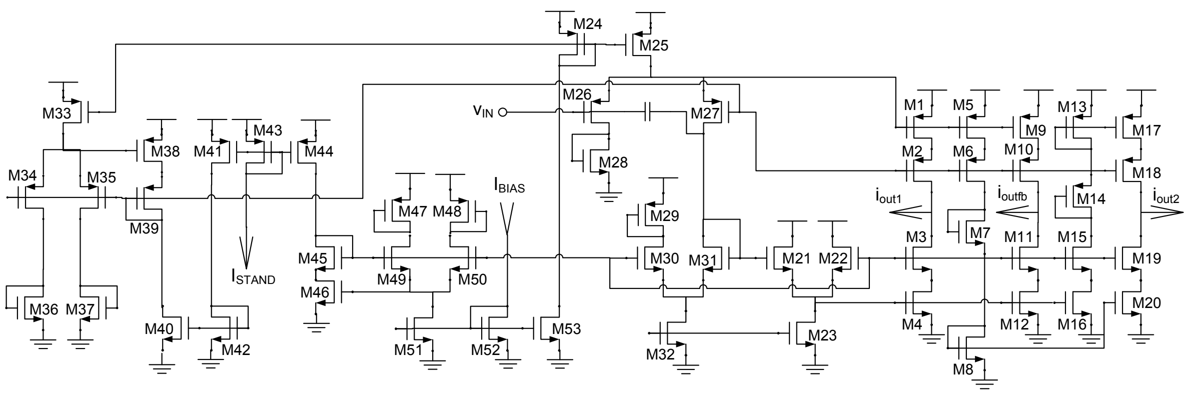

Figure 3.24: Schematic of the class-AB mirror used in the exponential circuit.

If terminal v_{IN} is at the same voltage as the gate of M27, transistors M1 through M20 are biased with a current of I_{STAND}. Discussion will focus on the output devices M9-M12, since they control the output current that is fed back to node v_{IN}. When v_{IN} is increased, M9 and M10 conduct less current and M11 and M12 conduct more. The range of output current is much larger than I_{STAND}. However, for the circuit to sink a large output current, the input signal must propagate through C, M31, M21 and M4. This path can have enough excess phase to make the circuit unstable when connected as a mirror, even with the feed-forward capacitor C. The circuit was stabilized by adding a grounded capacitor to the v_{IN}/i_{outfb} node in Fig. 5.12.

3.3.6 Logarithm

The logarithm circuit (Fig 3.25) uses a class-AB current mirror at its input, similar to that used at the output of the exponential circuit, that converts the differential input to a singled ended signal. This current is forced through a diode formed by the

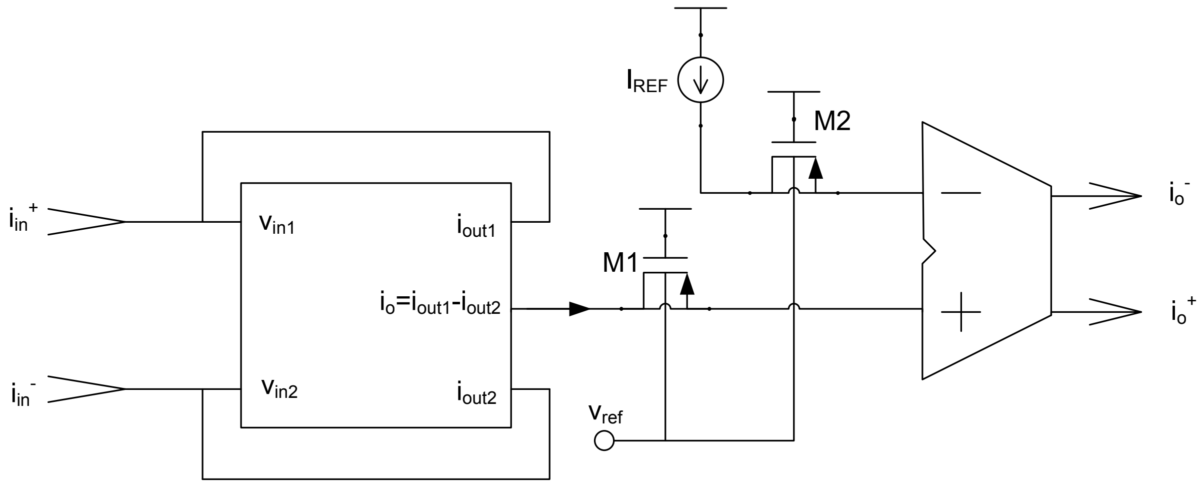

Figure 3.25: Schematic of the logarithm circuit.

source/drain to n-well junction of a PMOS device (M1). A linear transconductor, shown in Fig. 3.26, compares the voltage at the source/drain of M1 to the voltage at the source/drain of M2 and generates a differential output current. The reference current (I_{REF}) through M2 determines the input current that gives zero output current. The input to output behaviour of the circuit is given by:

$$i_o^+ - i_o^- = G_m n \phi_t \log\left(\frac{i_{in}^+ - i_{in}^-}{I_{REF}}\right) \tag{3.42}$$

where G_m is the transconductance of the transconductor, n is the slope factor of the diodes and ϕ_t is the thermal voltage (kT/q).

The transconductor is similar to [21], with input devices M11 and M19 biased in the triode region. Cross-coupled devices M10 and M18 cancel the charge injected through the gate-drain capacitance of M11 and M19 and improve the circuit's high-

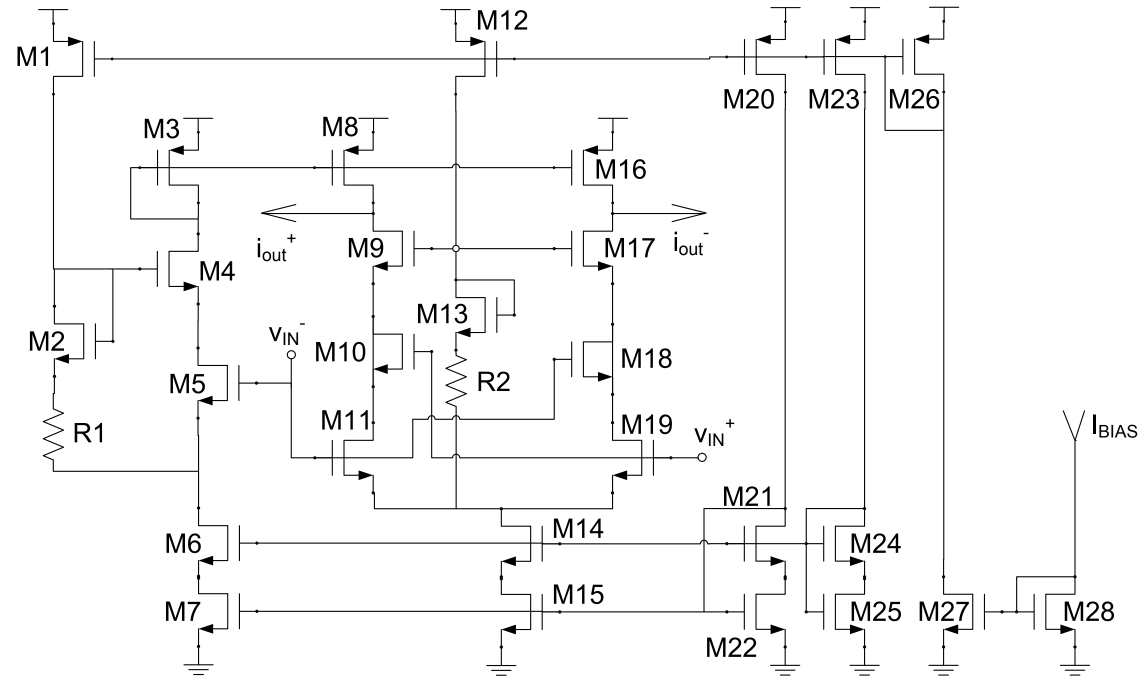

Figure 3.26: Schematic of the logarithm circuit's transconductor.

frequency performance. The transconductance (G_m) of the circuit is given by:

$$G_m = \frac{\partial(i_o^+ - i_o^-)}{\partial(v_{IN}^+ - v_{IN}^-)} = \mu C_{OX}\frac{W}{L}V_{DS} \approx \frac{I_{DS_{M8}}}{(V_{GS} - V_T)_0} \tag{3.43}$$

where V_T is the threshold voltage of the input devices, V_{GS} and V_{DS} apply to the input devices when the differential input voltage is zero, and μ, W, L and C_{OX} are parameters of the input devices. M1 through M7 generate the bias voltage for M8 and M16. This bias is not signal dependent, despite the gate of M5 being connected to v_{IN}^- since this input is tied to a DC voltage in the logarithm circuit. The current through M8 will be the difference between that flowing through M6 and M1.

Eq. 3.42 is only valid for positive values of $v_{IN}^+ - v_{IN}^-$. Whereas the logarithm function has a definition for negative (and complex) arguments, this logarithm circuit will saturate as its input is reduced, before the input reaches zero. The value to which it saturates is set by the bias current of the output transconductor.

3.3.7 Programmable Nonlinear Functions

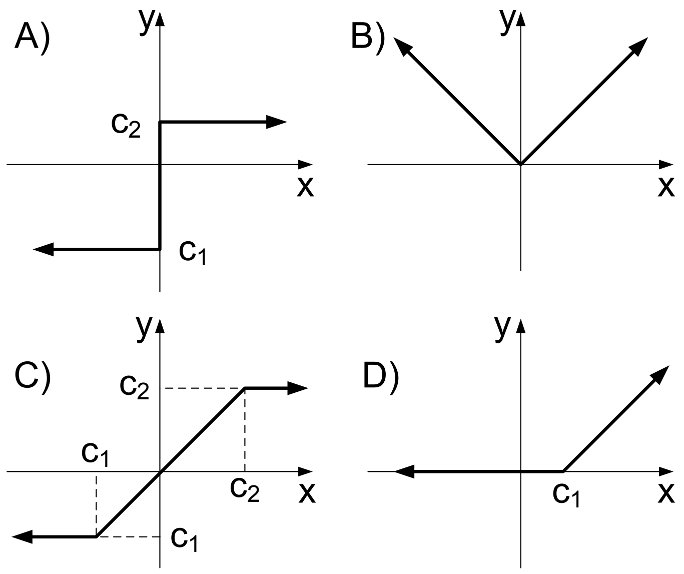

Figure 3.27: Nonlinear functions that can be implemented with one programmable nonlinear block.

There are four programmable nonlinear blocks within each Macroblock. Each can implement the following where x is the input and y is the output:

- sign: $y = c_1$ for $x < 0$ and $y = c_2$ for $x > 0$ where c_1 and c_2 are constants, programmable for each pair of blocks. (Fig. 3.27 A)
- Absolute value: $y = |x|$. (Fig. 3.27 B)
- Saturation: $y = x$ for $c_1 < x < c_2$; $y = c_1$ for $x < c_1$; and $y = c_2$ for $x > c_2$, where c_1 and c_2 are constants, programmable for each pair of blocks. (Fig. 3.27

C)

- Piece-wise linear basis function: $y = x - c_1$ for $x > c_1$ and $y = 0$ for $x < c_1$, where c_1 is a programmable constant. (Fig. 3.27 D)

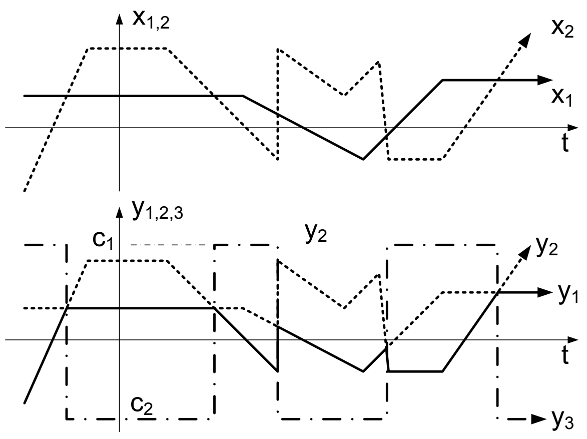

Figure 3.28: Minimum and maximum function.

Two neighboring blocks can be used together to implement the following where x_1 and x_2 are inputs and y_1, y_2 and y_3 are outputs:

- Minimum and maximum: $y_1 = min(x_1, x_2)$; $y_2 = max(x_1, x_2)$; $y_3 = c_1$ when $x_1 > x_2$ and $y_3 = c_2$ when $x_1 < x_2$, where c_1 and c_2 are programmable constants (Fig 3.28).

- Gate: $y_1 = x_1$ when $x_2 > 0$; $y_1 = 0$ when $x_2 < 0$; y_2 can be any one of the functions from the single block list; $y_3 = c_1$ when $x_2 > 0$ and $y_3 = c_2$ when $x_2 < 0$, where c_1 and c_2 are programmable constants.

- Sample and hold: $y_1 = x_1^*$ when $x_2 > 0$; $y_1 = 0$ when $x_2 < 0$; x_1^* is the value of x_1 when x_2 last transitioned from negative to positive; y_2 can be any one of

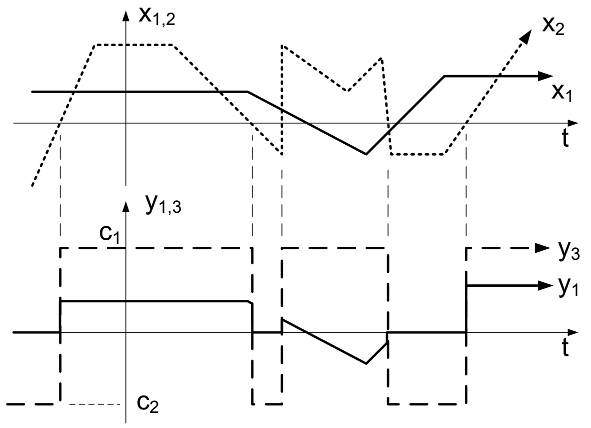

Figure 3.29: Gate function.

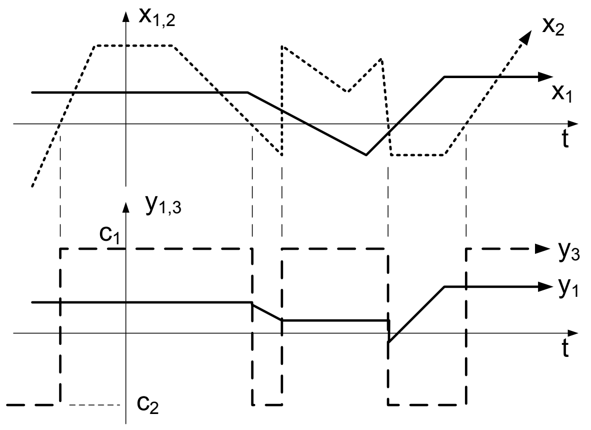

Figure 3.30: Sample and Hold function.

the functions from the single block list; $y_3 = c_1$ when $x_2 > 0$ and $y_3 = c_2$ when $x_2 < 0$, where c_1 and c_2 are programmable constants.

- Track and hold: $y_1 = x_1^*$ when $x_2 > 0$; $y_1 = x_1$ when $x_2 < 0$; x_1^* is the value of x_1 when x_2 last transitioned from negative to positive; y_2 can be any one of the functions from the single block list; $y_3 = c_1$ when $x_2 > 0$ and $y_3 = c_2$ when $x_2 < 0$, where c_1 and c_2 programmable constants.

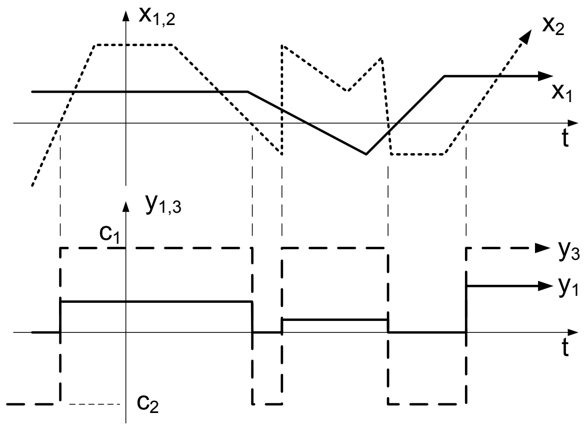

Figure 3.31: Track and Hold function.

Each nonlinear block consists of several current mirrors, current to voltage (I to V) converts, comparators, switches and some combinational logic (Fig 3.32). Two I to V convertors generate voltages that correspond to the currents generated by the DACs which are similar to those used in the integrator. These voltages are compared against those corresponding to the input current. The outputs of the comparators are processed by the combinational logic to determine when the various switches should be closed, generating the above single-block functions. Wherever the input-output characteristic has an abrupt slope change, the circuit is switching internally. For example, when the input signal to a block implementing the saturation function is between the lower and upper saturation limits a copy of the input signal is connected to the output. When the input level reaches the upper saturation limit, the input copy is disconnected from the output and replaced by a copy of the upper saturation limit. Mismatch between the input and its copy and the upper limit and its copy can lead to discontinuity in the input-output characteristic of the circuit. To limit

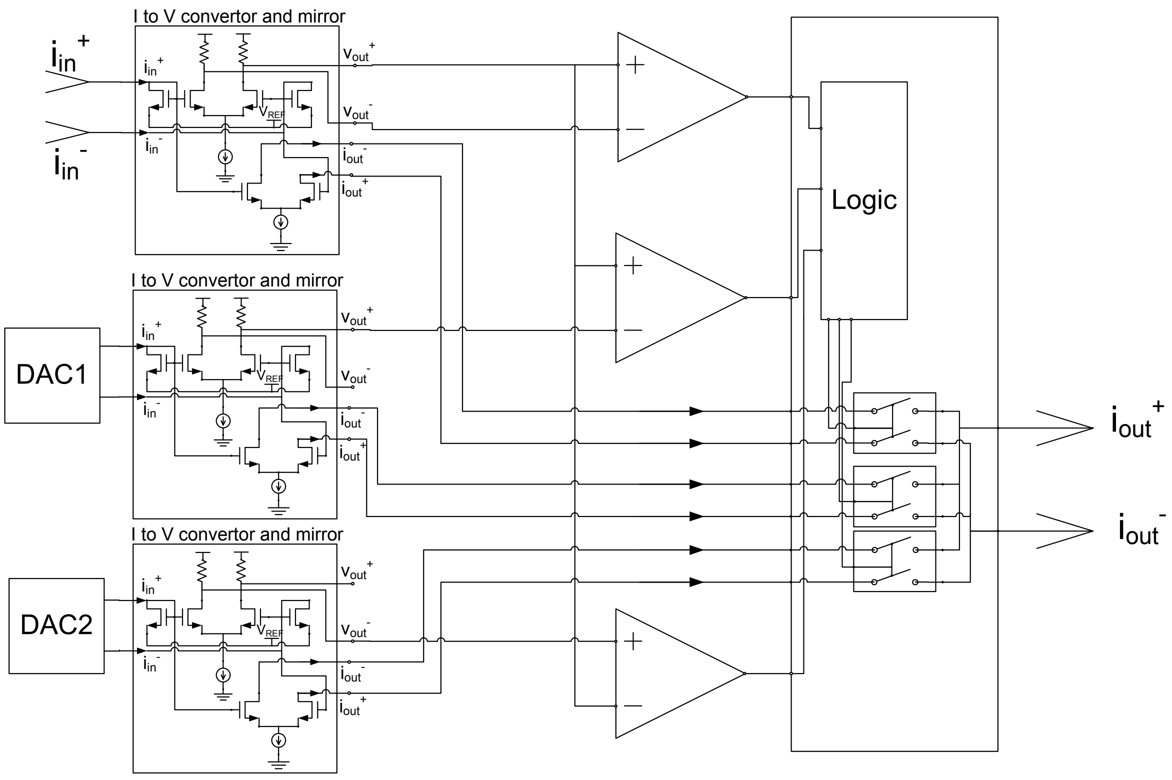

Figure 3.32: Schematic of one nonlinear block.

this, an attempt was made to select sizes for critical devices that limit mismatch in currents to 1 %. This in turn limits the size of discontinuities in the input-output characteristics to 1 %.

For the dual block functions, there are more comparators, more combinational logic, and a mirror in which the gates of the output devices are separated from the gates of the input devices by switches, allowing for the input current at particular times to be sampled.

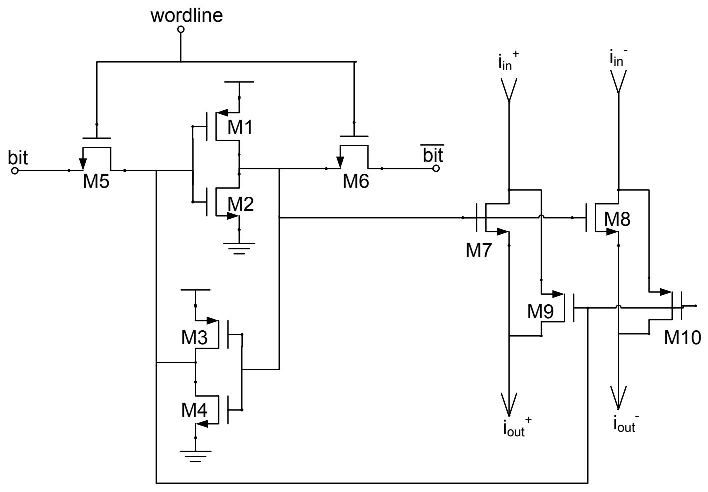

Figure 3.33: Schematic of one switch and SRAM cell.

3.3.8 Switches and Memory

As mentioned in Sect. 3.1 connections between blocks are made through complementary pass-transistor switches (Fig. 3.33) whose state is stored in an adjacent SRAM cell. M7 through M10 implement the switch. M1 through M6 form a 6-transistor SRAM cell which holds the state of the switch. Programming the cell, by raising *wordline*, when *bit* is pulled to ground will close the switch. Because the input voltage of blocks is near $V_{DD}/2$, complementary transistors help keep the on-resistance of the switch low. M7 and M8 have $W = 0.9\,\mu$m and $L = 0.24\,\mu$m and M9 and M10 have $W = 1.5\,\mu$m and $L = 0.24\,\mu$m. These dimensions are a compromise between very narrow devices which would reduce the capacitive loading on the inputs and outputs of blocks and wider devices which have lower on-resistance. It is important to realize that each path from the output of one block to the input of another is loaded by almost 90 switches, each containing $2.4\,\mu m$ of drain diffusion. Signals that must be routed from one macroblock to another are loaded by an additional 160 switches.

This switch/SRAM-cell is tiled in arrays wherever collections of input wires cross collections of output wires. However, between the output of a given block and its connection to the wire that enters the switching matrix is the circuit in Fig. 3.34.

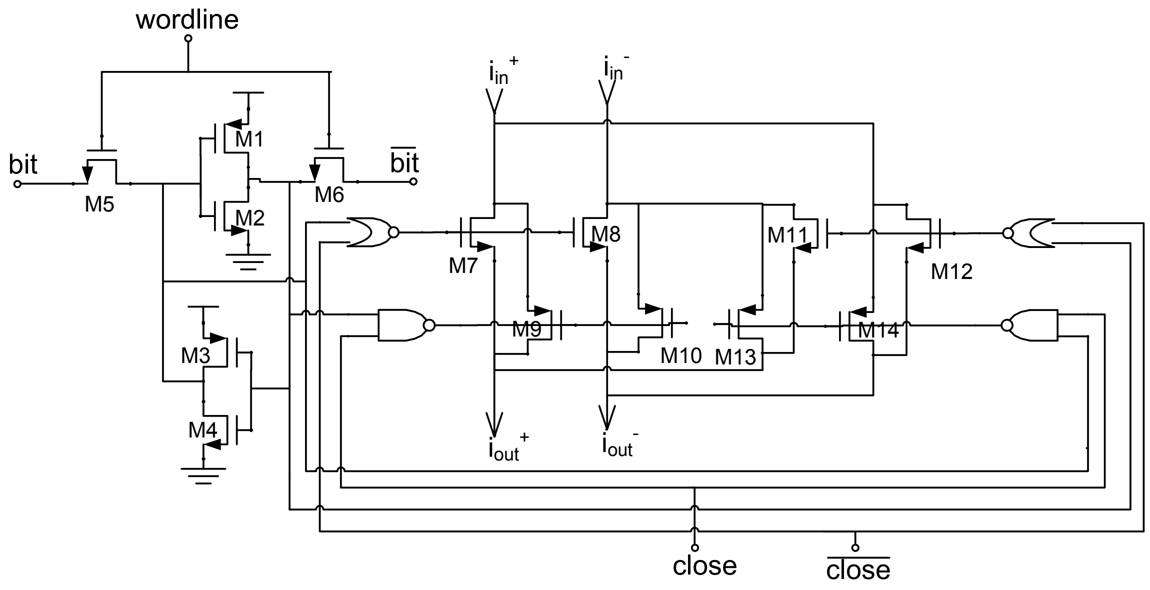

Figure 3.34: Schematic of output polarity switch and SRAM cell.

Based on the state of the SRAM-cell (M1 through M6) i_{in}^{+} is either connected to i_{out}^{+} or to i_{out}^{-} (Fig. 3.34). The latter is done to invert the signal. The signal *close* is a control signal that, when raised, connects the switch, with the polarity determined by the SRAM-cell's state. For reasons outlined in Sect. 3.4.3, the outputs of the blocks are connected and disconnected from the switch arrays at various times during the preamble to a simulation.

The SRAM cells are programmed using a standard scheme using clock (CLK) and write enable (WR_EN) signals. A wide fan-in NOR-based word line decoder converts 6 address bits to up to 64 word lines. When CLK is high, every *bit* and $\overline{bit}$ line is pre-charged to V_{DD}. When CLK is lowered, if the particular column has been selected, either *bit* or $\overline{bit}$ is pulled to ground by a footer circuit.

3.4 Chip Architecture in More Detail

3.4.1 Interconnection of Blocks

The functional blocks are organized within the macroblock as shown in Fig. 3.2. Within the macroblock, input wires for the blocks in a given row run above the row of blocks with the exception of the wires for the row of nonlinear blocks, which run below the row. This is different from how it is depicted in Fig. 3.2, as the latter is intended only for conceptual purposes. Most output wires run to the right of a given column of blocks with the exception of the wires connected to the outputs of the nonlinear blocks. The three outputs for the pairs of nonlinear blocks (not including the log/exponential blocks) run in the column in between the pair. From bottom to top, this scheme leads to the following count of input wires (pairs) in each row: 5, 10, 5, 11 since integrators each have one input, VGA/2-input multipliers each have two inputs, fanouts each have one input and the nonlinear blocks each have one input. From left to right the columns have the following numbers of output wires: 11, 8, 11, 8, 10. There are a total of 31 input wires and 48 output wires. Where each group of input wires crosses a group of output wires, there is an array of complementary pass transistor switches and SRAM cells which hold the state of the CMOS switches. A 3 by 3 array of these switches is shown in Fig. 3.35. To simplify the schematic, signals in the figure are single-ended and switches are only NMOS. A schematic of the actual SRAM-cell/switch used is in Fig. 3.33. The $\overline{bit}$ terminal of all of the cells (Fig. 3.33) in a given row are connected together, as are the *bit* terminals. Collectively, all of the signals that connect *bit* and $\overline{bit}$ terminals are known as bit lines. The *wordline* terminal of all of the cells in a given column are connected together.

The SRAM cells are controlled in a standard, clocked, programming scheme, with the notable difference between this and other memory applications that this is a write-only memory, as there is no need to read its state. Word lines run vertically, parallel to the wires connected to the outputs of blocks, and bit lines run horizontally, parallel to the wires connected to the inputs of blocks (Fig. 3.35). In terms of physical layers, the horizontal input wires run on metal-5 (M5), output wires on M4, word lines on M2 and bit lines on M3.

The state of a switch is controlled using the following sequence:

- With all word lines low, all bit lines are pre-charged to V_{DD}, whenever both a clock signal and a write enable signal are high.
- When the clock signal falls, the switch's word line is raised, closing M5 and M6 (Fig. 3.33).
- Also when the clock signal falls, either the *bit* signal or the $\overline{bit}$ signal is pulled low by a "footer" circuit, causing the state of the cell to be set.
- When the clock signal rises, the word line signal falls, disconnecting the cell

from the bit lines.

There are 31 inputs within a macroblock and 31 pairs of bit lines running parallel. The state of the switches is programmed by an 8-bit word, meaning that the 31 bit lines are divided into 4 groups (3 groups of 8 and 1 group of 7), selected by two address bits. The footer circuitry is such that writing a 0 to the cell pulls the *bit* side low and closes the switch. Each word line corresponds to a particular output and typically each output is connected to only one input. This means that only one of the bits in the 8-bit word is a 0.

If the bit lines ran parallel to the outputs and multiple outputs in the same group of 8 were connected to one input, more than one bit in the eight bit word would be a zero. To properly generate the programming data for the memory, multiple connections must be considered simultaneously, which complicates programming. In the scheme that is used, multiple outputs connected to one input requires that multiple word lines go high in separate programming words. This means that only one connection needs to be considered at a time.

Wherever a group of input wires (numbering 5, 10, or 11) crosses a group of output wires (numbering 8, 10, or 11) there is an array of switches, making for 9 possible array sizes. To reduce the number of blocks needing to be laid out, only three array sizes were used: 5 by 11, 10 by 11 and 11 by 11, where the first number is the number of rows (i.e. number of input wires to connect) and the second number is the number of columns (i.e. number of output wires to connect). In several instances, only 8 columns are needed, meaning that three columns are unused. Of 55 columns of cells in the memory arrays, only 48 are needed. The improper connection of the unused columns of switches led to the most significant design flaw on the chip, which is described below.

In addition to the arrays of switches wherever groups of input wires cross groups of output wires, there are more complicated switches (Fig. 3.34) that separate each output from its vertical wire in the grid. Each of these output switches can connect its input side to its output side directly, or with cross-coupling. This allows signals to be inverted, and possibly subtracted from others. These switches take a signal, *close*, which, when high, closes the switch with the polarity determined by the state of the SRAM cell. When *close* is low, the switch is open. These switches are connected such that their word line is connected to the word line associated with the column in the switch arrays corresponding to their output wire, so that the output switch can be set whenever a connection is made. The bit lines of these output switches are connected together and are controlled by a separate footer circuit. The complete programming word is 9-bits, consisting of the 8-bits discussed above and the single bit that controls the polarity of the output's connection. A signal passes through two switches to get from the output of one block to the input of another when the two blocks are in the same macroblock.

The input and output wires within each macroblock extend outside the macroblock, allowing for connections between blocks in different macroblocks. Below each row of macroblocks are 16 pairs of wires over which the output wires of blocks within the macroblocks extend. Each output can be connected to any of these 16 wires through a 16-row, 48-column array of switches and memory. Beside each column of macroblocks are 16 pairs of wires over which the input wires of blocks within the macroblocks extend. Each input can be connected to any of these 16 wires through a 31 row, 16 column array of switches and memory. Wherever the groups of 16 horizontal wires cross a group of 16 vertical wires is a 16 by 16 array of switches. These external switches span 64 columns and 47 rows. Together, they can be thought of as 47 (31+16) row, 64 (48+16) column array, with a section (31 rows, 48 columns) removed. The thick, dashed line in Fig. 3.3 shows how a signal is routed from the output of a block in macroblock W to the input of a block in macroblock Z. The three 'X's indicate three switches that are closed to allow this connection. The signal must also pass through the output switch associated with the block inside marcoblock W, for a total of four switches in the signal path.

Each end of the vertical and horizontal wires between macroblocks can be connected off chip. The horizontal wires are used to route signals from blocks on-chip to off-chip. At each end of the 16 wires are 16 switches. Adjacent switches have their output sides connected together. Each switch's input side is connected to one of the 16 wires. This schemes lets the pair of switches act as a 2-1 multiplexor. The 8 output sides are connected off-chip. Likewise, there are groups of 16 switches at the top and bottom of the groups of vertical wires which allow connections from off-chip to be made to the inputs of the chip's blocks. These are connected such that adjacent switches' act as 1:2 demultiplexors with input sides connected together and each output side connected to a vertical wire.

Memory Design Flaw

Two choices are possible for how the word lines could be controlled. In the first, all 55 word lines could be addressed and controlled, while in the other, the 7 word lines connected to the extra, unused columns of cells could be disabled, leaving only 48 word lines. Both schemes require 6 bits to address, however, circuits for an earlier chip that had 48 word lines had been designed and were used; thus the latter scheme was used. The cells in the unused columns of switches/SRAM cells were each connected in the following way:

- The input side of the switch was connected to the input wire, and hence to the input of a block.
- The output side of the switch was connected to ground.
- The bit lines of the SRAM cells were connected to the bit lines used for the active SRAM cells.
- The word lines of the SRAM cells were connected to ground.

This approach has the flaw that the SRAM cells in the unused columns cannot be altered. While it is not necessary to program these cells to close switches, this scheme doesn't allow them to be reset properly when the chip is powered on. An SRAM-cell consists of two back-to-back inverters (M1/M2 and M3/M4 in Fig. 3.33), and is a bistable system. If the circuit had perfect symmetry, the state into which it settled when the chip is powered on would be randomly determined by thermal noise, with a 0.5 probability for each state. If an unused SRAM cell/switch powers up (when the chip is powered on) in the closed state, the input wire connected to the switch is shorted to ground since the output side of the switch is connected to ground. With 7 unused switches connected to each input wire, all 7 SRAM cells would have to power up in the OFF state for an input to not be shorted to ground. If the two states of a given SRAM cell were equally likely, this would occur with a probability of $(\frac{1}{2})^7 = \frac{1}{128}$. Clearly, if the power-on behaviour of the memory behaved in this fashion almost no block would be functional, since there would be a $\frac{127}{128}$ probability that its input was shorted to ground.

In HSPICE and Nanosim simulations, all blocks were functional, and this error was not detected. This suggests that some electrical condition exists that favours the OFF state over the ON state when the chip is powered on. The capacitance between the input/output wires of the switches and the *bit* side of the SRAM cell is larger than that to the $\overline{bit}$ side of the SRAM cell. On power-on the inputs of the blocks will start to charge up to a voltage determined by the input current mirrors of the block, assuming that they are not shorted to ground by an SRAM cell. It is speculated that this increase in voltage is capacitively coupled more strongly to the *bit* side and nudges it sufficiently that the *bit* side reaches V_{DD} and the $\overline{bit}$ side is pushed to *gnd*. The coupling is stronger because the PMOS devices are wider than the NMOS devices. For this trend to hold up in practice, the actual waveforms on start-up would have to be the same as those in simulation, and this nudge needs to be larger than the random noise in the system, absent from HSPICE and Nanosim simulations, which might offset this push. The prevalence of this problem and solutions to it are discussed in Chapter 5.

3.4.2 Functional Block Memory

Every functional block, with the exception of the logarithm blocks, has some memory for holding range settings, DAC input words, and other parameters. The memory of each block is a collection of level-sensitive gated latches. Input data come from the same data lines as are used to program the switch memory. The output of a 6-input AND gate, implemented by a combination of 2-input gates, activates the latches. 5 address lines, and their complements, are routed around each macroblock. 5 inputs of the AND gate are connected either the true or complement polarity of each address signal, while the sixth signal is connected to a control signal, which is toggled to latch in the data.

3.4.3 Simulation Control

This section describes control of the chip after the states of the switches and the block-memory has been programmed. The following signals are involved in simulation control:

- CAP_CON: This corresponds to the signal V_{CAP} in Fig. 3.5. During the preamble to a simulation, this signal is high. It is low during a simulation.

- SIM: All of the *close* signals associated with the output switches for the integrators are connected to this signal. During the preamble to a simulation, this signal is low. Raising it begins the simulation.

- CON_INP: All of the *close* signals associated with the output switches for every block except the integrators are connected to this signal. During the preamble to a simulation and during a simulation, this signal is high. It is lowered when integrators are reset.

- S_IN: This corresponds to the signal V_2 in Fig. 3.4. This connects the integrator in feedback during offset cancellation. It is high during part of the preamble to the simulation and low during the simulation.

Typical waveforms for the four simulation control signals are shown in Fig. 3.36. One progression through a simulation sequence is described below, beginning at the falling edge of SIM:

- The signal CON_INP falls T_7 after SIM falls. The requirement on T_7 is $T_7 \geq 0$.

- The signal CAP_CON is raised T_8 after CON_INP falls, resetting the integrators. The requirement on T_8 is $T_8 \geq 0$.

- T_1 is the duration over which CAP_CON is high. This must be long enough to reset the integrators. The time constant for resetting them is much smaller than their integration time constants, since the resistance through which each capacitor is charged or discharged is much smaller than the $1/g_m$ of the transistors in the integrator's signal path. T_1 needs to be longer than a few microseconds, but is likely longer than that because it is during this interval that the chip is programmed.

- T_2 is the interval between CAP_CON falling and CON_INP being raised. The requirement on T_2 is $T_2 \geq 0$.

- T_3 is the interval between CON_INP being raised and S_IN being raised. The latter starts the offset cancellation procedure. The requirement on T_3 is $T_3 \geq 0$.

- T_4 is the duration of the offset cancellation procedure. During this interval, the integrators are placed in unity-gain feedback. During $T_2 + T_3$ the output of the integrator may have drifted. T_4 needs to be longer than a few time constants of the integrator with the slowest time constant. Integrators have a nominal time constant of 40 μs. If the output of the integrator had drifted (because of an input offset) to the point of saturating the integrator to its fullscale output, it would take 7 time constants for its output (and input since it is in unity-gain feedback) to reach < 0.001 of full scale. For integrators with nominal time constant, $T_4 \geq 280\,\mu s$. This would reduce the input offset to be no more than 0.1 % of fullscale. One could wait longer; however, the nonidealities of the scheme (charge injection and capacitance division) prevent the offset from being smaller than about 0.1 %. Lengthening T_4 does not improve offset cancellation.

- T_5 is the interval of time between the end of the offset cancellation phase and the connection of the output of the integrators to the rest of the system. Ideally, this interval of time should be small; this prevents the outputs of the integrators from drifting significantly due to their residual input offsets.

- T_6 is the duration of the actual simulation. Shortly after SIM is raised, the inputs of the system are applied and the outputs are observed. In the case of systems that are nominally low pass, it may be prudent to delay applying the inputs to the system so that the response of the system to residual offsets can be observed, allowing this effect to be subtracted out from the total response of the system.

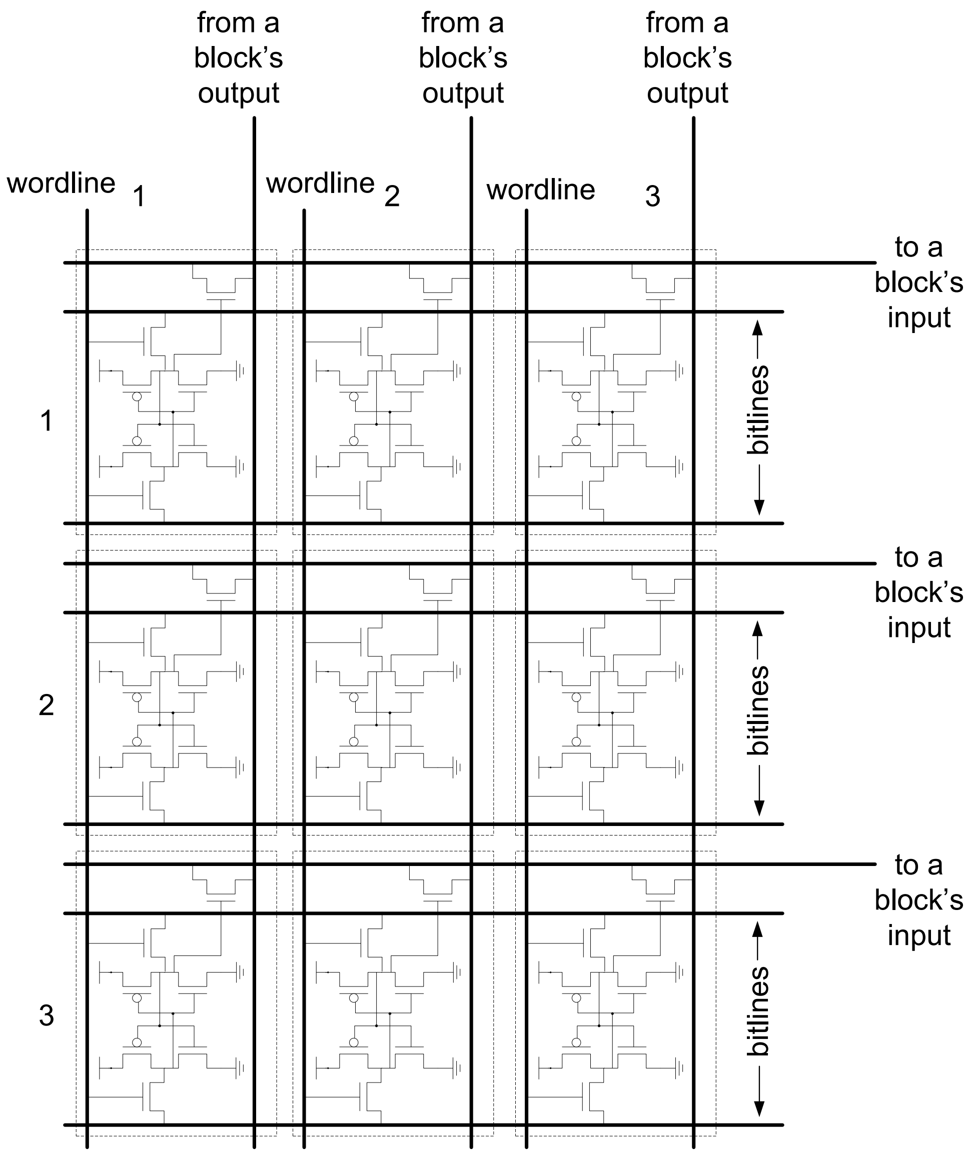

Figure 3.35: 3 x 3 section of the SRAM / switch grid.

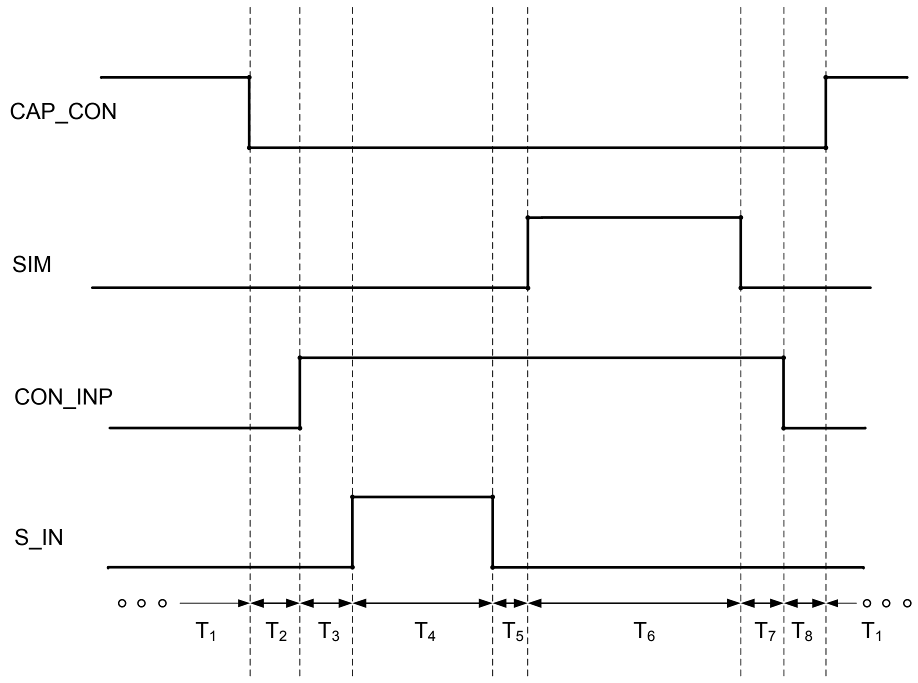

Figure 3.36: Typical waveforms of simulation control signals.

Chapter 4

Analog Computation Environment

4.1 Overview

A brief overview of the analog computation environment can be found in Sect. 1.1 and in Fig. 1.1. Critical aspects of the computation environment are to allow:

- The tight coupling of analog and digital portions.
- Rapid programming of the analog portion.
- Pre-simulation calibration of the analog portion.
- Close monitoring of signal levels.
- The connection of measurement equipment.
- The display of results.

The rest of this chapter describes the implemented system. Suggestions for enhancements are found in Ch. 9.

4.2 Implemented Environment

In the current environment a user controls the analog computer in the following way:

- The user draws a block diagram in Matlab's Simulink. Simulink creates a model file, with the ".mdl" file extension.
- Matlab programs interpret the ".mdl" file that Simulink produces and generate the necessary programming parameters for the analog computer.
- Matlab programs, via a data acquisition card and some interface circuits on a PCB, program the connectivity and various parameters of the analog blocks.
- The data acquisition card generates inputs and measures outputs for the chip.
- Matlab programs display the results.

A user selects blocks from a separate Analog Computer Library, which have the necessary options to properly configure the analog computer's blocks. Additionally, they have the necessary behavioural workings to allow them to be used in Simulink simulations. This is useful for the purpose of characterizing the chip, since a user can draw a block diagram, simulate it using both Simulink and the analog computer, and compare the results. To enable the simulation in Simulink, the Sum block must be used to perform addition rather than simply connecting outputs together, since

Simulink happens to correspond closely to a voltage-mode representation. To allow the simulation on the AC, the user (at present) must explicitly use the Fanout block to enable the output of one block to be connected to the inputs of multiple blocks. Each Simulink block is mapped to its on-chip counterpart through its instance number. The instances are enumerated in the following fashion:

- If there are N blocks of a particular type of block in each macroblock, the macroblock in the top, left (0^{th} row and 0^{th} column) of the chip contains instances 0 to $N - 1$.
- Column indices increase moving from the left side of the chip to the right side.
- Row indices increase moving from the top side of the chip to the bottom.
- Block instances are counted by moving across the row before moving to the next row.

For example, Fanout120 (there are 10 Fanout blocks per macroblock) is the 0^{th} Fanout block in the macroblock in the 3^{rd} row and 0^{th} column (bottom, left macroblock).

Matlab programs generate the necessary bit stream to program the connectivity of the blocks and the parameters within each block. They remove the Sum blocks, which are unnecessary on the AC owing to the circuits being current-mode.

The data acquisition card is a National Instruments PCI-MIO-16E-1. It allows for a maximum of 8 differential inputs with 12-bit resolution, 2 analog, single-ended outputs with 12-bit resolution and 8 digital I/Os. Input sampling occurs at a maximum rate of 1.25 MSamples/s. However, the per-channel rate is, at most, 1.25 MSample/s divided by the number of active channels. When multiple input channels are active, sampling takes place in a round-robin fashion, meaning that samples are taken from each channel consecutively. Output sampling occurs at 1 MSamples/s on both channels, synchronously. The card is connected to the chip's printed circuit board (PCB) via a 68-pin cable.

The small number of digital outputs (8) of the data acquisition card, and the large number of digital inputs of the chip (> 32) requires the use of a serial to parallel interface, consisting of several off-chip serial-in, parallel-out shift registers. Digital data ($a < 1 : 15 >$, $d < 0 : 15 >$, RST, WR_EN) are clocked into the shift register chain. The data acquisition card's digital outputs are used for the following:

- $DATA\_IN$: This is the data input for the shift register.
- C: This is the clock for the shift register.
- $a < 0 >$: This is the $a < 0 >$ signal for the chip's block memory. See Chapter 3.
- CLK: This is the clock signal for on-chip memory.
- S_IN and S_OUT. The signal for the offset cancellation scheme in the integrator is actually composed of two signals, one of which controls M3 and the other which controls M4 in Fig. 3.4.
- CAP_CON: Reset signal for the integrators.

- CON_INP: Signal to connect outputs of blocks, other than integrators, to the switch grids.

The signal SIM is controlled by one of the data acquisition card's analog outputs. This is because it is important to ensure proper synchronization between SIM and the other analog output V_{IN} which serves as the forcing function for the system being simulated. Typical waveforms for the control signals are found in Fig. 3.36.

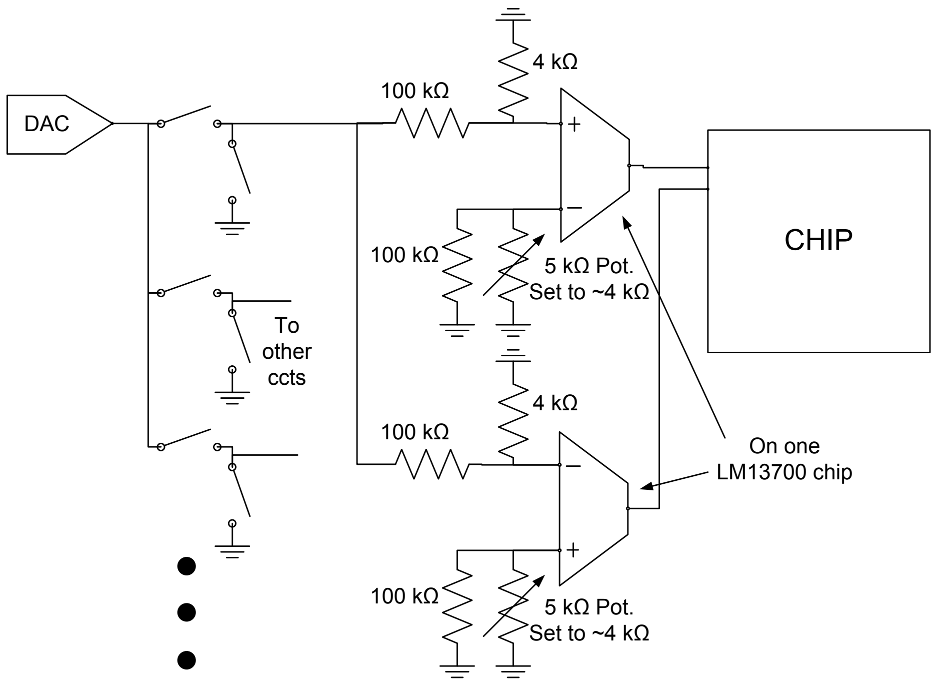

Figure 4.1: Transconductor circuit for generating analog inputs.

The chip's PCB has eight LM13700 transconductor chips, which each have two differential-in single-ended-out transconductors. The outputs of each pair of transconductors is connected to an input port of the chip. The input to the transconductor-pair can be connected to V_{IN}, to ground, or left open (Fig. 4.1). Variable resistors allow the offsets of the transconductors to be manually canceled.

The two polarities of output current from seven output ports are converted to voltages by LM359 operational amplifiers with resistive feedback, acting as transresistance amplifiers. Seven inputs of the data acquisition card measure these differential voltages. The 8^{th} input of the card can measure the output current of a block's DAC when it is routed off-chip.

4.2.1 Calibration

Before a simulation has been run with a particular chip, the time constants of integrators and the gains of VGAs are measured for each integrator and VGA for a finely spaced sequence of DAC settings, and the results are stored for future recall in look-up tables. When a simulation is run, the gains of VGAs and the time constants of integrators are set. When a particular gain or time constant is programmed, the appropriate DAC setting is selected by interpolating between the values stored in the look-up table. It was noted in Chapter 3 that the DACs were designed to be non-monotonic. However, the amount by which the series transistors were scaled introduces non-monotonicity only every 8 levels. For example, the DAC's current may be smaller for word 32 than 31, but its output increases monotonically between 32 and 39. For this reason, the behaviour for the VGAs and integrators was measured for DAC words in the sequence 7, 8, 15, 16, 23, 24 . . . . . Algorithms assume that both the τ of integrators and the gain of VGAs are linear over the 7 bit range between consecutive entries in the look-up tables.

The multiplication constants of the multipliers is measured during calibration.

When a multiplier block is instantiated in a Simulink model, its multiplication constant is included in the model, allowing a user to correct for it. That is, if a user wants a multiplier that implements $y = x_1 * x_2$ but the instantiated model implements $y = \frac{1}{2}(x_1 * x_2)$ the Simulink block will indicate this allowing the user to follow the multiplier by a VGA with a gain of 2.

Chapter 5

Circuit Measurements

5.1 Integrator

5.1.1 Measurement Set-up

The various configurations used to test the integrators are shown in Fig. 5.1. These diagrams are drawn taking into account that the blocks are current-mode. Therefore, the connection of two lines at the input of a block indicates that two signals are added and applied to the input. For example, in set-up E an output of F_1 and an output of F_2 are added and applied to the input of the integrator under test (IUT). The "-" sign below the feedback path in set-ups C, E and F indicates that the connection was made with negative polarity. That is, in set-up E, the output of F_2 is inverted before being added to the output of F_1. Because the blocks are current-mode, the input of the IUT in set-up B is not floating; it has a signal of 0 applied to it, analogous to a voltage mode input being grounded. Blocks ADC and DAC are part of the data

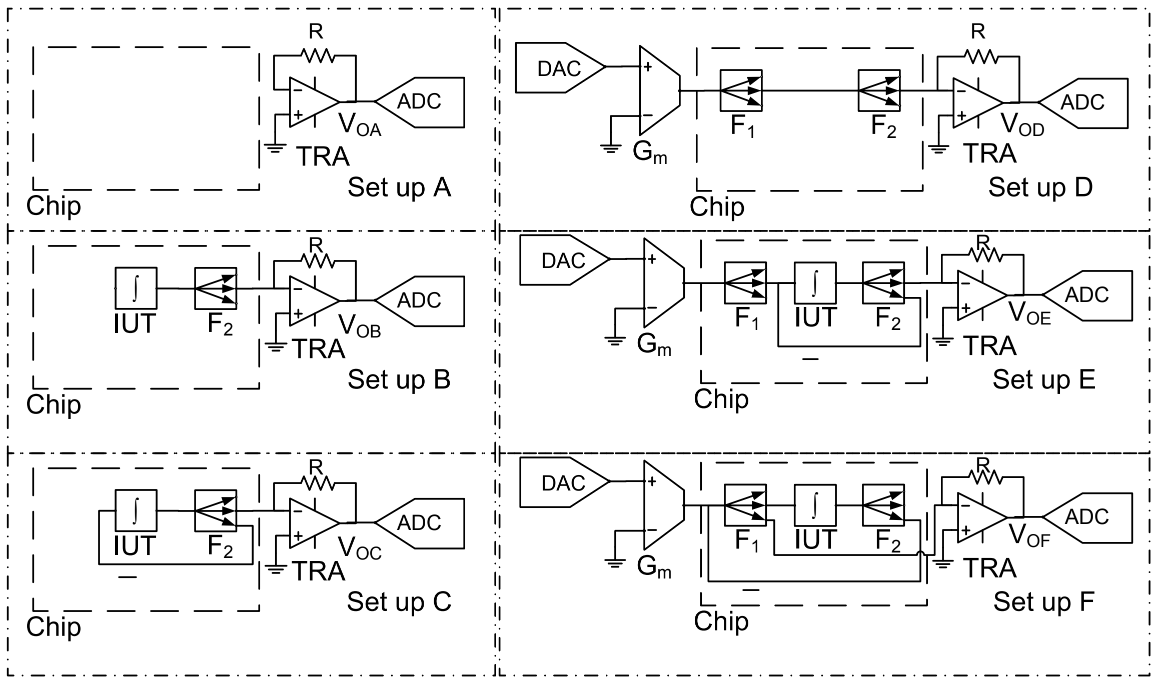

Figure 5.1: Test set-up for measuring Integrator blocks.

acquisition card used in the test set-up. Transresistance amplifiers (TRA) are needed because the ADC measures voltage while the chip's output signals are currents. The circuits on chip are all differential, meaning that there are actually two TRAs and the ADC measures the difference between their two outputs. Similarly, there are two transconductors connected to the DAC, applying opposite currents to the blocks on-chip. For simplicity, the blocks are drawn in single-ended fashion.

Blocks F_1 and F_2 are fanout blocks. F_2 is needed to connect the IUT in feedback. The input of F_1 and F_2's middle output are set on the high range. The middle output of F_1 and input of F_2 are set on the same range as the integrator. In set-ups C and E the bottom output of F_2 is set on the same range as the IUT while in set-up F it is set on the high range. The bottom output of F_1 is set on the high range. This scheme allows the integrator to be tested over its different signal ranges, while the test equipment always applies and measures large signals corresponding to the high range. This is important because the off-chip capacitances are larger than those on chip. Loading the input of a block with such a large capacitance can slow the block's response when the input is set to one of the lower signal ranges, owing to the larger input resistance of the block on a lower signal range. An explanation of each type of measurement follows.

Offsets

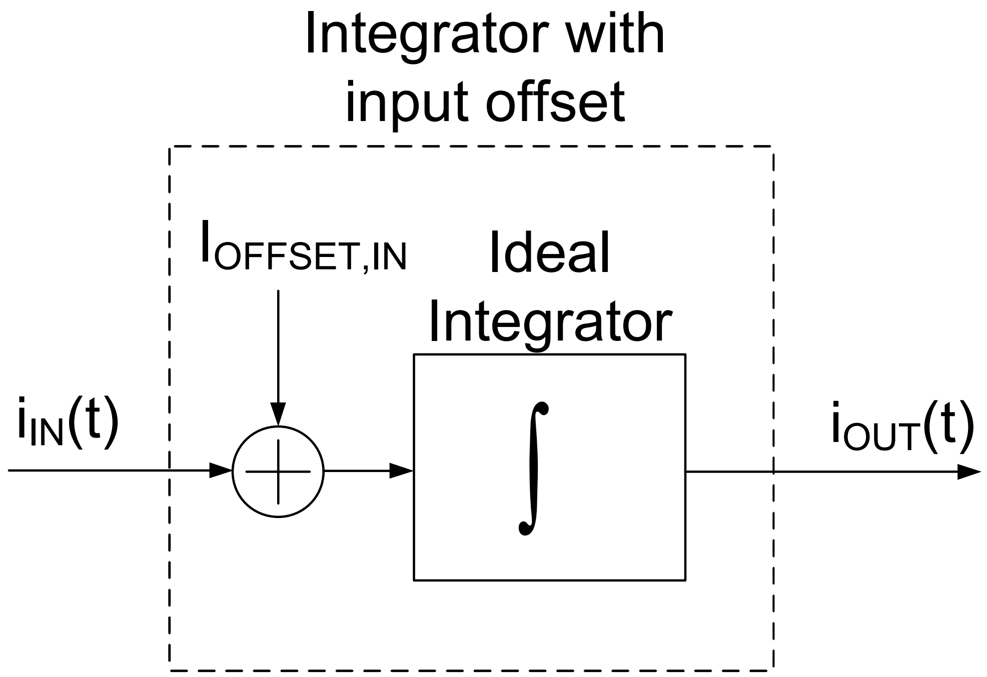

Figure 5.2: Integrator with an input offset.

Fig. 5.2 shows a model of an integrator with an input offset, $I_{OFFSET,IN}$. When $i_{IN}(t) = 0$, and the output of the integrator is set to zero at $t = 0$, the output will behave in the following fashion:

$$i_{OUT}(t) = \frac{t}{\tau} I_{OFFSET,IN} \tag{5.1}$$

where τ is the time constant of the integrator as defined in Sect. 3.3.2. When the output of the integrator is measured using set-up B, V_{OB} behaves as follows:

$$V_{OB}(t) = \frac{tRA_{F_2}}{\tau} I_{OFFSET,IN} \tag{5.2}$$

where A_{F_2} is the gain of F_2 from input to its middle output. The input offset can be calculated from Eq. 5.2 by differentiating with respect to time and solving for $I_{OFFSET,IN}$, giving:

$$I_{OFFSET,IN} = \frac{dV_{OB}}{dt} \frac{\tau}{RA_{F_2}} \tag{5.3}$$

The offset was measured when the integrator's offset cancellation scheme was used and when it was not. A line was fit to the measured output of the integrator over an interval of time following the connection of the integrator's output to the measurement set-up. The slope of this line was interpreted as an average value for $\frac{dV_{OB}}{dt}$.

Table 5.1 shows results for the input offset of the integrator when the integrator's offset cancellation scheme was used and when it was not used. The three columns of data correspond to measurements of the integrator over its three input/output signal ranges. The numbers reported are the root mean squared average of the offsets of the integrators on one chip.

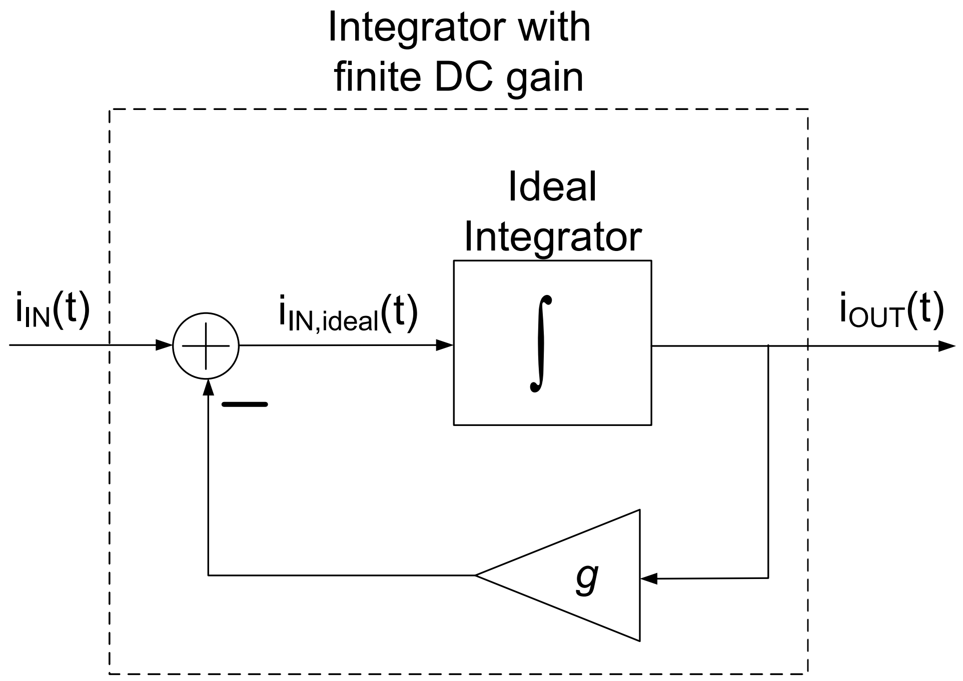

Figure 5.3: Integrator with finite DC gain.

DC Gain

When a step input of height u is applied to the input of an ideal, open loop integrator, the integrator's output will be a ramp ($i_{OUT} = \frac{t}{\tau}u$). However, most real integrators are modeled by the diagram in Fig. 5.3, where the gain g ($g \ll 1$) models loss in the integrator. This system, from input to output has a transfer function given by:

$$H(s) = \frac{1}{g}\left(\frac{1}{\frac{s\tau}{g}+1}\right) \tag{5.4}$$

In steady state this has a gain of $H(0) = \frac{1}{g}$. Frequently $\frac{1}{g}$ is referred to as the DC gain of the integrator. In theory one could measure the DC gain of the integrator by exciting an open loop integrator by a very small step input (small enough that $\frac{u}{g}$ is less than the output limit of the integrator) and measuring the output. However, open loop integrators are difficult to measure, since the input referred offset will also induce an output. Instead a combination of set-up E and F are used. Recall that

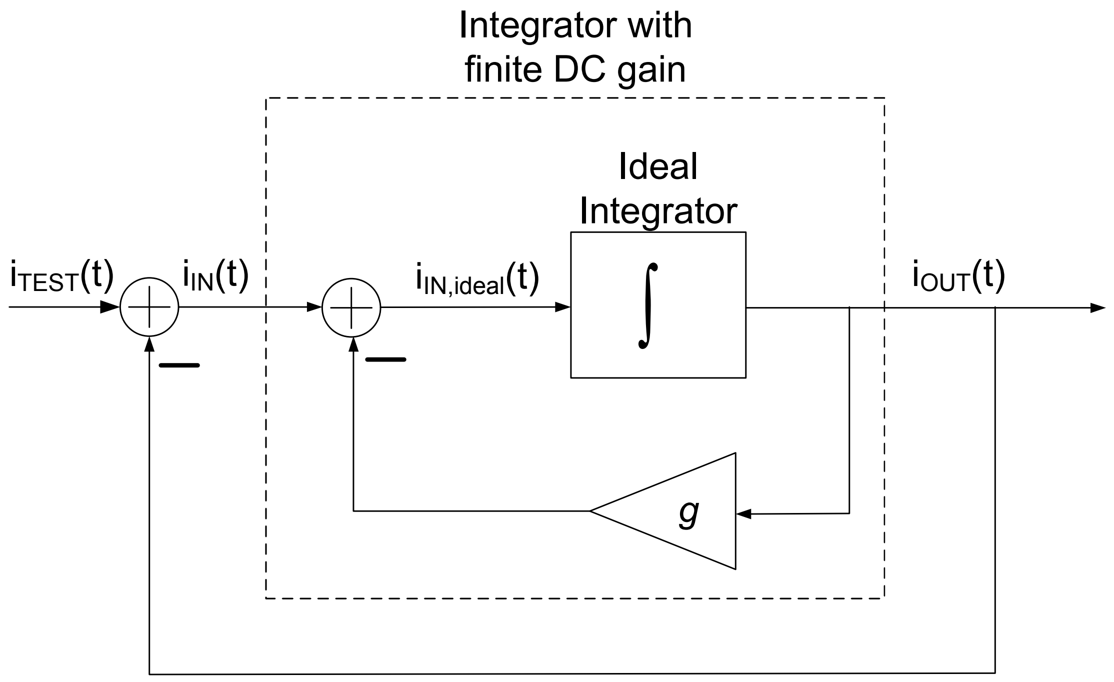

Figure 5.4: Integrator with finite DC gain connected in unity-gain negative feedback.

the DC gain is the steady-state ratio of the output of the integrator to the input of the integrator. This ratio can be measured whether or not the integrator is in open loop. Fig. 5.4 shows a simplified block diagram of set-ups E and F when the integrator is set to be on the high signal range, meaning that F_1 and F_2 have unity gain. The main difference between set-ups E and F is the quantity routed off-chip for measurement. In set-up E, i_{out} is measured whereas in set-up F, i_{in} is measured. The transfer function from i_{TEST} to i_{OUT}, H_{CL-LP} (the quantity measurable in set-up E), is given by:

$$H_{CL-LP}(s) = \frac{1}{s\tau + 1 + g} \tag{5.5}$$

The subscript CL-LP is used because this is the low-pass transfer function of the closed loop system. The DC gain of H_{CL-LP} is $\frac{1}{1+g}$. The transfer function from i_{TEST} to i_{IN}, H_{CL-HP} (the quantity measurable in set-up F), is given by:

$$H_{CL-HP}(s) = \frac{s\tau + g}{s\tau + 1 + g} \tag{5.6}$$

The subscript CL-HP is used because this is the high-pass transfer function of the closed loop system, the DC gain of which is $\frac{g}{1+g}$. If a step of height of u is applied to both of set-ups E and F, the steady-state output of set-up E will be $u\frac{1}{1+g}$ while the output of set-up F will be $u\frac{g}{1+g}$. A ratio of the steady-state outputs is $\frac{1}{g}$, the DC gain of the integrator. In terms of the output voltages of the set-ups this ratio is expressed as:

$$A_{DC} = \frac{V_{OE}(t_f) - V_{OE}(t_0^-)}{V_{OF}(t_f) - V_{OF}(t_0^-)} \tag{5.7}$$

where both set-ups E and F are excited by steps of equal height at t_0. t_0^- denotes a time shortly before t_0. t_f denotes the time at which the output voltages can be assumed to be sufficiently close to their steady-state values.

Some of the integrators were measured to have a negative DC gain. This comes from the loss term g being negative. For active-RC integrators, the DC gain is a large positive number stemming from the finite gain of the opamp. In the case of the integrator used in the AC, the finite DC gain stems from non-zero conductances that are in parallel with the integration capacitors, and mismatch within the pairs of current mirror devices M19/M20 and M20/M21 (Fig. 3.5). In the event that the mirror's gain is greater than unity, the integrator is modeled with $g < 0$. To maintain a steady-state positive output in set-up F, a small negative input current must be applied to the input of the integrator, meaning that the numerator of Eq. 5.7 is negative. This leads to the calculation of a negative DC gain. This, however, does not mean that the output of an open loop integrator, when the input is excited by a very small step input, will reach a steady state of some large negative output, as the term "negative DC gain" would imply. In openloop, there is no negative feedback to offset the small amount positive feedback, and the system is unstable. When a small step is applied the integrator's output grows exponentially in the positive direction until it saturates.

Table 5.1 reports the smallest magnitude of DC gain ($min(|A_{DC}|)$) for each signal range on the chip that was tested. In addition, it reports a type of average in the row labeled $\mu(|A_{DC}|)$. This is intended to represent the average DC gain. However, it is inappropriate to take a simple arithmetic mean of the DC gains of all of the integrators. In the event that the current mirrors (M19/M20 and M21/M22) of one integrator had ratios such that they perfectly canceled the losses of the integrator, the integrator would have infinite DC gain, and the average of this integrator with all of the others would also be infinite. DC gain measurements serve to characterize the size of the loss term g. We express it as the reciprocal of g, namely DC gain, because DC gain is more intuitive to circuit designers. Ultimately we are interested in a measure of the average size of the absolute value of this loss term, but expressed as a DC Gain. As such, it was decided that a better way to combine DC gains is to take an average of the reciprocals of the DC gains, and invert this average. This result is recorded in the table.

Noise

The mean squared value of V_{OA} was subtracted from the mean squared value of V_{OC} to compute the mean squared output noise voltage. This was converted to a mean squared current by dividing it by R^2 to get the mean squared output noise current. The noise of the integrator was measured in a feedback configuration, in part, because it is difficult to take open loop measurements. Also, in many simulations the integrator will have feedback around it. As stated in Ch. 3 the output noise of an integrator is highly configuration-dependent. In this configuration some of the integrator's low frequency output noise is canceled by the feedback. In feedback, the input referred noise of the integrator is processed by the transfer function $H_{CL}(s) = \frac{1}{\tau s+1}$ whereas in open loop, the input noise is processed by the transfer function of the integrator alone, $H_{OL}(s) = \frac{1}{\tau s}$. The latter transfer function has more gain at frequencies below $f = \frac{1}{2\pi\tau}$.

The output noise in set-up C includes noise from F_2, while the noise in set-up A does not. For some initial measurements the output noise of set-up A in Fig. 5.6 was subtracted from the noise in set-up C. In some instances, the calculated noise was negative, meaning that the addition of the integrator and feedback path reduced the overall noise of the system. This is possible since the input noise of F_2 is reduced by the loop gain of the system, which for low frequencies is high. Also, the connection of the output of the IUT to the input of F_2 loads the input of F_2 by the capacitance of long connection wires, which filter the input referred noise of F_2 at higher frequencies. This filtering is more pronounced when the input range of F_2 is on the smallest range, which is the scenario that gave negative noise.

In response to the measurement of negative noise, it was decided that the total noise of the integrator and F_2 would be reported, but referred to the output of the integrator by scaling the mean squared current by $\frac{1}{A_{F_2}^2}$.

Table 5.1 reports the integrators' output referred RMS noise averaged across the integrators on the chip, averaged in an RMS fashion.

Nonlinearity

For any linear system, if the input u results in the output y, then the input $2u$ results in the output $2y$. If the output of a nonlinear system when $2u$ is the input is $\hat{y}$, the nonlinearity of the system will be based on the difference between $\hat{y}$ and $2y$. This statement is vague because many different calculations are possible. In some initial measurements of the integrator, a staircase function (Fig. 5.7) was applied to set-up E, with constant input levels lasting long enough that the integrator would reach steady state. The output voltage for the sections of the treads (level sections) during which the output can be considered to be equal to the steady-state value were averaged to produce a series of points. A line was fit to these average values and the nonlinearity of the system was inferred by the RMS deviation of the averaged points from the straight line. However, the integrator itself could be woefully nonlinear but the loop gain of the system would mask it.

Instead, the nonlinearity of the system must be examined during a transient. To do this, the output of set-up E was measured for steps of two different heights, one twice the other. If the system were perfectly linear, the step response for the larger step would be exactly twice that of the smaller step. In other words, if a step of u led to the output y_1 and a step of out $2u$ led to y_2, the nonlinearity of the system is:

$$\delta y = y_2 - 2y_1 \tag{5.8}$$

over some interval of time. For this measurement, the two steps were compared over the interval of time during which the output of the system was between 10 % and 90 % of its final value. In terms of the quantities in Fig. 5.1 the nonlinearity of the system, $i_{err}(t)$ is given by:

$$i_{err}(t) = \frac{V_{OE}(t)|_L - 2V_{OE}(t)|_S}{R} \tag{5.9}$$

The subscript L denotes the output for the larger step input, while S denotes the output for the half-sized step input. The mean squared nonlinearity was calculated by the following:

$$\overline{i_{NL}^2} = \frac{1}{A_{F_2}^2}(\overline{i_{err}^2(t)} - \frac{\overline{V_{OE}^2(t_f - \Delta t, t_f)|_L} + 4\overline{V_{OE}^2(t_f - \Delta t, t_f)|_S}}{R^2}) \tag{5.10}$$

Scaling by $\frac{1}{A_{F_2}^2}$ converts the measured mean squared output current from the set-up E to a mean squared current at output of the IUT. Eq. 5.10 is more complicated than simply scaling i_{err}^2 because i_{err} contains the effects of noise, in addition to nonlinearity. Even if the system were perfectly linear, $i_{err} \neq 0$ because the two measurements at the output of set-up E would contain noise and hence a perfectly linear system would appear nonlinear if the mean-squared value of i_{err} were reported. To account for this, typical values for the block's output noise is subtracted. The second term in Eq. 5.10 (fraction over R^2) represents the total noise of the outputs from each of the two measurements over an interval of time during which the output has reached steady-state. This measurement appears in the table as "RMS Nonlinearity". This row of the table represents the results for each integrator combined in an RMS fashion.

Time constant (τ)

$\mu(\tau)$ is the mean of τ for all integrators on the chip, when the DAC is set to 512. $\sigma(\tau)$ is the standard deviation of the measured τ across the integrators on the chip. The integrator-to-integrator matching of τ is not important since each integrator's behaviour is measured, and individually programmed when a simulation is run. The time constant of the step response of set-up D was subtracted from the time constant of the step response of set-up E to compute the reported time constant.

5.1.2 Results

Measured results for a chip are summarized in the Table 5.1. The three numeric columns report results for three different signal ranges.

DAC Characteristic

Fig. 5.5 shows the τ Vs. DAC tuning word for a typical integrator. The non-monotonicity of the DAC is clearly visible in the discontinuities of the curve.

Signal Range:	20 μA	1 μA	100 nA
$I_{OFFSET,IN}$, with cancellation (nA)	7.3	0.21	0.017
$I_{OFFSET,IN}$, without cancellation (nA)	150	7.4	1.8
$min(\|A_{DC}\|)$	132	107	102
$\mu(\|A_{DC}\|)$	476	425	422
RMS Noise, nA, nA, nA	14.3	0.76	0.072
RMS NonLinearity (nA)	10.8	2.87	0.62
$\mu(\tau)$, μs	28.3	28.0	26.4
$\sigma(\tau)$, μs	0.61	0.64	0.57

Table 5.1: Measured results for the integrator block.

5.2 VGA/2-Input Multipliers Blocks Measured in VGA Mode

5.2.1 Measurement Set-Up and Definitions of Reported Quantities

The various configurations used to test the VGAs are shown in Fig. 5.6. The amplifier under test is denoted by the block labeled “AUT”. The input range of F_2 and the output range of F_1 are set to be the same range as the output range of the AUT and the input range of the AUT, respectively. The input of F_1 and the output of F_2 are on the high range.

For many of the tests, a stair-case input function was applied to the chain (Fig. 5.7). Each tread of the function was long enough that the circuit was considered to be in steady-state for the majority of the tread. The output of the circuit was

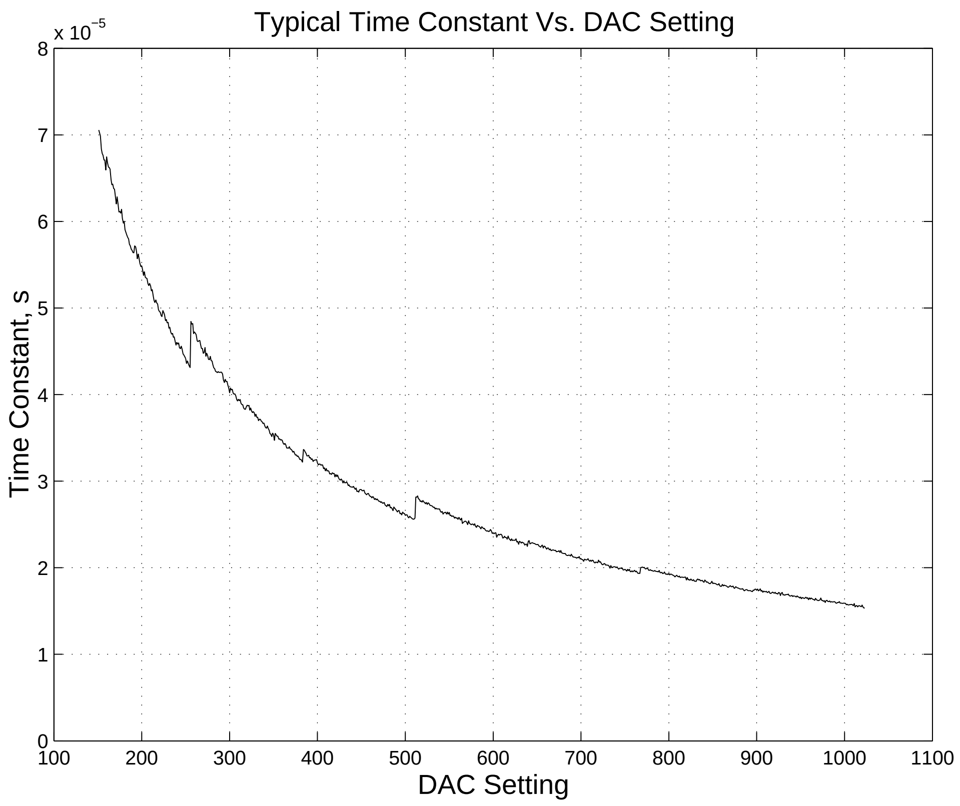

Figure 5.5: Typical τ Vs. DAC tuning word.

averaged over most of each tread to compute a set of average values, shown with * signs. A best fit line was fit to this set of average values using a least-squares fit, shown by the upward sloping straightline.

Results for the VGA are summarized in the Tables 5.2, 5.3, and 5.4. The column labeled "Range" refers to the input and output range settings of the block. The first letter denotes the input range and the second letter denotes the output range. "h" refers to the largest signal range (20 μA), "m" to the middle range (1 μA) and "l" to the smallest signal range (100 nA). The column "DAC" refers to the setting of the 10-bit DAC that controls the gain of the block. The other columns are

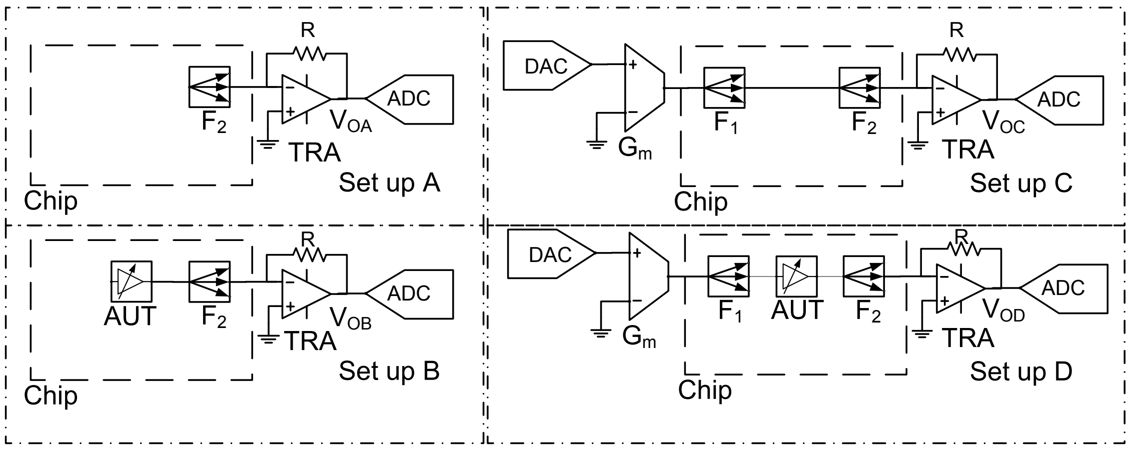

Figure 5.6: Test set-up for measuring VGA blocks.

explained below.

Gain

The gain measurement is done by computing the slope of the input-output transfer characteristic (using the staircase function) of set-up D (Fig. 5.6) and comparing this slope to that of set-up C. $\mu(K)$ is the arithmetic mean of the gain for all VGAs on the chip for the particular combination of range and DAC settings. $\frac{\sigma(K)}{\mu(K)}$ is the standard deviation of the gain normalized to the average gain over all of the VGAs on the chip.

Nonlinearity

The nonlinearity of the amplifier reported in the tables is the RMS deviation of the actual input-output transfer characteristic of set-up D (average values of the treads, computed using the staircase input) from a linear fit of the characteristic. NL FS (denoting nonlinearity, full-scale) and NL HS (denoting nonlinearity, half-scale) are

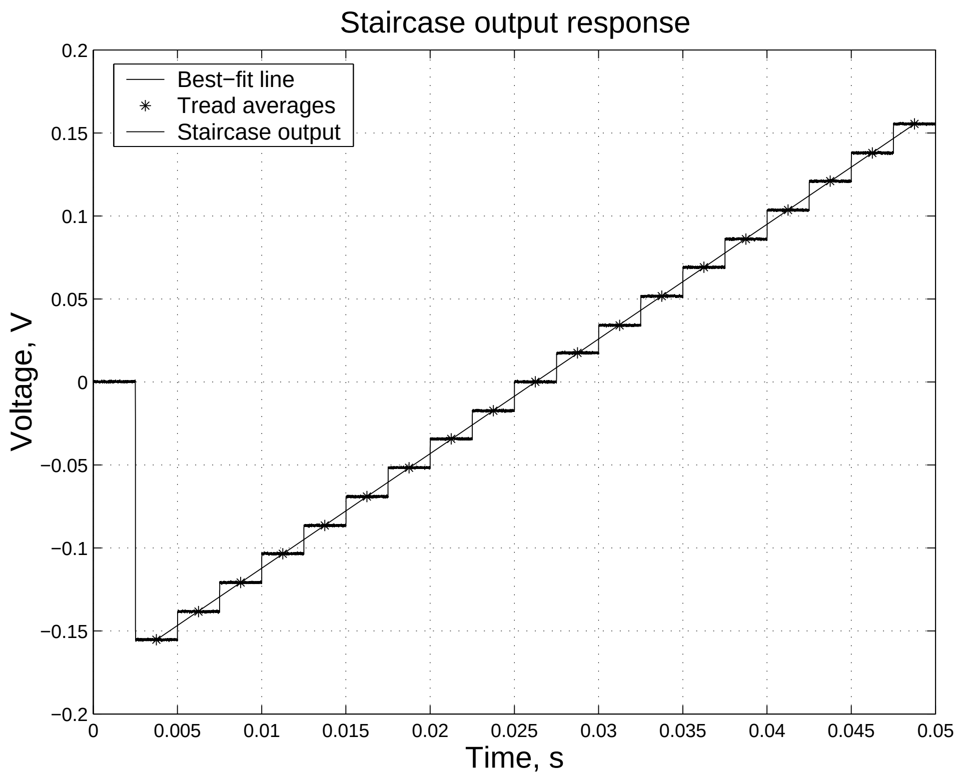

Figure 5.7: Technique for determining the slope of input-output characteristics.

the RMS deviation of the measured characteristic of the amplifier over 80 % and 40 % of its input range, respectively, from a linear fit. The reported numbers are normalized to a full scale input.

Offset

The output offset of the amplifier is the difference between the average value of V_{OB} and V_{OA}, divided by the product of R and the gain of F_2. $\mu(O_{OS})$ is the arithmetic mean of the input offsets for all gains on the chip. This indicates a deterministic shift in the input-output characteristic of the block. $\sqrt{\mu(O_{OS}^2)}$ is the root-mean squared

Range (in/out)	DAC Input	$\mu(K)$ (A/A)	$\frac{\sigma(K)}{\mu(K)}$ (%)	$\mu(O_{OS})$ (nA)	$\sqrt{\mu(O_{OS}^2)}$ (nA)	NL FS (%)	NL HS (%)	Noise (nA)
h/h	127	1.00	2.3	-94.7	219	0.020	0.019	14.9
h/h	255	1.74	1.9	-97.2	209	0.019	0.018	14.0
h/h	511	2.78	2.5	-104	210	0.19	0.032	13.7
h/m	127	0.053	2.5	-36.3	222	0.022	0.019	13.0
h/m	255	0.092	2.1	-32.2	209	0.020	0.017	10.8
h/m	511	0.155	1.8	-31.9	203	0.020	0.017	9.29

Table 5.2: Measured results for the VGA block. Largest input range.

Range (in/out)	DAC Input	$\mu(K)$ (A/A)	$\frac{\sigma(K)}{\mu(K)}$ (%)	$\mu(O_{OS})$ (nA)	$\sqrt{\mu(O_{OS}^2)}$ (nA)	NL FS (%)	NL HS (%)	Noise (nA)
m/h	127	19.7	2.31	-7.6	13	0.049	0.053	0.92
m/h	255	34.3	1.96	-7.9	12	0.049	0.053	0.90
m/h	511	55.8	2.53	-8.2	12	0.16	0.061	0.92
m/m	127	1.01	2.39	-3.1	12	0.050	0.052	0.79
m/m	255	1.76	2.04	-3.5	11	0.049	0.050	0.69
m/m	511	2.99	1.83	-3.9	11	0.049	0.049	0.63
m/l	127	0.104	2.78	-3.5	14	0.049	0.050	0.42
m/l	255	0.181	2.30	-3.9	13	0.048	0.048	0.35
m/l	511	0.307	1.91	-4.2	12	0.050	0.048	0.30

Table 5.3: Measured results for the VGA Block. Middle input range.

input offset for all gains on the chip.

Noise

The output noise is computed by subtracting the mean squared value of V_{OA} from the mean squared value of V_{OB}. The voltage noise is converted to an input referred current noise by dividing it by R^2, $A_{F_2}^2$, and A_{AUT}^2. The Noise column in the table is the input referred RMS noise of the VGAs.

Range (in/out)	DAC Input	$\mu(K)$ (A/A)	$\frac{\sigma(K)}{\mu(K)}$ (%)	$\mu(O_{OS})$ (nA)	$\sqrt{\mu(O_{OS}^2)}$ (nA)	NL FS (%)	NL HS (%)	Noise (nA)
l/h	127	190	3.8	-1.9	2.5	0.053	0.053	0.12
l/h	255	335	2.9	-2.0	2.5	0.053	0.052	0.15
l/h	511	542	3.5	-2.0	2.6	0.22	0.072	0.18
l/m	127	9.69	3.9	-1.6	2.3	0.053	0.055	0.10
l/m	255	17.1	2.9	-1.6	2.3	0.052	0.054	0.098
l/m	511	29.3	2.3	-1.7	2.3	0.054	0.053	0.094
l/l	127	0.987	4.1	-1.5	2.2	0.053	0.054	0.057
l/l	255	1.74	3.1	-1.6	2.3	0.052	0.053	0.051
l/l	511	2.99	2.4	-1.7	2.3	0.054	0.053	0.047

Table 5.4: Measured results for the VGA block. Smallest input range.

5.2.2 Results

Measured results for the VGA are found in Tables 5.2, 5.3, and 5.4.

5.3 VGA/2-Input Multiplier Blocks Measured in Multiplier Mode

5.3.1 Measurement Set-up and Definitions of Reported Quantities

Fig. 5.8 shows various measurement set-ups for characterizing the multiplier blocks. The multiplier under test is denoted by the block labeled "MUT". An ideal multiplier

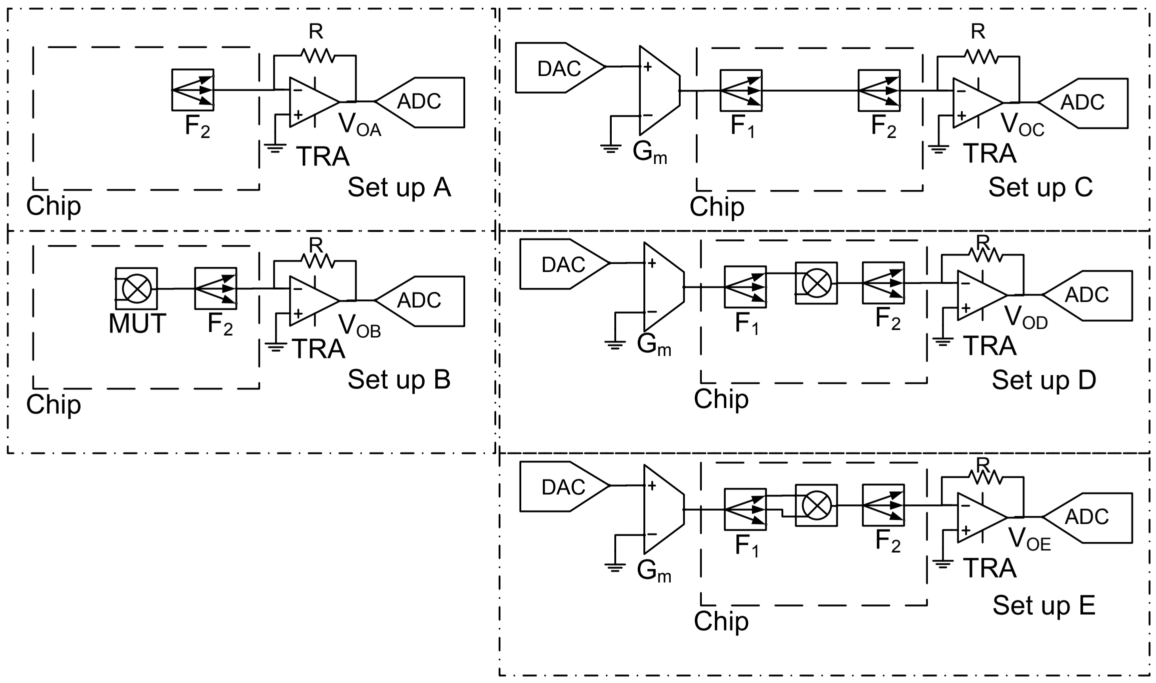

Figure 5.8: Test set-up for measuring Multiplier blocks.

implements the following input-output behaviour:

$$i_o^+ - i_o^- = K(i_1^+ - i_1^-)(i_2^+ - i_2^-) \tag{5.11}$$

where K has units of A^{-1}, i_1 and i_2 are differential input currents and i_o is a differential output current. Due to mismatch between devices and noise, a real multiplier may implement the following:

$$i_o^+ - i_o^- = K(i_1^+ - i_1^- + i_{off1})(i_2^+ - i_2^- + i_{off2}) + i_{offO} + i_n(t) \tag{5.12}$$

where i_{off1}, i_{off2} and i_{offO} are offset currents at the input of port 1, the input of port 2 and the output, respectively. The term $i_n(t)$ is an output referred noise current. Eq. 5.12 is a simplified model of a real multiplier, which may have signal depended noise and may implement a higher order polynomial function. The following sections discuss the measurement of the multiplication coefficient (K), the offset currents and the output referred noise.

The reported measurements for the multiplier block, in Table 5.5, are for a DAC setting of 300.

Multiplication Constant

The measurement of the multiplier is limited by the fact that the data acquisition system has only one free output. To determine K in Eq. 5.12 the same input signal was applied to both inputs and the multipliers behaviour as a squarer was measured. Two outputs of F_1 (in set-up E) were each connected to the two inputs of the multiplier with both positive and negative polarity for a total of four input combinations. That is, each connection has two possibilities for its polarity meaning that there are four ways to make the two connections. For each combination the system was excited by the staircase function. A second-order polynomial was fit to the level sections' average values, leading to four different polynomials. The four second-order terms were averaged to give one value for K for the multiplier for a particular combination of input range, output range and DAC setting.

In Table 5.5 $\mu(K)$ is the arithmetic mean of the multiplication constants across the chip. $\frac{\sigma(K)}{\mu(K)}$ is the standard deviation of the multiplication normalized to its mean for the multipliers across one chip.

Offsets

As Eq. 5.12 shows, the inputs and output each have offsets. The output offset of the multiplier is the difference between the DC output voltage in set-up B and that in set-up A, divided by R and the gain of F_2. The input offset of the lower input is computed by comparing the slope of the input-output transfer characteristic in set-up D to that in set-up C. To determine the offset of the upper input, a similar procedure is used, but with F_1 connected to the lower input. In Table 5.5, $\mu(O_{OS})$, $\mu(I_{OS1})$, and $\mu(I_{OS2})$ are the arithmetic means for the given offsets. $\sqrt{\mu(O_{OS}^2)}$, $\sqrt{\mu(I_{OS1}^2)}$, and $\sqrt{\mu(I_{OS2}^2)}$ are the root mean squared offsets for each port.

Noise

The output mean-squared current noise is the difference between the mean squared values of V_{OB} and V_{OA}, scaled by R^2 and by $A_{F_2}^2$. Table 5.5 reports the root mean-squared current noise.

5.3.2 Results

Measurement results for one chip are summarized in Table 5.5.

Range (In & Out)	$\mu(K)$ (MA^{-1})	$\frac{\sigma(K)}{\mu(K)}$ (%)	$\mu(O_{OS})$ (nA)	$\sqrt{\mu(O_{OS}^2)}$ (nA)	$\mu(I_{OS1})$ (nA)	$\sqrt{\mu(I_{OS1}^2)}$ (nA)	$\mu(I_{OS2})$ (nA)	$\sqrt{\mu(I_{OS2}^2)}$ (nA)	Noise (nA)
h	0.08	2.07	-26.0	225	89.3	183	52.1	168	19.6
m	1.70	2.65	-9.14	16.5	6.98	10.7	5.85	10.3	1.02
l	17.0	2.54	-0.87	2.23	1.58	2.09	1.62	2.29	0.08

Table 5.5: Measured results for the Multiplier block.

5.4 Fanout

5.4.1 Measurement Set-up and Definitions of Reported Quantities

Various test configurations are shown in Fig. 5.9. The fanout under test (FUT) was preceded and followed by fanout blocks (Fig. 5.9) whose ranges were selected in the following way: F_1's input was set on its high range. Its output range was set to be the same as the input range of the FUT. F_2's input range was set to be equal to the output range of the FUT, while F_2's output range was set to be high. Table 5.6 summarizes the measurements of the Fanout blocks. The first column of Table 5.6 gives the input and output ranges of the block. "h" denotes the largest signal range (9 or 18 μA), "m" denotes the middle signal range (1 or 2 μA) and "l" denotes the smallest signal range (111 or 222 nA).

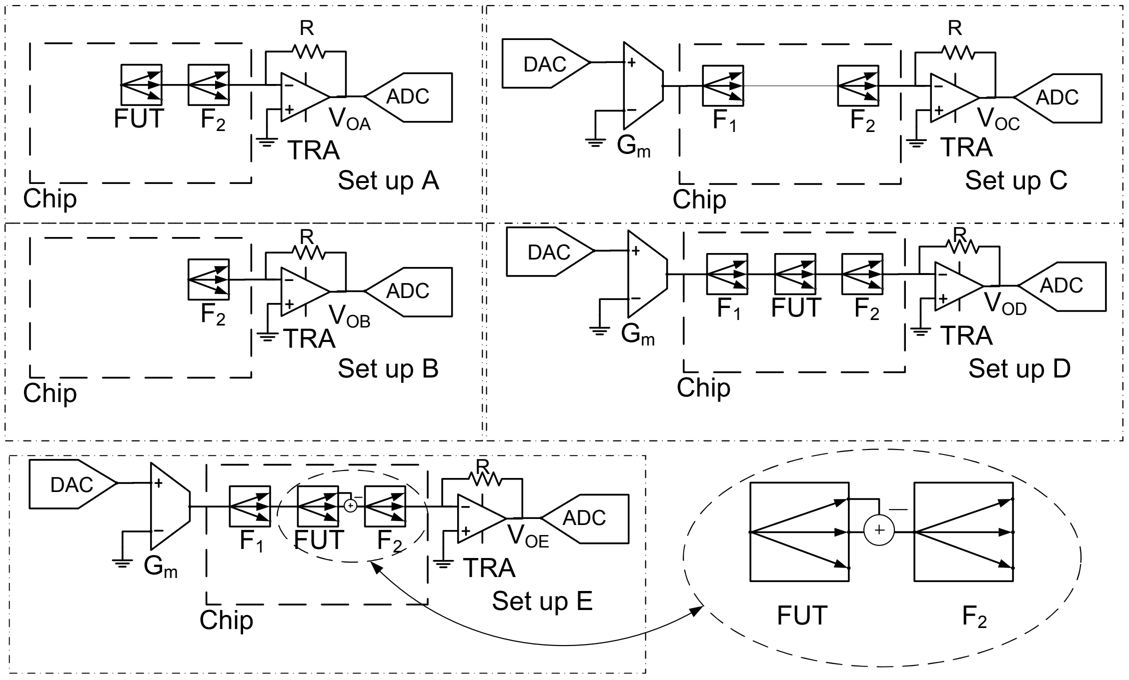

Figure 5.9: Test set-up for measuring Fanout blocks.

Output Offsets

The offsets reported are the output offsets for each output of each fanout block, computed by subtracting the average value of V_{OB} from V_{OA}. This DC voltage was converted to a DC current by dividing it by R, and dividing it by the gain of F_2. The numbers in the left “Op. Offset” column are the RMS output offsets over all fanout outputs across the chip. The right “Op. Offset” column shows the output offsets normalized to the output signal range.

Noise

The mean squared value of V_{OB} was subtracted from the mean squared value of V_{OA}. To compute the RMS output noise current of the fanout, the square root of the difference was divided by the gain of F_2 and by R. The reported numbers are for the output referred RMS noise current.

Gain

For this, and other specifications, the stair case function was applied to set-up D and to set-up C. The gain of the fanout is the ratio of the slope of the best fit line for set-up D to the slope of the line measured using set-up C. The numbers in the "Gain" column are the averages over the three paths for each fanout and over the chip. "RMS Dev" refers to the standard deviation of the gains, normalized to the average gain.

Nonlinearity

The nonlinearity numbers reported are the RMS difference between the averages of the treads and the line of best fit for an input that is 80 % of the fullscale range of the FUT. RMS NL refers to the RMS nonlinearity referred to the input of the block. The reported numbers in the left column are the RMS across all fanout outputs on the chip, and the right RMS NL column has the results normalized to the input range.

Mismatch

This specification is a measure of the difference between the gain from the input of the fanout to one of its outputs and the gain to another of its outputs. It is measured using set-up E. Two outputs of a fanout block are subtracted from one another at the input of F_2. A stair-case input is applied to this arrangement, and matching is

Range (in / out) name, (uA) / name, (uA)	Gain (A/A)	RMS Dev (%)	Op. Offset (nA)	Op. Offset (%)	RMS NL (nA)	RMS NL (x 1e-6)	Mismatch (%)	Noise (nA)
h, 18 / h, 18	1.00	0.18	132.73	0.74	4.76	264.34	0.22	1.07
m, 1 / m, 1	1.00	0.22	11.36	1.14	0.54	544.45	0.25	0.20
l, 0.111 / l, 0.111	1.00	0.23	1.26	1.14	0.06	537.64	0.26	0.03
h, 18 / m, 2	0.11	0.83	15.67	0.78	4.87	270.38	0.20	0.16
h, 18 / l, 0.222	0.01	0.85	11.93	5.37	11.33	629.67	0.24	0.00
m, 1 / h, 9	9.00	0.22	101.18	1.12	0.55	547.58	0.26	2.02
m, 1 / l, 0.111	0.11	0.61	1.64	1.48	0.52	523.92	0.26	0.01
l, 0.111 / h, 9	80.92	0.39	129.05	1.43	0.06	553.96	0.26	3.26
l, 0.111 / m, 1	9.01	0.24	13.32	1.33	0.06	557.97	0.24	0.35

Table 5.6: Measured results for the Fanout block.

the ratio of the slope the output's best fit line to the slope of a similar line measured with the same input at the output of set-up D.

5.4.2 Results

The results are summarized in Table 5.6.

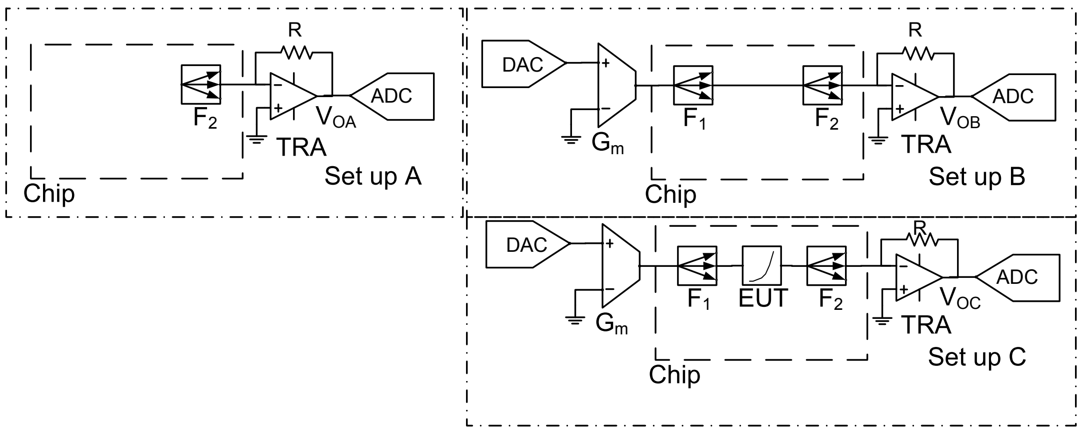

Figure 5.10: Test set-up for measuring Exponential blocks.

5.5 Exponential

Various configurations used to measure the exponential blocks are shown in Fig. 5.10. A combination of set-ups A and B was used to compute the input-referred offset of the chain of blocks consisting of the DAC, the transconductor (G_m) and F_1 allowing the offset to be cancelled. Set-up C was used to measure the input-output transfer characteristic of the block. The purpose of measuring the exponential blocks is to determine the degree to which they exhibit exponential input-output behaviour. An ideal exponential block implements the following input-to-output characteristic:

$$i_o^+ - i_o^- = 2I_1 \exp\left(\frac{i_{in}^+ - i_{in}^-}{I_{REF}}\right) \tag{5.13}$$

The equation above is a modified version of Eq. 3.41. $\frac{n\phi_t}{R}$ has been replaced by I_{REF}. The block's transfer characteristic becomes the following when it has an input and output offset current:

$$i_o^+ - i_o^- = 2I_1 \exp\left(\frac{i_{in}^+ - i_{in}^- + I_{IN}}{I_{REF}}\right) + I_O \tag{5.14}$$

where I_{IN} and I_O are the block's input- and output-offset currents, respectively. Eq. 5.14 can be written as the following:

$$i_o^+ - i_o^- = 2I_1 \exp\left(\frac{I_{IN}}{I_{REF}}\right) \exp\left(\frac{i_{in}^+ - i_{in}^-}{I_{REF}}\right) + I_O \tag{5.15}$$

From Eq. 5.15 it is clear that the input-offset current modifies the transfer characteristic only as an output, multiplicative scale factor $\exp \frac{I_{IN}}{I_{REF}}$ while the output-offset current deviates the transfer characteristic from exponential. In the measurements of the blocks, an attempt was made to determine the size of the output offset current so that it could be subtracted from measured results. This was done by measuring the output in set-up C when a large negative input was applied, effectively eliminating the exponential term's contribution to the output, leaving only an output due to the output-offset current of the block. This offset was subtracted from the measured output when smaller inputs were applied.

A typical exponential block's input-to-output transfer characteristic is shown in Fig. 5.11. The vertical axis has a logarithmic scaling. Deviation from exponential was computed in the following fashion, assuming the measured data, with output offset subtracted is a vector y:

- The base-10 logarithm of the output current was computed for each data point. $y_{log} = \log_{10}(y)$
- A line was fit to these data using a least-squares technique over the range of inputs from -4.3 μA to 6.0 μA. The points on the line will be denoted as y_{fit}
- The deviation of the logarithm of the measured data from the fit line was

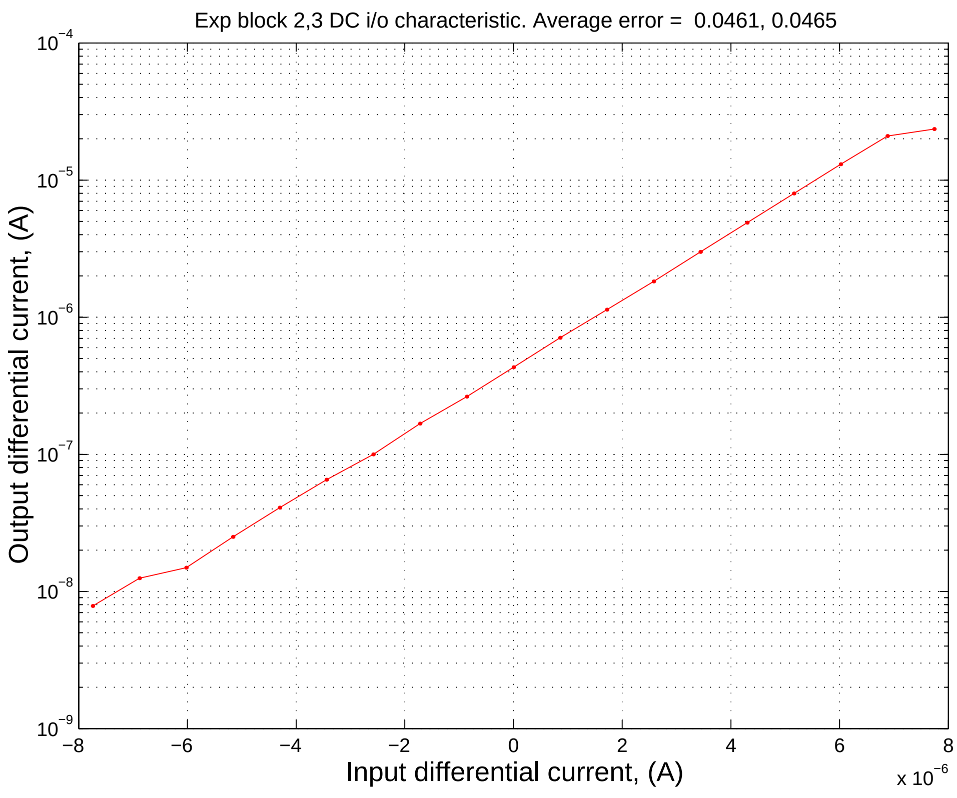

Figure 5.11: A typical exponential block's input-to-output transfer characteristic.

computed ($y_{ratio} = y_{log} - y_{fit}$). This corresponds to finding the logarithm of the ratio of measured data to the fit line, since a difference in logarithm corresponds to the logarithm of a ratio.

- The ratio was converted to a difference. y_{ratio} is the logarithm of a ratio. Hence, $10^{y_{ratio}}$ is the ratio, in linear units, between the measured output and the fit line. Ideally, this ratio would be equal to one. We are interested in quantifying its difference from one. Therefore, we consider the error in the input-output characteristic of the block to be: $y_{diff} = \text{abs}(10^{y_{ratio}} - 1)$.

- The reported error for the exponential block is the RMS average of y_{diff}

Fig. 5.12 shows the input-to-output transfer characteristics of all of the exponential blocks from one chip.

Measured Deviation: The measured RMS deviation from exponential for all exponential blocks from one chip was 2.6 %.

5.6 Logarithm

Various configurations used to measure the logarithm blocks are shown in Fig. 5.13. A combination of set-ups A and B were used to compute the input-referred offset of the chain of blocks consisting of the DAC, the transconductor (G_m) and F_1 allowing it to be cancelled. The input-output transfer characteristic was measured using set-up C. The purpose of measuring the logarithm blocks is to determine the degree to which they exhibit logarithmic input-output behaviour. An ideal logarithm block

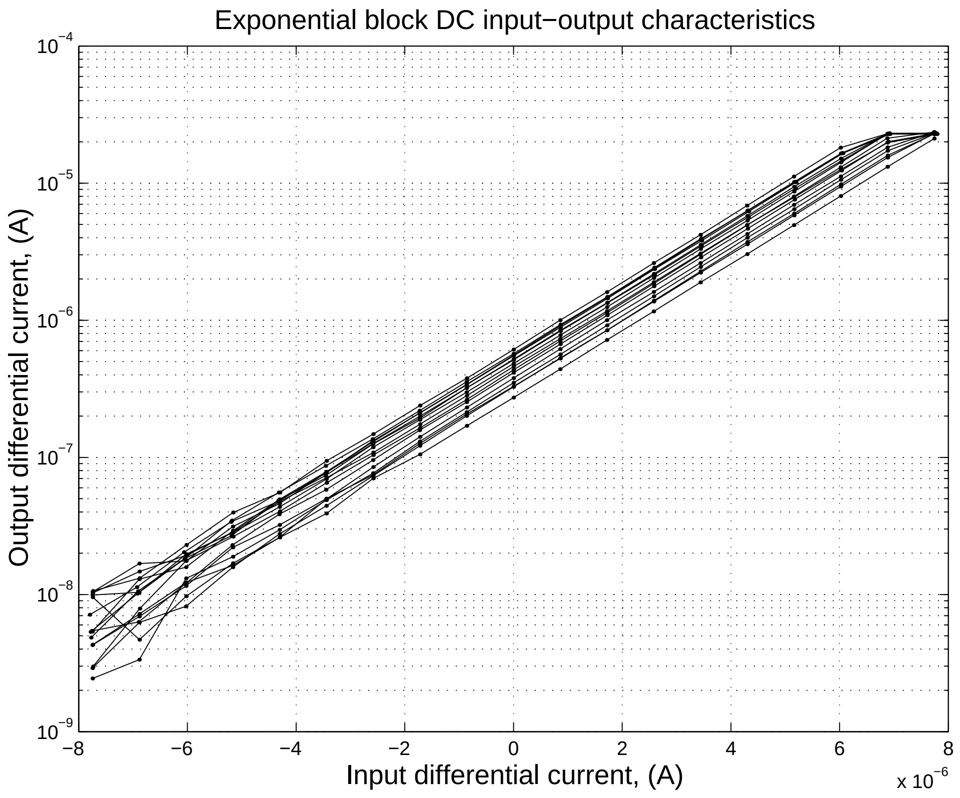

Figure 5.12: Exponential blocks' input-to-output transfer characteristic from one chip.

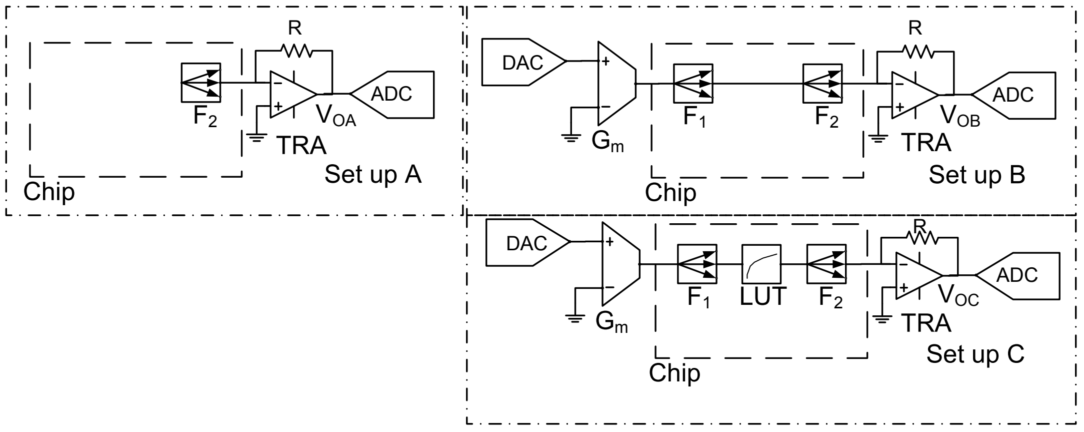

Figure 5.13: Test set-up for measuring Logarithm blocks.

implements the following input-to-output characteristic:

$$i_o^+ - i_o^- = K \log\left(\frac{i_{in}^+ - i_{in}^-}{I_{REF}}\right) \tag{5.16}$$

The equation above is modified version of Eq. 3.42. $G_m n\phi_t$ has been replaced by K. The block's transfer characteristic becomes the following when it has an input and output offset current:

$$i_o^+ - i_o^- = K \log\left(\frac{i_{in}^+ - i_{in}^- + I_{IN}}{I_{REF}}\right) + I_O \tag{5.17}$$

where I_{IN} and I_O are the block's input and output offset currents, respectively. Eq. 5.17 can be written as the following:

$$i_o^+ - i_o^- = K \log\left(10^{\frac{I_O}{K}} \frac{i_{in}^+ - i_{in}^- + I_{IN}}{I_{REF}}\right) \tag{5.18}$$

From Eq. 5.18 it is clear that the output-offset current modifies the transfer characteristic only as an input, multiplicative scale factor $10^{\frac{I_O}{K}}$ while the input-offset current deviates the characteristic from logarithmic. In the measurements of the blocks, an attempt was made to determine the size of the input offset current so that it could be subtracted from the applied input to the circuit during actual measurements. When $i_{in}^+ - i_{in}^- < I_{IN}$ the output of the logarithmic block saturates to its maximum negative output. I_{IN} was estimated by finding the largest input for which the output was saturated to this large negative output. This was done by gradually increasing $i_{in}^+ - i_{in}^-$ from a negative value (larger than the expect I_{IN}) until the output was not saturated to the block's negative maximum. The value causing the output to increase from the block's saturated output was taken to correspond to the input-offset current of the block.

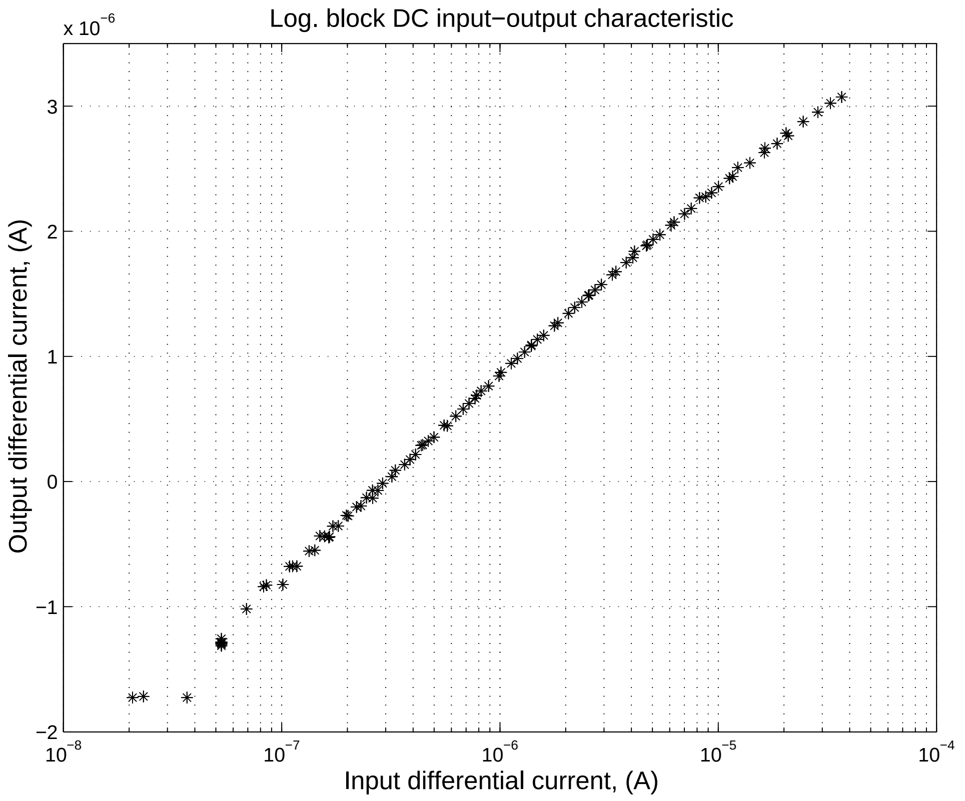

Figure 5.14: A typical logarithm block's input-to-output transfer characteristic.

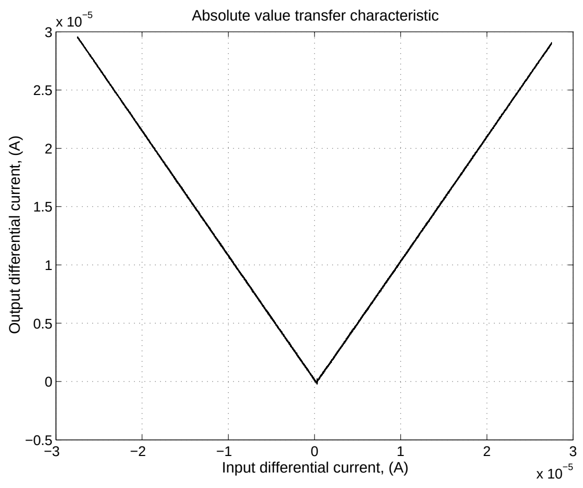

Figure 5.15: A typical programmable nonlinear block's input-to-output transfer characteristic when implementing the absolute value function.

A typical logarithm block's input-to-output transfer characteristic is shown in Fig. 5.14. The horizontal axis has a logarithmic scaling.

5.7 Programmable Nonlinear Blocks

This section shows some representative measurements of the programmable nonlinear blocks. An attempt was made to eliminate the output offset of the circuits connected to the input of the nonlinear block.

Fig. 5.15 shows the input-output transfer characteristic of a representative programmable nonlinear block when it is implementing the absolute value function. Fig. 5.16 shows the input-output transfer characteristic of a representative programmable nonlinear block when it is implementing the saturation function. The output current to which this blocks saturates is programmable through a 10-bit DAC. This

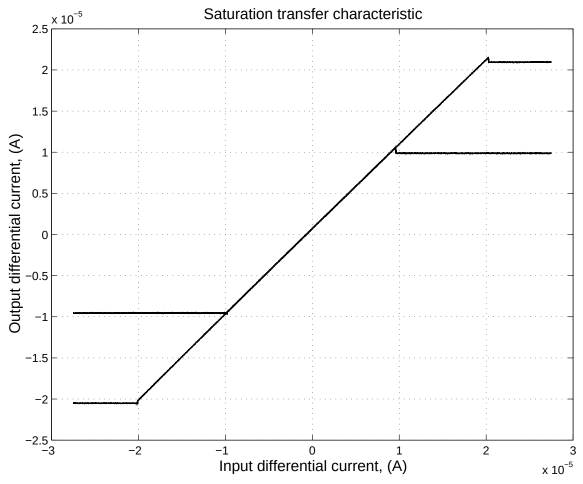

Figure 5.16: A typical programmable nonlinear block's input-to-output transfer characteristic when implementing the saturation function. Two different saturation levels are shown.

figure shows results for two different saturation levels. Fig. 5.17 shows the input-output transfer characteristic of a representative programmable nonlinear block when it is implementing the sign function. The output current to which this blocks saturates is programmable through a 10-bit DAC. This figure shows results for two different saturation levels. Fig. 5.18 shows the input-output transfer characteristic of a representative programmable nonlinear block when it is implementing the ramp function. The definition of this function is found in Sect. 3.3.7. The point on the x-axis at which block's characteristic begins increasing is programmable through a 10-bit DAC. This figure shows results for five different break points.

Figs. 5.19 and 5.20 show the time-domain characteristic of a representative programmable nonlinear block when it is implementing minimum and maximum functions. Fig. 5.19 shows the block's inputs (the ramp input is applied to input-port 1

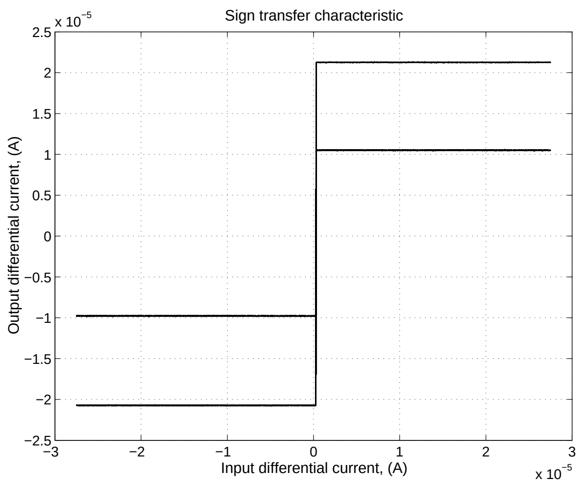

Figure 5.17: A typical programmable nonlinear block's input-to-output transfer characteristic when implementing the sign function. Two different output levels are shown.

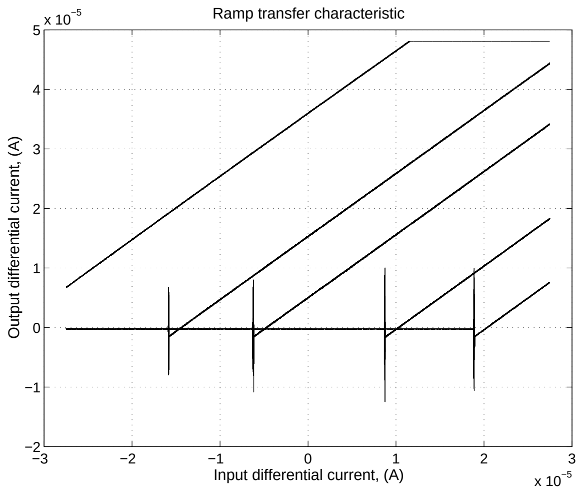

Figure 5.18: A typical programmable nonlinear block's input-to-output transfer characteristic when implementing the ramp function. Different break points shown.

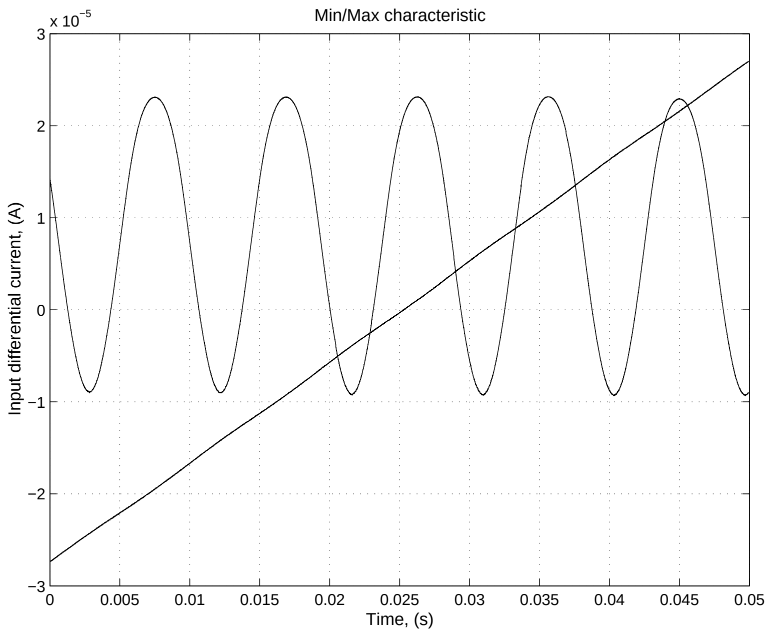

Figure 5.19: Inputs to a programmable nonlinear block.

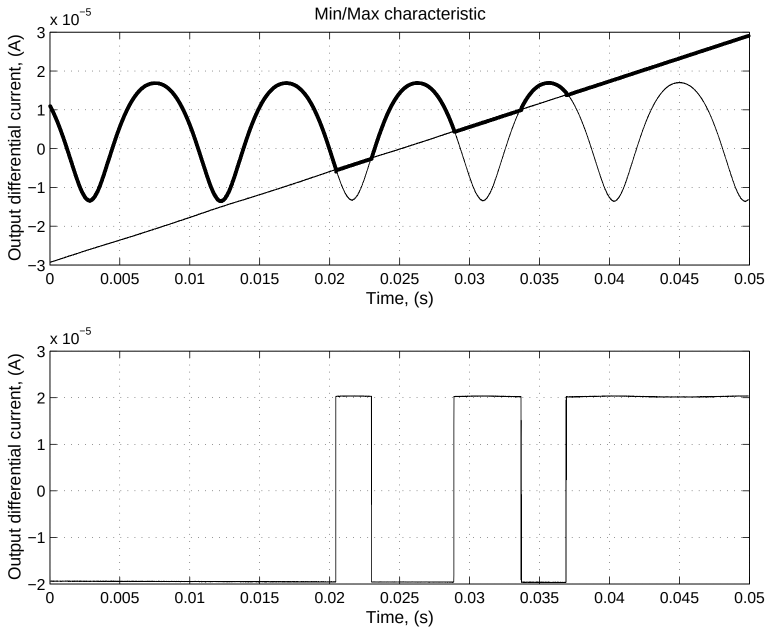

Figure 5.20: A typical programmable nonlinear block's characteristic when implementing the minimum and maximum functions. Outputs from the block.

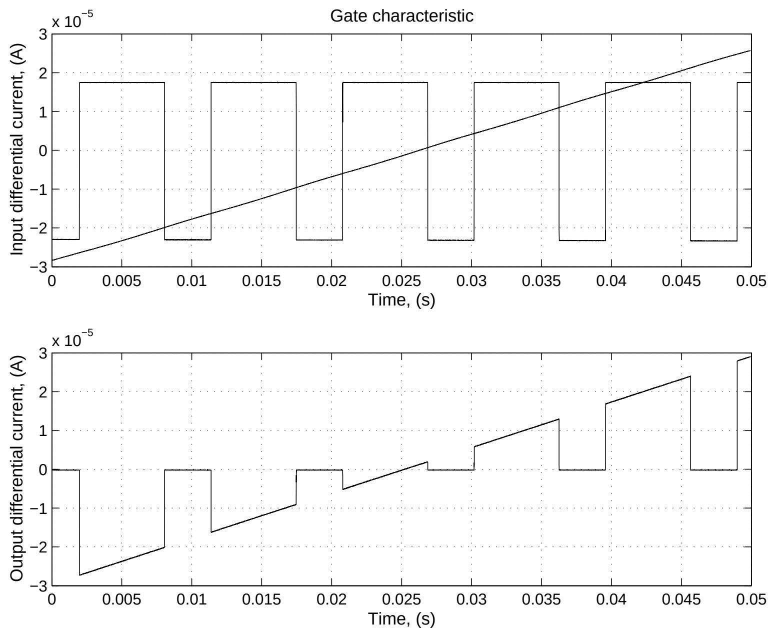

Figure 5.21: A typical programmable nonlinear block's characteristic when implementing the gate function.

and the sinusoid is applied to input-port 2). Fig. 5.20 shows the outputs of the block. The lower portion of Fig. 5.20 shows a square wave that indicates which input is greater. That is, the output signal shown is high when the signal at port 1 is greater than the signal at port 2. The upper portion of Fig. 5.20 shows the results of the minimum and maximum operations. The thick line shows the maximum, while the thin line shows the minimum of the block's two inputs.

The right most peak of the sinusoid in Fig. 5.19 is greater than the ramp. However, the circuit behaves as though the ramp is greater than the sinusoid's peak. This is due to one of the block's input ports having an offset.

The remaining functions the programmable nonlinear blocks can implement are: gate (or chopper), track-hold, and sample-hold. For these three functions, one of the inputs is processed by a current-mode comparator, comparing this input with a current of 0 A. The output of this comparator determines when the chopper function chops, when the track-hold function tracks and holds and when the sample-hold function samples and hold. The operation of these functions is described in more detail in Sect. 3.3.7. The comparator to which the following paragraphs make reference is this 0 A comparator.

Fig. 5.21 shows the time-domain characteristic of a representative programmable nonlinear block when it is implementing the gate function. The upper portion of the figure shows the block's input signal (ramp function) and the output of the circuit's "zero comparator". The comparator's output is high when the block's other input is greater than zero and low when the input is less than zero. The output in the lower portion of Fig. 5.21 is equal to the input when the comparator's output is high and is equal to zero when the comparator's output is low.

Fig 5.22 shows the time-domain characteristic of a representative programmable nonlinear block when it is implementing the track-hold function. The upper portion of the figure shows the block's input signal (ramp function) and the output of the circuit's comparator. The output in the lower portion of Fig. 5.22 is equal to the input when the comparator's output is high and is held when the comparator's output is low.

Fig 5.23 shows a representative time-domain characteristic of the programmable nonlinear block when it is implementing the sample-hold function. The upper portion of the figure shows the block's input signal (ramp function) and the output

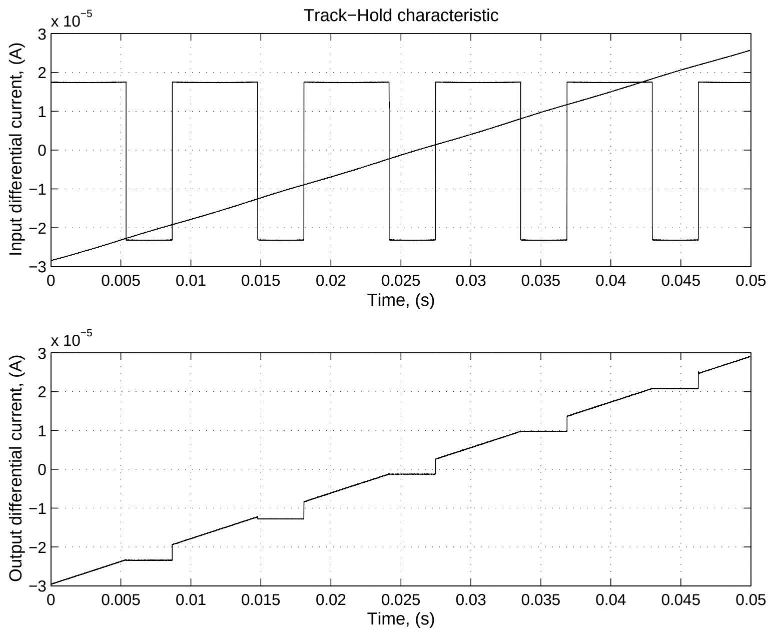

Figure 5.22: A typical programmable nonlinear block's characteristic when implementing the track-hold function.

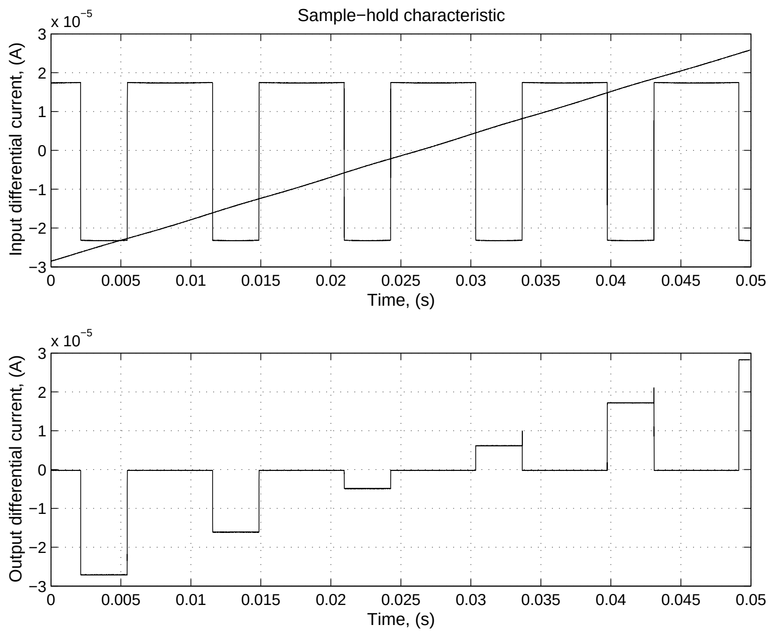

Figure 5.23: A typical programmable nonlinear block's characteristic when implementing the sample-hold function.

of the circuit's comparator, described above. The output in the lower portion of Fig. 5.23 is equal to zero when the comparator's output is high. The input is sampled when the comparator's output falls. The sample is held until the comparator's output rises.

5.8 Switches and Memory

As described in Chapt. 3, a design flaw that went undetected during simulation leaves 7 SRAM cells connected to each block's input unprogrammable. If one or more of the switches powers up in the ON state, that block's input is shorted to ground, rendering that block useless.

In initial measurements approximately 25% of blocks suffered this fate, which is far less than the $\frac{127}{128}$ fraction of blocks that would fail if the state of the cell into which it fell on power-up were a random process with equal likelihood. The chip has 2.5 V and 3.3 V power supplies. Since the 3.3 V controls the digital I/O and powers up the V_{dd} to which the electrostatic discharge (ESD) diodes are connected, it was powered up first, followed by the 2.5 V supply. Both were powered on by first connecting the supplies' cables, with the supplies off and then activating the supplies, with the sources already set to their appropriate voltages. By accident, the 2.5 V wires were once connected when the source was on. When this connection was done "hot", the vast majority of the blocks worked. The "hot" connection ramps up the supply voltage on the chip more quickly than when the source is activated while connected to the chip. This faster slew on the supply resulted in more of the switches powering up in the OFF state, leaving the vast majority of the blocks functional.

A double pole, single throw switch was put between the power supplies and the chip's PCB. This allows the supply voltages on the chip to be ramped up quickly and simultaneously. With this scheme nearly all circuits are functional. Interestingly, if many circuits' inputs are shorted to ground, the current consumption of the chip is higher, allowing a skilled user to observe this and power on the chip again until a lower current consumption is observed. This is of course unsatisfactory for the longterm utility of the chip.

As part of the calibration scheme, after the chip is powered on a routine measures the voltage at the input of every block, and determines whether the input has been shorted to ground. If a block is not functional, the software keeps track of it, so that a user cannot try to use it in a simulation. With such a large chip, even without this design flaw, one would want the interface software to determine which blocks meet specifications.

5.9 Power Dissipation

The chip, with all circuits active typically draws 100 to 120 mA from a 2.5 V supply meaning that its power dissipation is typically between 250 and 300 mW.

Chapter 6

Using the Analog Computer to Obtain Fast, Approximate Solutions

6.1 Solving Partial Differential Equations on the Analog Computer Using the Method of Lines

The method of lines was introduced in Section 2.2.1 of Chapter 2.

6.1.1 One-Dimensional Linear Diffusion Equation

The class of PDEs whose solution is discussed here is the one-dimensional diffusion equation, given by:

$$\alpha \frac{\partial^2 T}{\partial x^2} = \dot{T} \tag{6.1}$$

where $T(x, t)$ is a scalar function at a point x and time t. The coefficient α represents a physical property of the material. $\dot{T}$ denotes the time derivative of T.

Physical Interpretation: The two most frequently discussed physical examples of this equation are heat flow and current flow. In the former, T is the temperature along a uniform rod (Fig. 6.1), oriented in the x direction from 0 to L. The parameter α is equal to $\frac{k}{C}$ where k is the material's thermal conductivity and C is the material's specific heat capacity. It is assumed that heat flow is only in the x direction. That is, the temperature across the rod in the y and z directions is uniform and heat does not escape out the walls of the rod, except for possibly at the ends of it, depending on the boundary conditions of the problem.

In the second physical manifestation of this equation, T is the voltage along a uniform, distributed RC line and α is equal to $\frac{1}{RC}$, where R is the per unit length resistance and C is the per unit length capacitance of the line.

Discussion of Spatial Discretization Technique - Preliminaries to AC Solution

In investigating the analog computer's ability to solve the one-dimensional diffusion equation, the following things were considered:

- The accuracy of the solution for various numbers of discretization points.
- A comparison of two different spatial discretization techniques in terms of accuracy and the largest system that the chip can simulate using each of the two techniques.
- The effects on the overall solution accuracy of random errors in the coefficients of the discretized system and the effects of deterministic errors.

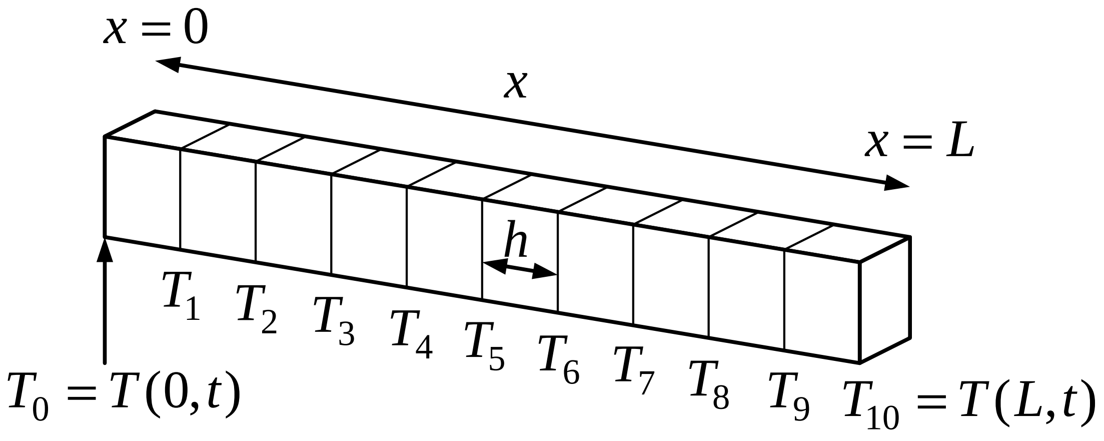

Figure 6.1: Discretization of solid rod.

The following boundary conditions were used:

$$T(L,t) = 0,\ -\infty < t < \infty \tag{6.2}$$

$$T(0,t) = 0,\ t < 0 \tag{6.3}$$

$$T(0,t) = 1,\ t \geq 0 \tag{6.4}$$

When the spatial partial derivative is approximated by the Forward and Backward Euler combination in Eq. 2.18, the following set of ODEs results:

$$\dot{\mathbf{T}} = \mathbf{A_E}\mathbf{T} + \mathbf{b_E}T_0 \tag{6.5}$$

The subscript E is used to denote that these are the terms associated with the Euler approximation for the spatial partial derivative. T_0 is equal to $T(0,t)$. $\mathbf{T}$ is a vector of temperatures of length n, (T_1, T_2,. . . ., T_{n-1}, T_n) where T_i is the temperature at the i^{th} cross-section of the rod. The rod extends from cross-section 0 to cross-section $n+1$, meaning that there are $n+1$ steps from $x = 0$ to $x = L$. Therefore, the distance between consecutive cross-sections, h, is $\frac{L}{n+1}$. The $n \times n$ matrix $\mathbf{A_E}$ is given by:

$$\mathbf{A_E} = \frac{\alpha}{h^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 1 & -2 & 1 & \ddots & & & \vdots \\ 0 & 1 & -2 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & 1 & -2 & 1 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 & -2 \end{bmatrix} \tag{6.6}$$

The length n column vector, b_E, can be written as:

$$\mathbf{b_E} = \frac{\alpha}{h^2}[1,0,0,\cdots,0]^T \tag{6.7}$$

where T denotes the transpose operation. According to Eq. 2.18, $\dot{T}_n$ depends on T_{n+1}, which does not appear in Eq. 6.5, since $T_{n+1} = T(x,L) = 0$, as stated in Eq. 6.2.

For the Central Differences approximation, the time derivative at T_i is a function of T_{i-2}, T_i, and T_{i+2}. This requires that the temperatures at T_0 and T_{n+1} be specified, as well as at T_{-1} and T_{n+2}. The simplest approach is to choose $T_{-1} = T_0 = 1$, for $t \geq 1$ and $T_{n+2} = T_{n+1} = 0$ for $-\infty < t < \infty$. Physically, this would correspond to there being portions of the rod extending beyond $x = 0$ and $x = L$, which are held at the same temperature as $x = 0$ and $x = 1$, respectively. When the spatial partial is approximated by using Central Differences twice (Eq. 2.19) and the assumptions above for T_{-1} and T_{n+2} are made, the following set of ODEs results:

$$\dot{\mathbf{T}} = \mathbf{A_C T} + \mathbf{b_{C0}} T_0 + \mathbf{b_{C1}} T_{-1} \tag{6.8}$$

The subscript C is used to denote that these are the terms associated with the Central Differences approximation for the spatial partials. $\mathbf{T}$ is a vector of temperatures of length n, (T_1, T_2,. . . ., T_{n-1}, T_n) where T_i is the temperature at the i^{th} cross-section along the rod. If Central Differences is used to approximate the partial at every point along x, the $n \times n$ matrix $\mathbf{A_C}$ is given by:

$$\mathbf{A_C} = \frac{\alpha}{4h^2} \begin{bmatrix} -2 & 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & -2 & 0 & 1 & 0 & & \vdots \\ 1 & 0 & -2 & 0 & 1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 1 & 0 & -2 & 0 & 1 \\ \vdots & & & 1 & 0 & -2 & 0 \\ 0 & \cdots & \cdots & 0 & 1 & 0 & -2 \end{bmatrix} \tag{6.9}$$

The length n column vector, $\mathbf{b_{C0}}$, can be written as:

$$\mathbf{b_{C0}} = \frac{\alpha}{h^2}[0, 1, 0, \cdots, 0]^T \tag{6.10}$$

The length n column vector, $\mathbf{b_{C1}}$, can be written as:

$$\mathbf{b_{C1}} = \frac{\alpha}{h^2}[1, 0, \cdots, 0]^T \tag{6.11}$$

We expect that if the rod is uniform in shape and in physical properties the temperature will reach a steady-state as $t \to \infty$ which will linearly decrease along the length of rod from 1 C at $x = 0$ to 0 C at $x = L$ when the boundary conditions in Eq. 6.2 to Eq. 6.4 are applied. This steady-state temperature can be found by solving Eq. 6.5 or Eq. 6.8 with the left side set to 0. That is, in steady-state, the temperature does not change and hence the time derivative of temperature is zero. Rearranging Eq. 6.5 when $\dot{T} = 0$ gives:

$$\mathbf{T_{E,final}} = -\mathbf{A_E}^{-1}\mathbf{b_E}T_0 \tag{6.12}$$

$\mathbf{T_{E,final}}$ is the steady-state temperature when the Euler discretization is used. Likewise, rearranging Eq. 6.8 when $\dot{T} = 0$ gives:

$$\mathbf{T_{C,final}} = -\mathbf{A_C}^{-1}(\mathbf{b_{C0}}T_0 + \mathbf{b_{C1}}T_{-1}) \tag{6.13}$$

$\mathbf{T_{C,final}}$ is the steady-state temperature when the Central Differences discretization is used. Solutions of Eq. 6.12 in Matlab have shown that it gives the expected linearly decreasing temperature as x increases; however, Eq. 6.13 does not. In the particular case for $n = 5$, the Euler approximation gives:

$$\mathbf{T_{E,final}} = [0.833, 0.667, 0.500, 0.333, 0.167]^T \tag{6.14}$$

whereas Central Differences gives:

$$\mathbf{T}_{\mathbf{C,final}} = [0.750, 0.667, 0.500, 0.250, 0.167]^T \tag{6.15}$$

Many of the properties of the Central Differences technique, which will be discussed later, can be retained, while forcing it to agree with the correct steady-state behaviour by using the Euler approximation for the first and last nodes giving:

$$\dot{\mathbf{T}} = \mathbf{A}_{\mathbf{C}^*}\mathbf{T} + \mathbf{b}_{\mathbf{C}^*}T_0 \tag{6.16}$$

The subscript C^* is used to denote that these are the terms associated with the Central Differences approximation for the spatial partial derivatives at all nodes except at T_1 and T_n. The matrix $\mathbf{A}_{\mathbf{C}^*}$ is given by:

$$\mathbf{A}_{\mathbf{C}^*} = \frac{\alpha}{4h^2}\begin{bmatrix} -8 & 4 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & -2 & 0 & 1 & & & \vdots \\ 1 & 0 & -2 & 0 & 1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 1 & 0 & -2 & 0 & 1 \\ \vdots & & & 1 & 0 & -2 & 0 \\ 0 & \cdots & \cdots & 0 & 0 & 4 & -8 \end{bmatrix} \tag{6.17}$$

The top and bottom rows of $\mathbf{A}_{\mathbf{C}^*}$ come from the use of the Euler approximation. However, the rows have entries of 4 and 8 rather than 1 and 2, as is found in $\mathbf{A}_{\mathbf{E}}$ because of the 4 in the denominator of the leading scaling factor in $\mathbf{A}_{\mathbf{C}^*}$. The length

n column vector, $\mathbf{b}^*_\mathbf{C}$, can be written as:

$$\mathbf{b}^*_\mathbf{C} = \frac{\alpha}{h^2}[1, 1, 0, \cdots, 0]^T \tag{6.18}$$

The steady-state temperature of this system of equations is given by:

$$\mathbf{T}_{\mathbf{C}^*,\mathbf{final}} = -\mathbf{A}_{\mathbf{C}^*}{}^{-1}\mathbf{b}_{\mathbf{C}^*}T_0 \tag{6.19}$$

which gives the correct steady-state temperature profile.

Under ideal circumstances, both the Euler and the modified Central Differences approaches, when implemented on the analog computer, would accurately predict the transient and steady-state response of the sets of ODEs. However, due to a variety of nonidealities, neither will produce the exact answer. The degree to which the analog computer's solution differs from the exact answer is determined by the accuracy of the circuits that implement the system and by the sensitivity of the system to those inaccuracies. To investigate the sensitivity of the systems to inaccuracies in the functional elements, the coefficients in the ODEs were varied and their effect on the solution of Eq. 6.12 and Eq. 6.19 was examined. The approach taken for this investigation is to change the coefficients in the ODEs and solve for the steady-state temperature using Matlab. This gives a prediction of how these errors, if present in the analog computer's circuits, would change the analog computer's steady-state solution of the ODEs.

Fig. 6.2 shows steady-state temperature profiles from several randomly generated sets of ODEs resulting from the two discretizations. In the top section, each

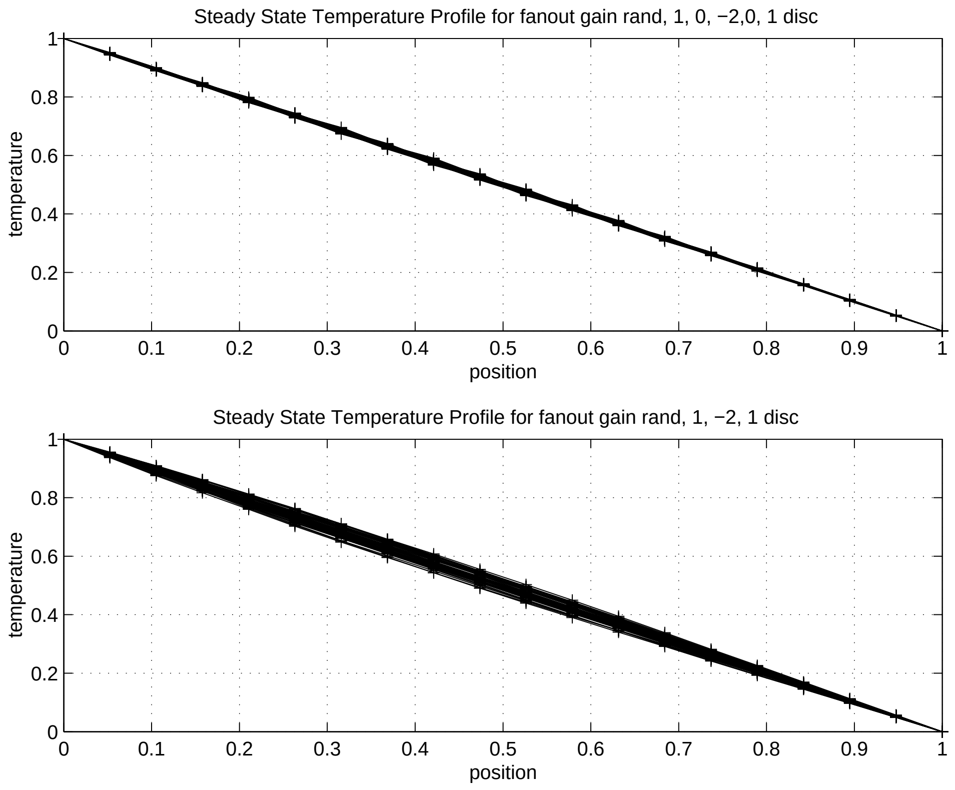

Figure 6.2: Steady-state temperature profiles for Central Differences (top) and Euler (bottom) with randomized coefficients.

coefficient in $\mathbf{A}_{\mathbf{C}^*}$ was scaled by a different sample of ϕ where:

$$\phi = 1 + 0.002\delta \tag{6.20}$$

where δ is a random variable with uniform distribution over a range of $(-\frac{\sqrt{12}}{4} < \delta < \frac{\sqrt{12}}{4})$ giving it a standard deviation $\frac{1}{2}$. In the bottom section, coefficients of $\mathbf{A}_{\mathbf{E}}$ were scaled in the same way and the resulting steady-state solutions were calculated. Clearly, the modified Central Differences technique is less sensitive to these random errors than the Euler technique.

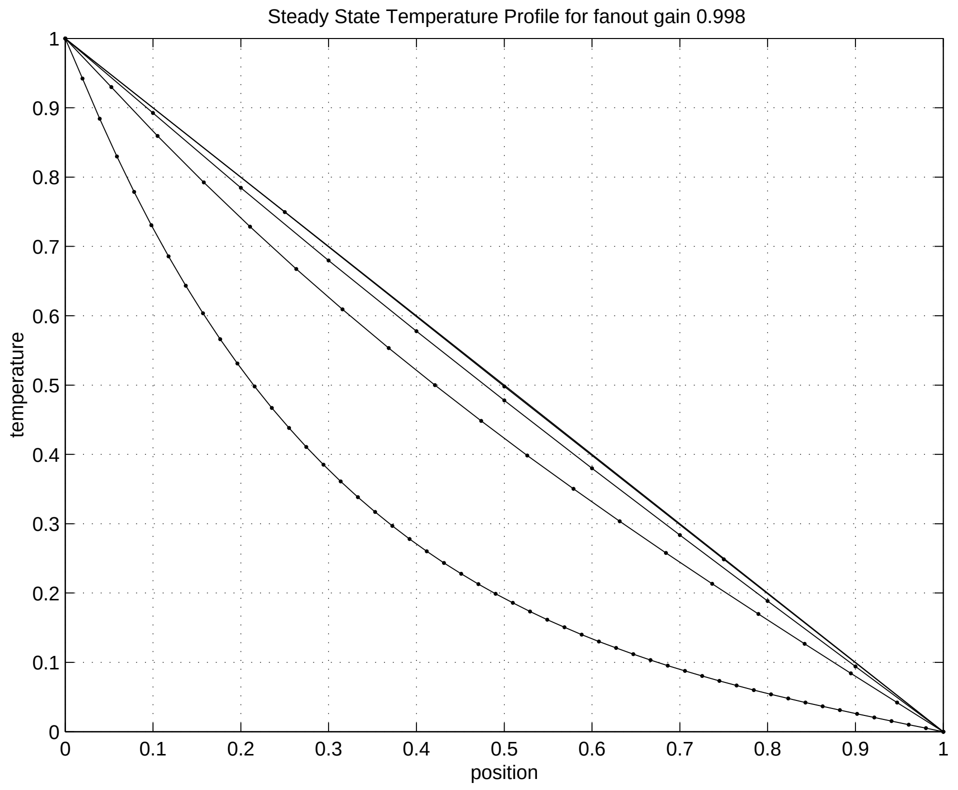

Figure 6.3: Steady-state temperature profiles for Euler discretization for n = 3, 9, 18 and 50. Coefficients scaled by 0.998 and 0.998^2. The most bowed line corresponds to $n = 50$ and the least bowed corresponds to $n = 3$.

When the ODEs are implemented on the analog computer, signals associated with the off-diagonal 1s in the A matrix pass through two fanout blocks whereas the signals associated with diagonal 2s pass through only one fanout block. To predict the effect of a systemic error in the gain of fanout blocks (for example: G=0.998 instead of 1.000), the off-diagonal elements were scaled by 0.998^2 and on-diagonal elements by 0.998. Curves in Fig. 6.3 show the steady-state temperature profiles for systems with systemic errors in the fanout gains, for different numbers of points, using the Euler discretization. Superimposed on the line representing the steady-state temperature when $n = 3$ is the ideal curve for the steady-state temperature, assuming all gains are correct ($G = 1$). Clearly, the larger the number of nodes into which the problem is discretized, the larger the steady-state error, when the fanout blocks that implement the coefficients have deterministic errors. The same can be said for the case when the blocks have random errors, though some of there errors cancel each other out resulting in smaller deviations from the ideal solution. These errors occur also for the Central Differences case, though the errors in steady-state temperature are smaller. As seen in Fig. 6.3, when n is large, the relative error in the steady-state solution is much greater than the error in the coefficients of the differential equation.

The choice in spatial discretization technique determines the largest number of nodes that can be simulated on this AC, and the availability of global wiring resources for directing signals off-chip. The architecture of the chip is detailed in Chapter 3 and the testing environment is described in Chapter 5; however, some aspects critical to this discussion are repeated here:

- There are 16 macroblocks in a 4×4 grid with 5 integrators and 10 fanout blocks in each. (Fig. 3.1)
- Below each row of macroblocks are 16 pairs of wires for routing signals between macroblocks and to off-chip.
- Beside each column of macroblocks are 16 pairs of wires for routing signals between macroblocks and from off-chip.
- The current measurement set-up has the capability to measure only seven analog

outputs at a time. These are connected to two outputs below the upper three rows of macroblocks and one output below the lowest row of macroblocks.

To reduce the use of global wiring resources, the five integrators in a given macroblock integrate the ODE for consecutive nodes along the rod. This corresponds to a macroblock implementing five adjacent rows in the state-space description of the ODE, that is a slice from T_i to T_{i+4}. The derivatives of this slice depend on T_i to T_{i+4} as well as T_{i-1} and T_{i+5}, which means that each macroblock (with the exception of the one implementing T_n) requires two global inputs, one from the macroblock implementing the slice from T_{i-5} to T_{i-1} and one from the macroblock implementing the slice from T_{i+5} to T_{i+9}. T_n's macroblock needs only one input since it is assumed that $T_{n+1} = 0$ Likewise, each macroblock (with the exception of the macroblocks implement macroblocks implementing T_n and T_1) needs two global outputs to direct signals to the macroblocks that implement adjacent slices in the ODE. The macroblocks implementing T_n's and T_1's ODE must each output only one signal to a neighbouring macroblock, since the former two macroblocks represent the ends of the rod.

With each of the four macroblocks in each row needing two outputs, eight of the horizontal global wires below the row of macroblocks are used, leaving eight for output to off-chip. The same numbers apply to the vertical global wires. All 80 integrators can be used and 32 output ports can be used. However, in the present test environment only 7 of the output ports can be measured, due to limits in the data acquisition card used. Even without this limit, to measure all 80 state variables would require that the system be simulated a few times consecutively, and that a subset of outputs be measured each time.

In the case of the modified Central Differences technique, each slice of 5 rows in the ODE, implementing T_i to T_{i+4} requires inputs from T_{i-2}, T_{i-1}, T_{i+5} and T_{i+6}. Therefore each macroblock needs four inputs and four outputs. This consumes sufficient resources so at to preclude using every macroblock. At most, 13 macroblocks can be used for a total of 65 state variables. This requires between 10 and 14 global horizontal wires for each set of 16, leaving a total of 18 outputs available.

Approximating partial derivatives with differences is similar to the process by which a distributed circuit is approximated by a lumped circuit. Fig. 6.4 shows

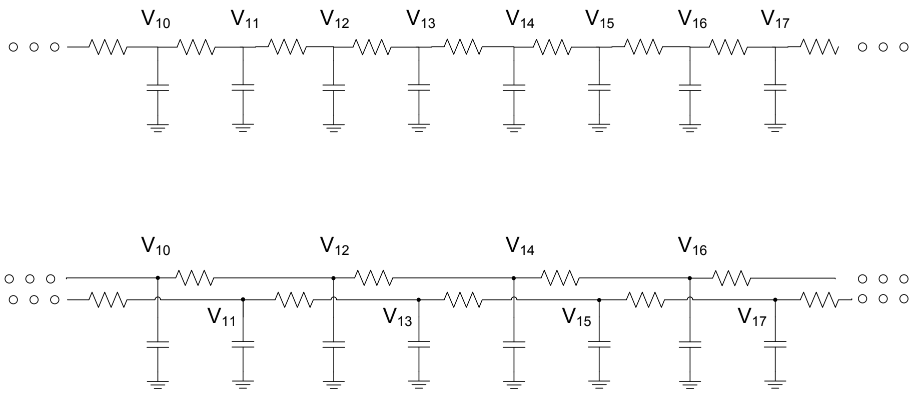

Figure 6.4: Lumped circuit equivalents of the distributed RC line.

two such examples. The upper portion of the figure shows the lumped circuit that corresponds to the Euler equations. The labels V_k denote the voltage at the k^{th} node. The k^{th} row in Eq. 6.5 corresponds to a Kirchoff Current Law equation written at the k^{th} node in the upper circuit, if each T_i in Eq. 6.5 is replaced by V_i. The lower part of Fig. 6.4 is a circuit whose electrical behaviour corresponds to the Central Differences approximation (Eq. 6.8).

System Implementation

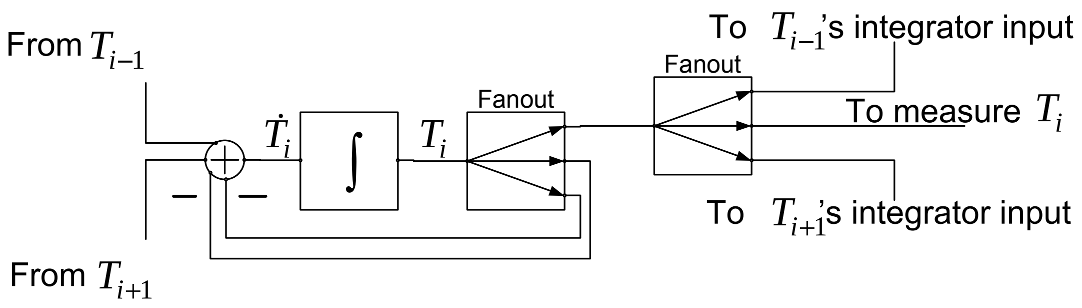

Figure 6.5: Per discretization point block diagram of the heat equation. Implementation 1.

The state-transition matrices of the above ODEs ($\mathbf{A_E}$, $\mathbf{A_C}$, and $\mathbf{A_{C^*}}$) have obvious patterns to them, and as such, the block diagram implementations of these systems have a high degree of regularity. For the case of the Euler method, one implementation for one discretization point is shown in Fig. 6.5, for the case when $\frac{\alpha}{h^2} = 1$. Fig. 6.5 represents one row in the $\mathbf{A_E}$ matrix, except the top or bottom row. Because the coefficients in $\mathbf{A_E}$ are integer multiples of one another, this ODE can be implemented using only integrators and fanout blocks. The coefficient of -2 is implemented by summing two outputs of a fanout, with negative polarity. By scaling the Eq. 6.5 (see Ch. 2) the cases in which $\frac{\alpha}{h^2} \neq 1$ can be handled with the implementation in Fig 6.5.

As noted in the previous section, the signals that implement the -2 coefficients are processed by one fanout block whereas the signals that implement the unity coefficients are processed by two fanouts blocks. As such, if there is a deterministic error in the gain ($G \neq 1$) of the fanout blocks, the signals implementing the unity coefficients are scaled by G^2.

The i^{th} row of Eq. 6.5, when $\dot{T}_i = 0$ is:

$$0 = T_{i-1} - 2T_i + T_{i+1} \tag{6.21}$$

The rearrangement of Eq. 6.21 gives:

$$T_i = \frac{T_{i+1} + T_{i-1}}{2} \tag{6.22}$$

In words, the steady-state behaviour of the heat equation is as follows: The temperature at the i^{th} node reaches the average of its neighbours, as shown in Eq. 6.22.

However, when the gain of the fanout is $G \neq 1$, and the per-node implementation shown in Fig. 6.5 is used, Eq. 6.21 becomes:

$$0 = G^2 T_{i-1} - 2GT_i + G^2 T_{i+1} \tag{6.23}$$

The rearrangement of Eq. 6.23 gives:

$$T_i = G\frac{T_{i+1} + T_{i-1}}{2} \tag{6.24}$$

In words this means that the $i^t h$ node reaches a temperature less than the average of its neighbours when $G < 1$. This equation predicts the downward bowing of the steady-state temperature profile shown in Fig. 6.3.

An alternative implementation to that in Fig. 6.5 is shown in Fig. 6.6. In this implementation, the signals implementing the -2 coefficients are scaled by G^2 and those implementing the unity coefficients are scaled by G. The equivalent expression for the steady-state temperature at a given node, in terms of its neighbours becomes:

$$T_i = \frac{T_{i+1} - T_{i-1}}{2G} \tag{6.25}$$

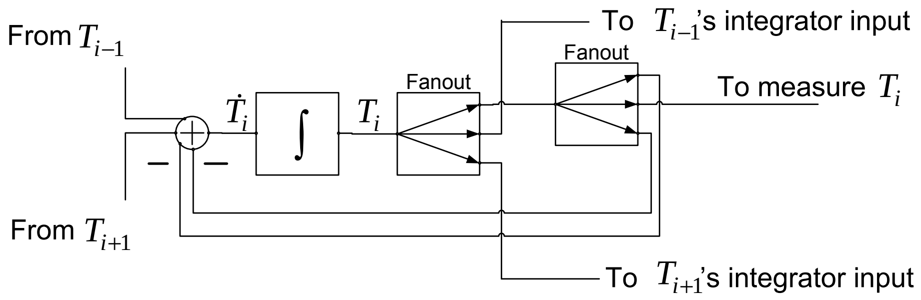

Figure 6.6: Per discretization point block diagram of the heat equation. Implementation 2.

When $G < 1$, this equation predicts an upward bowing of the steady-state temperature profile.

If the implementation for consecutive nodes alternates between the two, each row in the ODE has all of its elements scaled by the same coefficient - either G or G^2. If the i^{th} row uses the first implementation, the -2 coefficient is scaled by G, while the unity coefficients it supplies to rows $i-1$ and $i+1$ are scaled by G^2. Because the implementations alternate between rows, rows $i-1$ and $i+1$ supply unity coefficients to row i that are scaled only by G, and they supply -2 coefficients to themselves that are scaled by G^2. Accordingly, the net scaling of each row divides out when one solves for the steady state temperature, and the temperature at each node becomes the exact average of its neighbours.

Clearly, this interleaving of implementations in not obvious to the end user of the analog computer. However, this technique can be applied to higher dimensional PDEs and could be built into more sophisticated simulation software, thereby making the analog computer less sensitive to errors.

The effects of the deviations in fanout blocks' gains from 1 can be reduced be using gain blocks, with gains set to be the reciprocal of the fanout blocks' gains. For example, if one path of a fanout block has a gain of 0.998, it could be followed by gain block with a gain of $\frac{1}{0.998}$. However, there are twice as many fanout blocks as there are gain blocks, meaning that number of gain blocks would limit the possible size of the simulation.

Measured Results

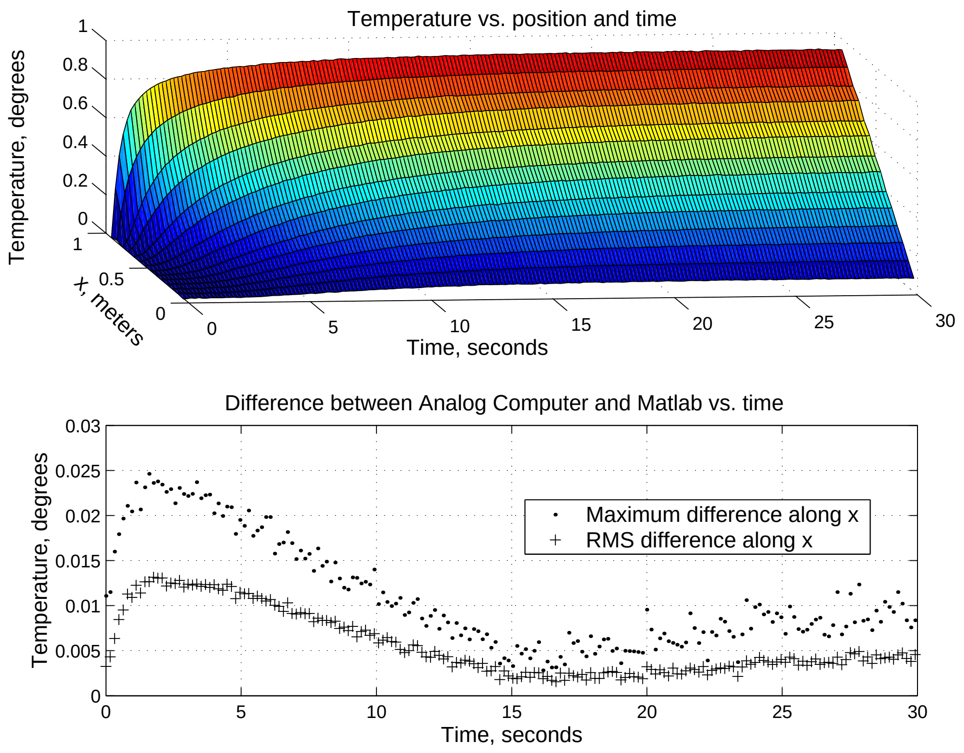

Figure 6.7: Results for an $n = 14$ one-dimensional heat equation.

Results for an $n = 14$ one-dimensional heat equation, discretized using the modified Central Differences method are shown in Fig 6.7. A numerical solution to the ODEs was computed using Matlab, against which the analog computer's results were compared. The lower portion of the figure shows the maximum error along the rod, as a function of time as well as the root mean squared error as a function of time. As seen, an RMS error of about 1 % results, with lower results as time increases.

The scaling is such that the 30 s of the solution in Fig 6.7 takes 1.2 ms to compute on the analog computer.

6.2 Solving Stochastic Differential Equations (SDEs) on the Analog Computer

6.2.1 Motivation for this Problem Area

Prediction of the noise behaviour of a system can be made in several ways. The noise in a linear system is processed linearly and predictions of the effects of input noise and internally generated noise can be made quickly and accurately using frequency domain techniques. Nonlinear systems are frequently linearized about a constant operating point and the same analysis is carried out on the linearized system as is done for a true linear system. This provides reasonably accurate results when the nonlinearities are soft (i.e., free of discontinuities), when the operating point is relatively unaffected by the input signal (i.e., the input signal is relatively small) and when the noise is small enough so as to not affect the operating point greatly. When the operating point changes significantly, the transfer functions by which the noise is processed also change.

Techniques have been developed to handle the noise analysis of systems with periodically time-varying conditions. For these, the time-varying operating point is computed. It is assumed that the noise does not affect this changing operating point, since the noise behaviour of the system is predicted by linearizing the system around the periodic operating point. These linear, or linearized, periodically time-varying systems are treated in a fashion similar to the linear, or linearized, time-invariant systems.

When the noise is large enough to influence the operating point, it is processed in a nonlinear fashion. In these situations, accurate results are only achieved through transient simulations. Usually, the detailed response of a system to a particular noise signal is not of interest but rather, the statistics of the solution, given the noise sources' statistics are of interest.

Stochastic differential equations contain random variables whose power spectral densities are white are have infinite bandwith. This is the same as saying that the random variables have autocorrelation functions which are dirac delta functions at the origin. This poses a problem for both numerical analysis and for simulation on an analog computer. In the former, to perform a transient simulation would require taking infinitely small time steps and in the latter, the exact equation cannot be simulated, since no white noise source exists which has infinite bandwidth. However, in both domains, reasonable transient simulations can be done. On a digital computer, some estimation is made of the necessary bandwidth of noise that needs to be accounted for, which in turn dictates how small the time steps in the simulation must be. For analog simulation, the same bandwidth estimation can be done, which dictates the specifications of the noise source to be used, in relation to the nominal time constants of the integrators in the computer. Stochastic differential equations appear in a wide range of fields from finance to material science. A discussion of more sophisticated mathematical techniques can be found in [22].

6.2.2 First Order Nonlinear SDEs: Particle in a Potential Well

Mathematical Description: The differential equation investigated here is an example of an Ornstein-Uhlenbeck process and is given by:

$$\dot{x} = -\nabla U(x) + n(t) \tag{6.26}$$

∇ denotes the gradient operation. For this example, x is a scalar and $U(x) = 0.185x^4 + 0.0139x^3 - 0.251x^2$ giving $\nabla U(x) = 0.740x^3 + 0.0416x^2 - 0.502x$. The function $n(t)$ is a random variable with zero mean and a Gaussian distribution.

Physical Interpretation and Qualitative Behaviour: $U(x)$ is referred to as a potential function. This system describes a particle moving in a double potential well. x is the horizontal displacement of the particle, while $\nabla U(x)$ is the steepness of the well. For this example, the cubic gradient has roots at: -0.852, 0.796 and 0. The first and second root give rise to stable equilibria. This means that in the absence of large noise, if the particle is near one of these roots, it will stay near one of them, and if the noise were reduced to zero, the particle would converge to one of these values of x. The root at zero is an unstable equilibrium, and hence, infinitesimally small noise will perturb the particle away from it. This is analogous to an inverted pendulum. Mathematically, the inverted vertical state is a solution, but the smallest noise will disturb the pendulum from its balance.

Measured Results

The function $n(t)$ was generated by a Noisecom noise generator, whose noise was amplified. The purpose of this experiment was to investigate the degree to which the analog computer can simulate a noisy differential equation, and not to investigate the degree to which the combination of the noise source and the amplifier produces white noise. To best compare the analog computer to a digital computer, the noise function used in the digital computer was a series of samples taken from the noise signal that was applied to the analog computer chip.

For a problem of this sort, mathematicians are usually interested in the statistics of the solution. Simulations using the analog computer and Matlab were conducted with the variance of the Gaussian noise source, $n(t)$, at 11 different levels. The simulations were repeated 20 times for each noise level over the interval of time (0 to 5000 s). A plot of $x(t)$ vs. t resulting from the analog computer's solution of the ODE is shown in Fig. 6.8.

When the noise is small, x remains close to one of the stable equilibria and transitions from one well to the other are infrequent. This is visible in the time

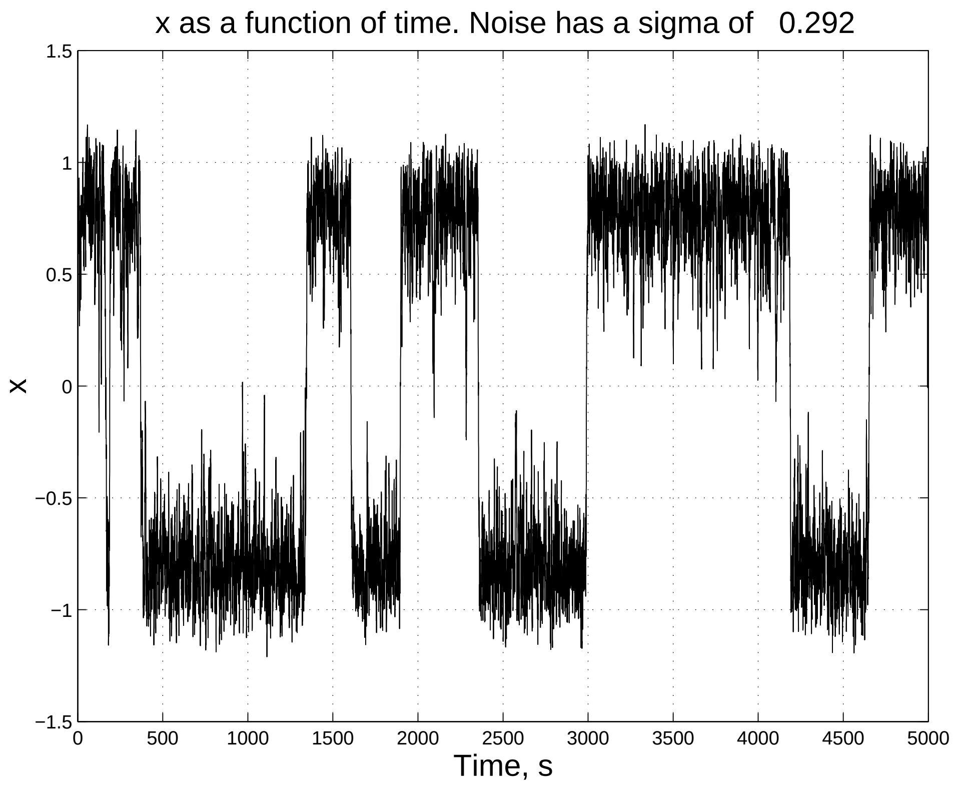

Figure 6.8: First order nonlinear SDE. Small noise ($\sigma_{n(t)} = 0.292$). Time domain solution computed by the analog computer.

domain plot Fig. 6.8. The relatively small number of transitions over the simulation interval necessitates long simulation intervals to generate meaningful statistics. If statistics were based on a shorter simulation interval, they could falsely suggest that the particle is limited to only one well.

In addition to examining the time domain behaviour of $x(t)$, probability density functions (PDFs) of $x(t)$ were computed. Strictly speaking, these PDFs are histograms of $x(t)$ over 50 bins. In Matlab, it is important that the samples of $x(t)$ are for equally spaced values of t. Equal temporal spacing in samples is achieved automatically for analog computer simulations, since the output of the analog computer is sampled by an analog to digital converter with a fixed sampling rate. Histograms computed from a Matlab simulation and an analog computer simulation can be found in Fig. 6.9.

Because x tends to be close to one of the two equilibria, the PDF is low near the origin and higher near the equilibria (Fig. 6.9). Agreement between Matlab's Simulink and the analog computer is good, as shown by the near coincidence of the two PDFs.

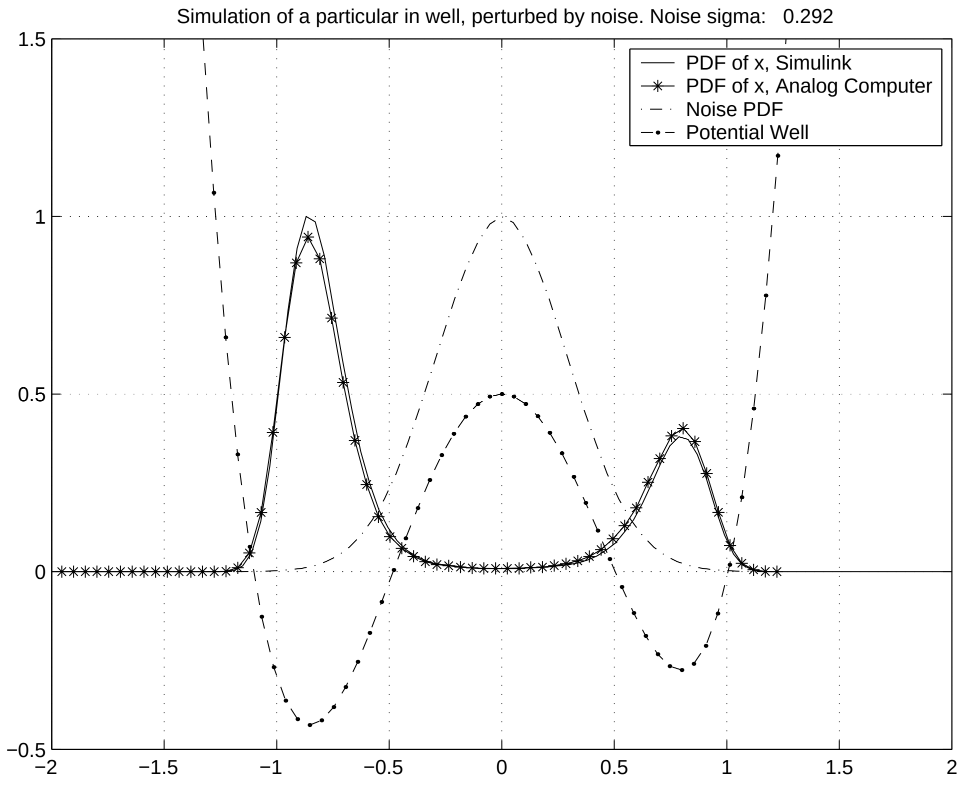

Figure 6.9: First order nonlinear SDE. Small noise ($\sigma_{n(t)} = 0.292$). Statistics.

Fig. 6.10 and Fig. 6.11 show results for the same SDE, but with somewhat larger noise. As expected, the frequency of transistions of x from one equilibrium to

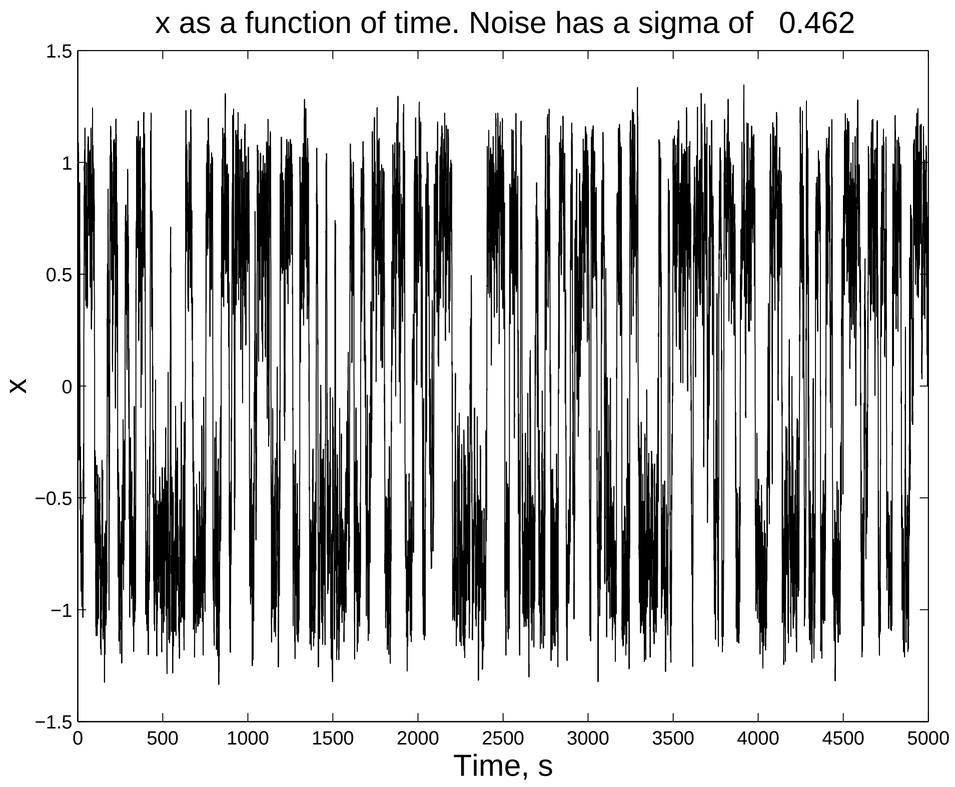

Figure 6.10: First order nonlinear SDE. Medium noise ($\sigma_{n(t)} = 0.462$). Time domain solution computed by the analog computer.

the other is greater.

Fig. 6.12 and Fig. 6.13 show results for the same SDE, but with even larger noise. Note that because transitions from one equilibrium to the other are so frequent, the horizontal axis in Fig. 6.12 was expanded.

Speed of Computation: In an effort to make a fair comparison, every ODE solver in Matlab was tried and the speed numbers reported below are for the fastest one. Tolerances were also relaxed to speed up Matlab, without introducing undue errors. Speed in Matlab will be determined by the time step of the simulation, which could be forced small by shortening the sampling interval of the output of the noise

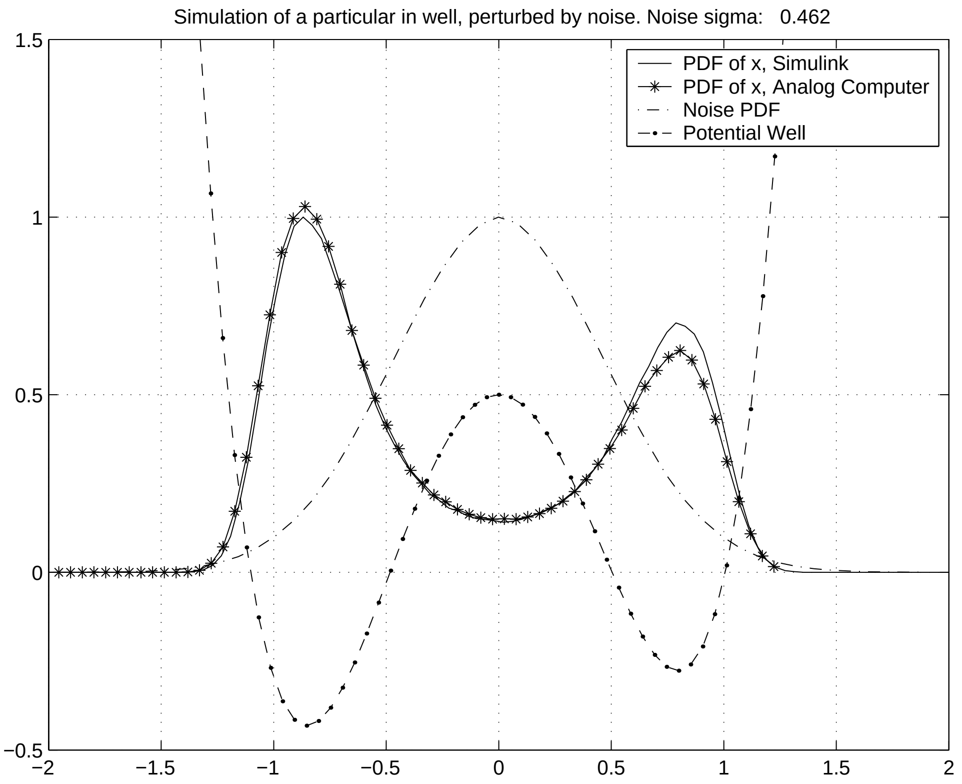

Figure 6.11: First order nonlinear SDE. Medium noise ($\sigma_{n(t)} = 0.462$). Statistics.

block. The integrators on the analog computer had a nominal time constant of 40 μs and the noise source was sampled at 1.25 MS/s, giving a sampling period of 0.8 μs. This ratio of 50 noise samples per integration time constant was maintained on the digital computer.

The analog computer was able to compute the solution significantly faster than a digital computer running Matlab. The 20 simulations for a given noise level took a total of 96 s (running on a Sun Blade 1000), whereas the analog computer took 4 s, or less than 4 % of the time. However, this first order system used only the hardware in one macroblock. If all macroblocks were used, 15 other simulations could take place

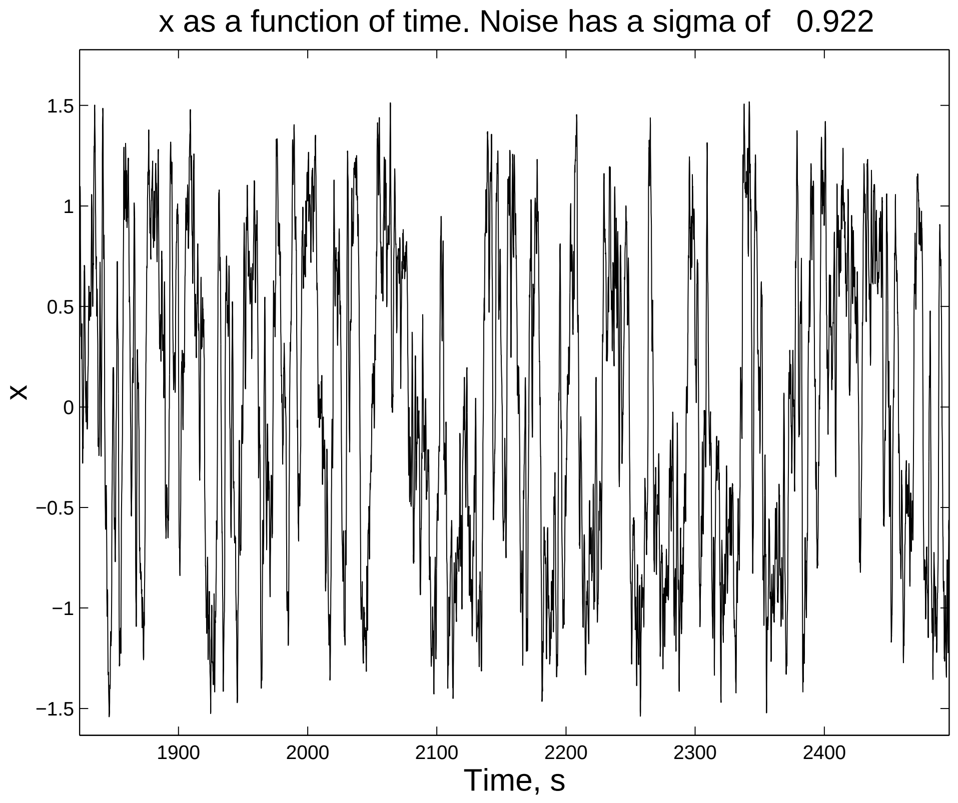

Figure 6.12: First order nonlinear SDE. Larger noise ($\sigma_{n(t)} = 0.922$). Time Domain solution computed by the analog computer.

simultaneously, allowing the analog computer to reach a solution in only 0.25 % of the time.

6.2.3 Heat Equation with Random Diffusivity

Equation and Physical Interpretation Eq. 6.1 describes one-dimensional heat flow when the parameter α is constant as a function of time and space. A more interesting problem arises when α is a random variable, varying with both space and time. In this section the discretized model was changed to model random thermal

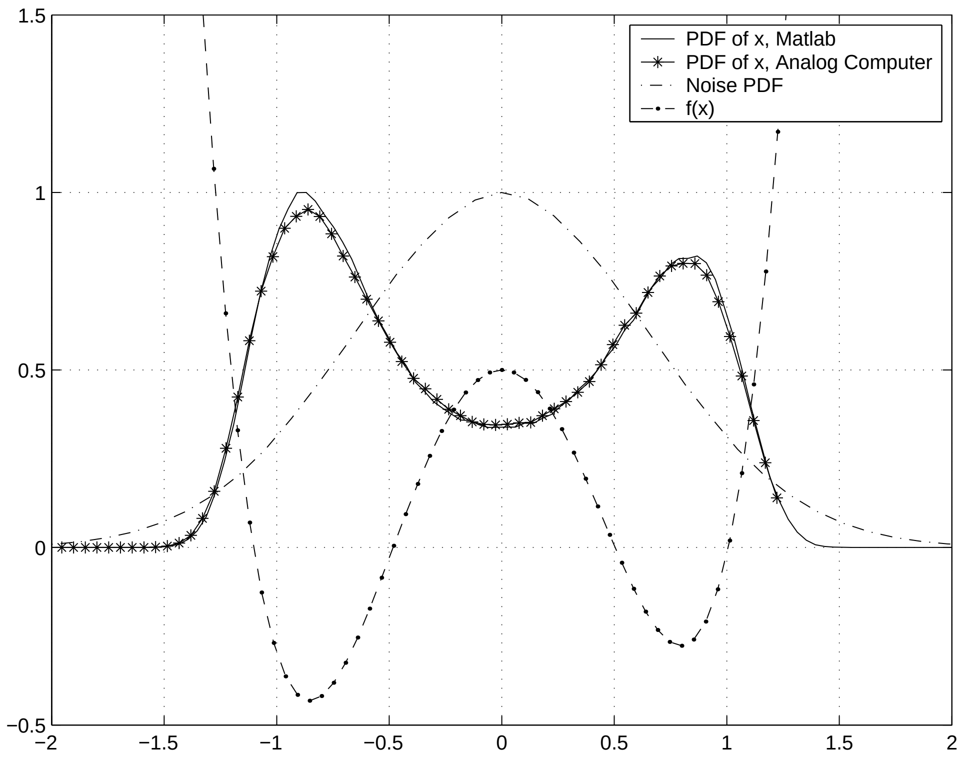

Figure 6.13: First order nonlinear SDE. Larger noise ($\sigma_{n(t)} = 0.922$). Statistics.

diffusivity between adjacent nodes. When α is no longer constant, each row of the Euler discretization can be rewritten as:

$$\dot{T}_i = \alpha_{i-1,i}(T_{i-1} - T_i) - \alpha_{i,i+1}(T_i - T_{i+1}) \tag{6.27}$$

assuming $h = 1$. $\alpha_{i-1,i}$ is the thermal conductivity of the section of the rod between the $(i-1)^{st}$ and i^{th} nodes. Those familiar with circuit analysis will see that this is simply equivalent to a Kirchoff Current Equation written at node i, assuming that $\alpha_{i-1,1}$ is the conductance separating nodes $i-1$ and i, $C = 1$, and T_i is the voltage at node i. See the electrical equivalent circuit shown in the upper portion of Fig. 6.4.

The complete set of ODEs is as follows:

$$\dot{\mathbf{T}}_{\mathbf{1,N}} = \alpha_{\mathbf{DU}}\mathbf{A}_{\mathbf{DU}}\mathbf{T}_{\mathbf{1,N}} + \alpha_{\mathbf{DL}}\mathbf{A}_{\mathbf{DL}}\mathbf{T}_{\mathbf{1,N}} + \alpha_{0,1}\mathbf{b}_{\mathbf{E}}T_0 \tag{6.28}$$

where N is the number of interior points in the discretization of the rod. $\mathbf{T}_{\mathbf{1,N}}$ is a length N column vector of temperatures at the discretization points. The N by N matrix $\alpha_{\mathbf{DU}}$ is given by:

$$\alpha_{\mathbf{DU}} = \begin{bmatrix} \alpha_{1,2} & 0 & \cdots & & \cdots & 0 \\ 0 & \alpha_{2,3} & 0 & & \cdots & 0 \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & \alpha_{N-1,N} & & 0 \\ 0 & \cdots & \cdots & & 0 & \alpha_{N,N+1} \end{bmatrix} \tag{6.29}$$

Each $\alpha_{i,i+1}$ is the thermal diffusivity between nodes i and $i+1$ in the discretized model of the rod. The N by N matrix $\mathbf{A}_{\mathbf{DU}}$ is given by:

$$\mathbf{A}_{\mathbf{DU}} = \begin{bmatrix} -1 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & -1 & 1 \\ 0 & \cdots & \cdots & \cdots & 0 & -1 \end{bmatrix} \tag{6.30}$$

The subscript DU denotes "diagonal, upper", since $\mathbf{A}_{\mathbf{DU}}$ has nonzero entries only on the main diagonal and the diagonal above the main diagonal. This notation is used for the matrix $\alpha_{\mathbf{DU}}$ to show the association between it and $\mathbf{A_{DU}}$. The N by N matrix $\alpha_{\mathbf{DL}}$ is given by:

$$\alpha_{\mathbf{DL}} = \begin{bmatrix} \alpha_{0,1} & 0 & \cdots & & \cdots & 0 \\ 0 & \alpha_{1,2} & 0 & & \cdots & 0 \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & \alpha_{N-2,N-1} & & 0 \\ 0 & \cdots & \cdots & & 0 & \alpha_{N-1,N} \end{bmatrix} \tag{6.31}$$

The N by N matrix $\mathbf{A_{DL}}$ is given by:

$$\mathbf{A_{DL}} = \begin{bmatrix} -1 & 0 & \cdots & \cdots & \cdots & 0 \\ 1 & -1 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & 1 & -1 & 0 \\ 0 & \cdots & \cdots & 0 & 1 & -1 \end{bmatrix} \tag{6.32}$$

The subscript DL denotes "diagonal, lower", since $\mathbf{A_{DL}}$ has nonzero entries only on the main diagonal and the diagonal below the main diagonal. This notation is used for the matrix $\alpha_{\mathbf{DL}}$ to show the association between it and $\mathbf{A_{DL}}$. The column vector $\mathbf{b_E}$ equals $\frac{1}{h^2}[1, 0, \cdots, 0]^T$.

Two sets of experiments were conducted. In the first, $\alpha_{3,4}$ was a random variable for a system with 7 internal nodes, and in the second, six αs were random variables. The boundary conditions were the same as were used in the deterministic PDE, and are stated in Eqs. 6.2 to 6.4.

Implementing Random Coefficients on the Analog Computer

In the deterministic case, each α corresponds to the gain of an amplifier that processes the difference in temperature of two adjacent nodes. In the deterministic example investigated, all values of α were 1, allowing the amplifiers to be omitted. To make a coefficient, α, time-varying requires the use of an amplifier of time-varying gain. This is implemented using a two-input multiplier with one input being α and the other is the difference between adjacent node temperatures. The function used for α was of the following form:

$$\alpha = 1 + n(t) \tag{6.33}$$

where $n(t)$ is a noise signal with zero mean. A diagram of the implementation of Eq. 6.33 is shown in Fig. 6.14.

Measured Results

The following are the results for the first example, where $\alpha_{3,4}$ is random. Several simulations of the transient response at the nodes on either side of the region with random diffusivity are shown in Fig. 6.15. If $\alpha_{34} = 1$, T_3 and T_4 would settle to 0.625 and 0.500, respectively, when there are 7 intermediate points. The nonlinear way in which the noise affects the solution is clearly visible in that the noise pulls each of T_3 and T_4 more in one direction than in the other.

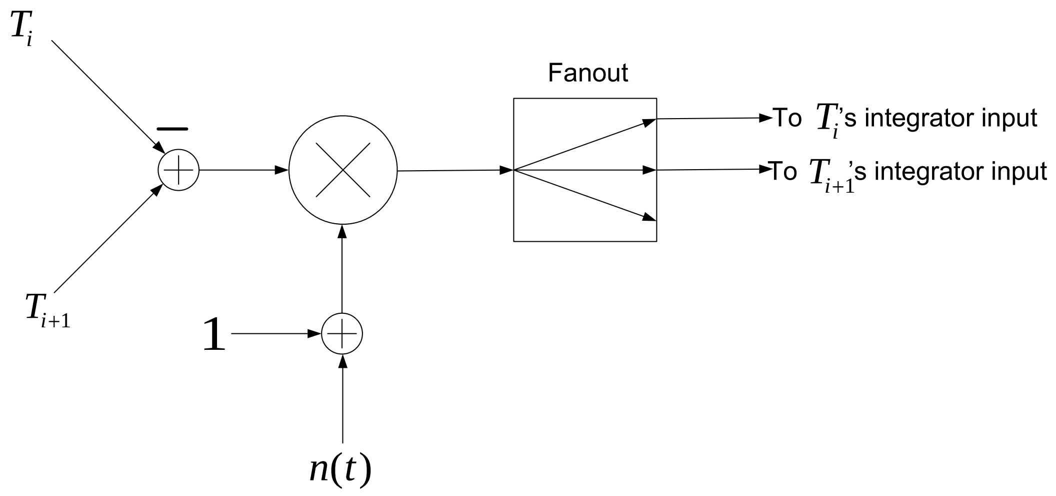

Figure 6.14: Circuitry for implementing a random coefficient.

Statistics were generated for the solution at each of the nodes for many simulations over an interval of time beginning when the system's response to the input step had reached a steady state. The start point of this interval was selected, qualitatively, to begin at $t = 40$. Statistics are shown in Fig. 6.16. Agreement is acceptable between the solutions generated by the analog computer and Matlab. For a linear noise, symmetrical noise sources will give rise to symmetrical distributions for the state variables. The asymmetry in the distributions is clear in the analog computer's solution and is testament to the need to perform transient simulations, rather than frequency domain simulations, for the noise behaviour of this system.

Likewise statistics of the solution to a system in which 6 of 7 nodes are random are shown in Fig. 6.17. These were generated for $t > 40s$.

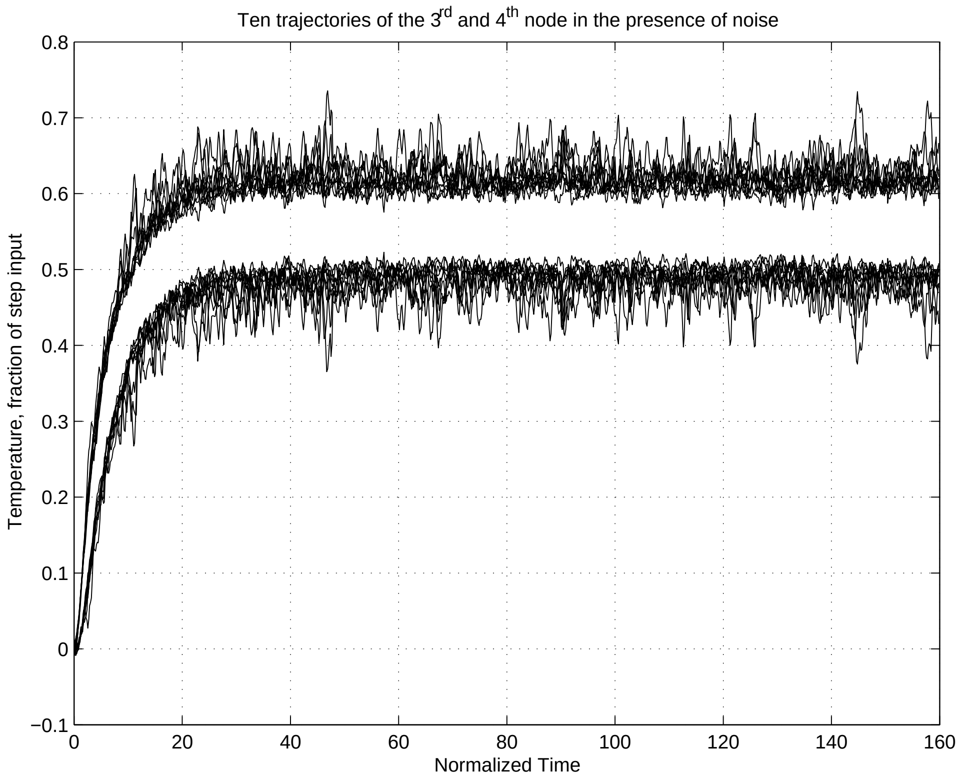

Figure 6.15: Transient response for T_3 (upper) and T_4. Generated by the analog computer.

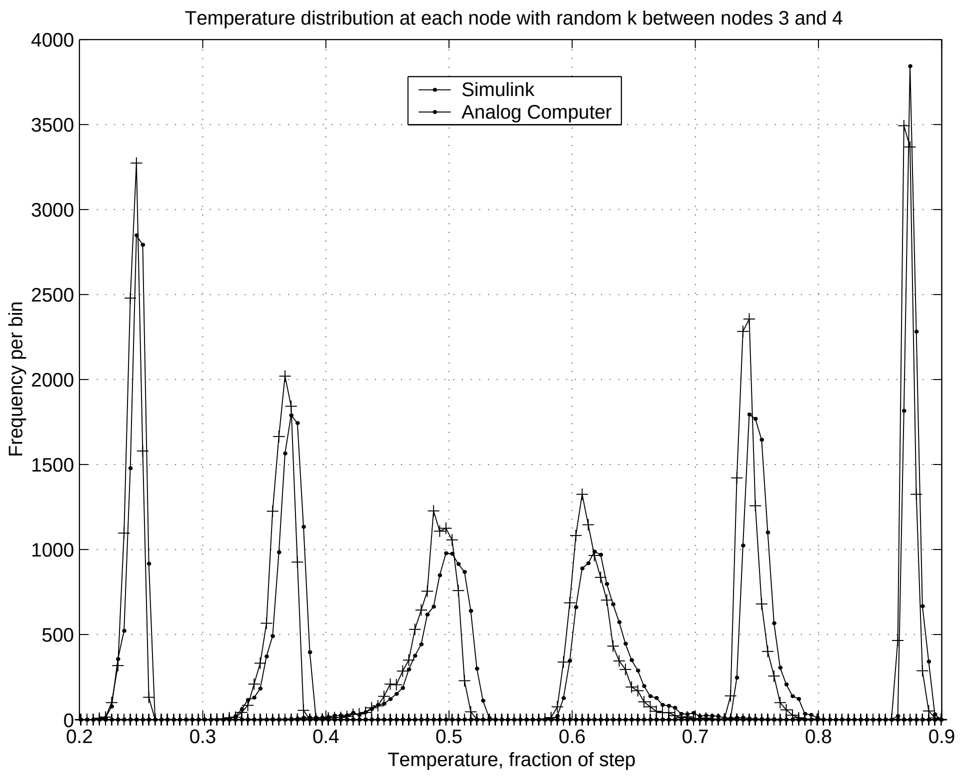

Figure 6.16: Probability density functions for the quasi-steady-state temperature at nodes T_1 to T_6 (left). One random coefficient.

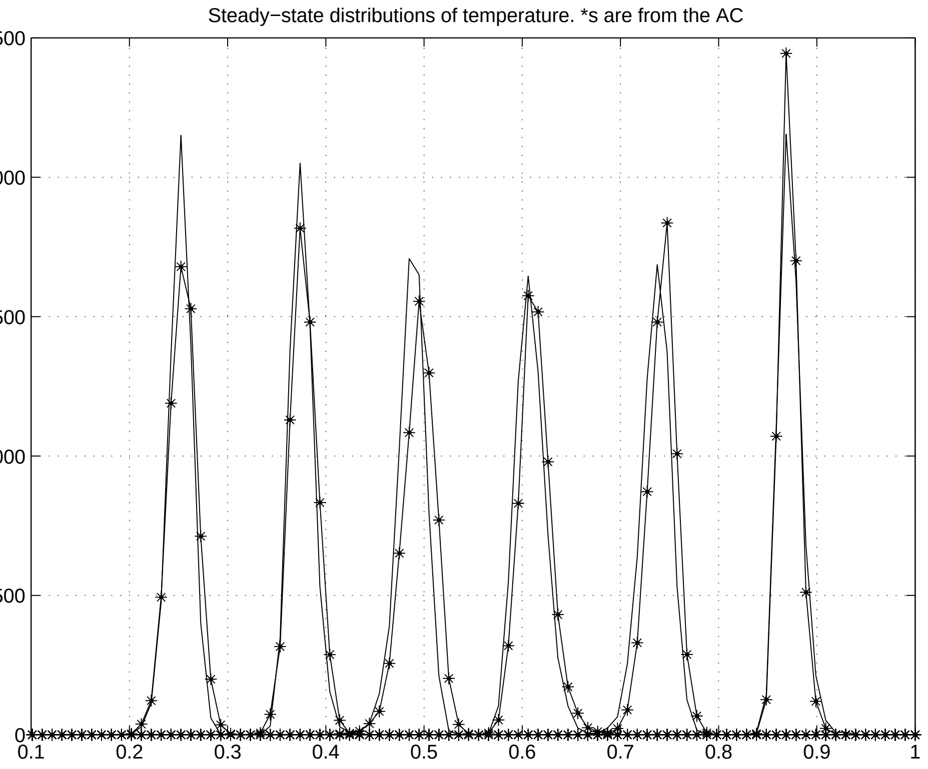

Figure 6.17: Probability density functions for the quasi-steady-state temperature at nodes T_1 to T_6 (left). Six random coefficients.

Chapter 7

Using the Analog Computer's Solution to Accelerate a Digital Computer's Algorithms

7.1 Motivation

Analog computers can find solutions to differential equations rapidly, albeit with only moderate accuracy. On the other hand, digital computers have the ability to reach arbitrarily high accuracy. However, if not used carefully, they may converge to a non-physical solution, may not converge quickly, or may not converge at all. There are ODEs that are particularly amenable to analog solution in that only a moderately accurate solution is necessary, and those which require sufficiently high accuracy so as to necessitate digital computation. One could solve the former with an analog system and the later with a digital system. However, the strengths of each approach can be utilized to a much higher degree if the analog computer is used to provide its solution to the digital computer, which will use the analog solution as a starting point for its numerical routine. This approach has the potential to speed up the digital computer's solution of ODEs for which high accuracy is needed, while avoiding some of the aforementioned convergence difficulties.

7.2 Newton-Raphson Based Periodic Steady-State Solvers

Engineers are frequently interested in the steady-state response of a nonlinear system to a periodic input. The condition for a system having reached this so-called periodic steay-state (PSS) is that all state variables at two times, separated by the period of the input T, are equal to one another. That is:

$$x(t+T) = x(t), \text{all}\, t \tag{7.1}$$

The condition in Eq. 7.1 means that the solution of the ODEs need only be calculated over one period, subject to Eq. 7.1. One period is discretized into n points. The derivatives at the last point will depend on the value at the first point, stemming from Eq. 7.1. If the system has m state variables, over the n points, there are a total of $m \times n$ unknowns. Some periodic steady-state solvers perform Newton-Raphson iterations on this vector of $m \times n$ unknowns.

Newton-Raphson iterations can be used to solve a set of nonlinear equations expressed in the form $f(x) = 0$. The technique iterates from an interim solution x_k of the equation toward the exact solution x_e. When the interim solution is near the exact solution, the routine exhibits quadratic convergence, meaning that:

$$\lim_{k\to\infty} \frac{|x_{k+1} - x_e|}{|x_k - x_e|^2} = K \tag{7.2}$$

for some nonzero constant K, where x_{k+1} is the interim solution at iteration number $k + 1$ and x_k is the interim solution at the previous iteration. However, when x_k is farther from x_e, convergence may be linear, or, if a local minimum or maximum separates x_k from x_e convergence may not occur.

The analog computer's solution for the forced Duffing's equation was used as a vehicle for investigating the degree to which a PSS routine could be accelerated. Duffing's equation [23] is as follows:

$$\dot{x} = y \tag{7.3}$$

$$\dot{y} = x(1 - x^2) + R\cos(\omega t) - \gamma y \tag{7.4}$$

The qualitative behaviour of this system depends on the parameters R, ω and γ. If $R = 0$, the system becomes an autonomous system that will either oscillate or will settle to one of its two stable equilibria, located at $x = \pm 1$ and $y = 0$, depending on how large γ is. For $R \neq 0$ the system will either oscillate periodically, or it will exhibit chaotic behaviour. Loosely speaking if the amplitude of the forcing function is small enough that x doesn't change sign, the solution approaches a stable limit cycle.

For fixed R, the transition from a stable limit cycle to chaos can be observed as γ is reduced. This parameter represents the damping of the autonomous system.

The digital computation of the PSS of the system proceeds in two steps. In the first step, the DC steady-state solution of the differential equation is computed assuming that the input source ($R\cos\omega t$) is equal to zero. This requires the use of a root-finding algorithm, such as Newton's method. This finds the solution to the nonlinear equation $f(z) = 0$, where $f(z)$ is the right-hand side of the state-space description. In this case, z is a vector of two unknowns, x and y. $z = 0$ is a typical starting point for this algorithm which in this case, leads to erroneous results since $z = 0$ satisfies the equation, but is an unstable equilibrium. To correctly find the DC steady-state the solution needs to be checked for stability, and if the routine has found an unstable equilibrium, a different starting guess must be used.

Typically, the DC steady-state solution becomes the starting guess for the actual PSS solver. In the case of Duffing's equation, the period, assuming its solution is periodic, is equal to $\frac{2\pi}{\omega}$. If this interval is discretized into 64 points, then the solution vector has a length of 128, since there are two variables. Newton's method is performed on this vector. If an unstable equilibrium is found for the DC steady-state, a non-physical PSS solution is also possible. Clearly, reasonable guesses are one way to avoid nonconvergence or convergence to a nonphysical solution.

For a one-dimensional equation, Newton's method is the following:

$$x_{k+1} = x_k - \theta \Delta x_k \tag{7.5}$$

Where Δx_k, referred to as the Newton step is:

$$\Delta x_k = \frac{f(x_k)}{f'(x_k)} \tag{7.6}$$

In the simplest form of the method for each iteration, $\theta = 1$ and the whole Newton step is taken. In more sophisticated schemes, values of θ between 0 and 1 are tried to see if an intermediate step gives a value of x_{k+1} that better satisfies $f(x_{k+1}) = 0$. If so, the intermediate step is taken. This has the benefit of preventing the routine from skipping over the solution and entering into a region in which convergence is less likely. These steps during which several values for θ are tried are computationally more expensive.

A program called PSIM based on algorithms in [15] was used in this investigation. It is a PSS solver that uses the two-step process described above. The number of iterations PSIM took varied based on the value of γ. In some cases, only 5 or 6 were needed to take the solution from the correct DC value to the PSS. However, as γ was reduced, the number of iterations increased. For the set of parameters of ($R = 0.4$, $\gamma = 0.67$ and $\omega = 2\pi$), PSIM took 37 iterations and 12.5 s running on a 2 GHz Pentium IV. However, when the routine started at a solution given by the analog computer, the number of iterations was reduced to 5 and the computation time to 0.76 s. The relative reduction in simulation time was greater than the reduction in the number of necessary iterations. This is because some of the iterations taken when the digital computer starts from a DC solution are the computationally more expensive routines described in the earlier paragraph. The analog computer's solution and the final solution computed by PSIM are both shown in Fig. 7.1, with the latter drawn

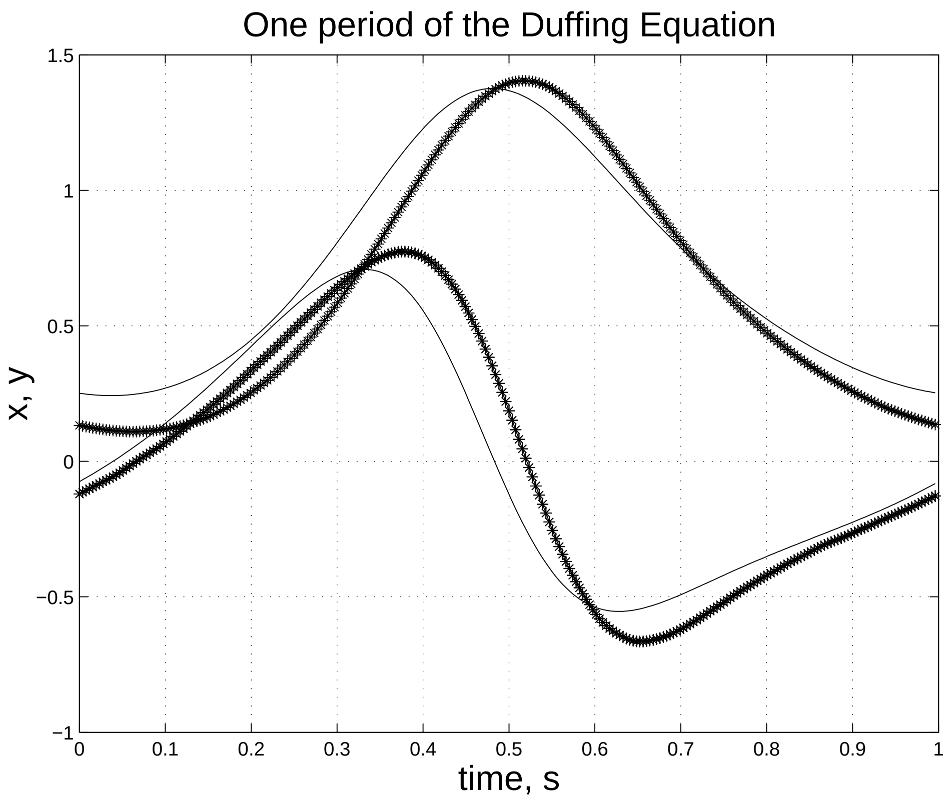

Figure 7.1: One period of the steady-state solution of the Duffing equation. $R = 0.4$, $\gamma = 0.67$ and $\omega = 2\pi$. The thick lines correspond to the analog computer's solution while the thin lines correspond to PSIM's solution.

with the thin lines. The state variable y is centered close to the time axis while x stays above the time axis.

An area for work is to extend this technique to larger systems whose PSS is desired and to apply this general concept to other numerical techniques.

Chapter 8

Performance Comparison Between Analog and Digital Techniques

8.1 Introduction

This chapter outlines some theoretical comparisons between digital and analog computers along two important performance criteria, namely power dissipation and computation speed. While speed is perhaps the most obvious metric, power consumption is becoming increasingly more important as the power consumption of digital computers increases. For portable applications, the consequence of increased power consumption is obvious: shorter battery life. However, even digital computers plugged into a wall socket have problems stemming from too high of a power dissipation, such as overheating and voltage drop due to IR losses in their power distribution network.

8.2 Energy Dissipation

The following assumptions have been made for this analysis:

- The analog computer's accuracy is adequate and it adequately solves the differential equation at hand.
- The power consumption overhead due to programming the analog computer or from the ADC/DACs is negligible; in other words, we limit this comparison to computationally intensive situations.
- Every floating point operation (FLOP) can be done on a digital computer in one instruction.
- All of the processing work done by the digital computer is carrying out floating point operations. That is, there is no overhead from instructions that are not executing floating point operations.

The last two assumptions are necessary simply because most data for the power efficiency of digital systems quantify energy per instruction. However, we can more readily gauge the number of floating point operations a routine performs. These assumptions let us count FLOPs but use the power data for instructions.

On the analog side, the total energy needed for a given computation is simply:

$$W = P_{AC}\Delta t \tag{8.1}$$

Where W is the total energy dissipation, P_{AC} is the power consumption of the analog computer and Δt is the duration of the computation. If the computation does not use all of the analog computer's blocks, P_{AC} is replaced by the consumption of only those blocks used. While power-down capability was not included in this design, a future design could easily be equipped with the necessary circuitry to power-down unused circuits.

For a digital computer, the power consumption estimation is more complicated. An estimation could be made in a similar fashion, using the duration of the simulation on a digital computer, and the processor's power consumption. Programs such as Matlab can show the elapsed CPU time of an operation or simulation. However, to estimate the power consumption of different digital devices it is more useful to estimate the number of floating point operations (FLOPs) required to carry out a simulation, and then scale this number by a given device's energy per FLOP.

The above technique allows the analog computer to be compared to various digital computers, in addition to more specialized digital hardware. The latter is important since the analog computer is somewhat specialized it is appropriate to compare it to specialized digital devices such as Digital Signal Processors (DSPs) or custom digital devices.

For this discussion, FLOPs denotes the plural of FLOP. FLOPs per second will be denoted by FLOPs/s.

Matlab has a number of ODE solvers. All of the routines take a function $f(y,t)$ describing the ODE: $\dot{y} = f(y,t)$. A Matlab function called FLOPS can be used to determine the number of FLOPs a routine takes. However, Matlab does not unsupported the function in Matlab 6 because the inclusion of the linear algebra package LAPACK makes this impractical. The rest of this investigation was done in the student release of Matlab 5, which supports this feature.

For some large systems, the use of LAPACK may reduce the number of FLOPs needed to invert matrices and perform other linear algebra functions. However, for the simple examples considered here, this is not the case, and FLOP analysis using Matlab 5 is appropriate.

In addition to FLOP count, the ODE solvers give a count of the number of each of the following:

1. Successful steps: This is the number of time steps at which the solution was evaluated.

2. Failed attempts: This is the number of time steps at which the solution failed or did not meet convergence criteria, resulting in a shorter time step being taken.

3. Function evaluations: This is the number of evaluations of $f(y,t)$.

4. Partial derivatives: This is the number of times the Jacobian, $\frac{\partial f}{\partial y}$ is computed

5. LU decompositions.

6. Solutions to linear systems.

The numbers of each of 3) through 6) per time step are influenced by the type of ODE solver used. For example, the last three are never done when an explicit routine is used. The time step is determined by tolerance requirements, the dynamics of the system, and the way in which any noise signals are represented.

To represent noise up to a given frequency, samples must be generated at least twice as frequently as the highest frequency of interested. This will force the time step of the simulation to be approximately as long as the spacing of the noise samples, when this spacing is much shorter than the system's shortest time constant. How the noise samples are interpolated further influences the time steps of the ODE solvers. For example, representing the noise as a zero-order hold (ZOH) of the noise samples can force smaller time steps since the noise changes abruptly at the steps in the ZOH. A linear interpolation (first-order) is computationally more expensive at each time step, but makes for smoother noise, a more accurate representation of continuous-time noise, and fewer FLOPs overall.

8.2.1 Example: First-Order Particle in a Well

The solution of this example (Eq. 6.26) was considered in Sect. 6.2.2. When solved with the routine $ode45$, with a relative tolerance ($relTol$) of 10^{-3}, over the interval $t \,\epsilon\, (0, 100)$, 1.6×10^6 FLOPs are needed. The interval $t \,\epsilon\, (0, 25000)$ can be simulated in 1 s on the analog computer, when the integrator's time constant is 40 μs. On the digital computer, this takes $\frac{25000}{100} \times 1.6 \times 10^6 = 400$ MFLOPs. The analog computer's circuits that are used in this simulation have a power consumption of approximately 7.8 mW. Therefore, the equivalent performance of the analog computer is $\frac{7.8\,\mathrm{mW}}{400\,\mathrm{MFLOPs/s}} \simeq 20$ pJ/FLOP, while a typical general purpose digital computer operates at closer to 10 nJ/FLOP to 100 nJ/FLOP [24].

8.2.2 Example: Stochastic Heat Equation

The solution of this example (Eq. 6.28) was considered in Sect. 6.2.3. This example, with one random coefficient was solved using the suite of ODE solvers in Matlab. To make a fair comparison, the right-hand side of Eq. 6.28 was coded explicitly, rather than with matrix multiplication. Each row requires only 3 FLOPs this way rather than the 19 FLOPs that are required for each row of 10 x 10 matrix multiplication. For this investigation all ODE solvers were used with a variety of tolerances. The fewest FLOPs needed to compute a time-domain solution visually equivalent to the solution when tighter tolerances are used was 2.8 MFLOPs (ode23 and $relTol = 10^{-3}$). This was over the interval 0 to 100 s. On the analog computer, this takes two marcoblocks (but only half the circuits in each macroblock), giving a power dissipation of about 15.6 mW, and a total equivalent performance of $\frac{15.6\,\mathrm{mW}}{2.8\,\mathrm{M}\times\frac{25000}{100}} \sim 22$ pJ/FLOP.

The performance criterion of "visually" equivalence was applied in the following way: For many ODE solvers, the solutions for $relTol = 10^{-3}$ and $relTol = 10^{-4}$ responded to the noise in a similar way. However, when the tolerance was relaxed to 10^{-2}, spikes due to the noise did not track those from the more accurate solutions. Frequently, spikes would overshoot the more accurate solution.

When there are nine random coefficients, the smallest number of FLOPs increased to 7.1 MFLOPs, bringing the equivalent power efficiency of the analog computer down to 8.9 pJ/FLOP.

Digital signal processors (DSPs) have a typical power efficiency of 100 pJ/FLOP to 1 nJ/FLOP [24]. Even custom digital ASICs, such as digital filters, have a power efficiency in the 10 pJ/FLOP range [24]. However, this analog computer has more programmability than a digital filter. The analog equivalent of a digital filter is, of course, an analog filter, which can be made to consume much less power than this device. This analog computer represents a first attempt at a large VLSI analog computer, and it is expected that future iterations would consume less power. Table 8.1 summarizes this energy dissipation analysis.

Device	Power Efficiency
	nJ/FLOP
Typical Microprocessor	2-100
DSP	0.1-1
This AC solving SDEs	0.008-0.022

Table 8.1: Energy dissipation comparison.

8.2.3 Fixed Point Versus Floating Point

To achieve the necessary accuracy for the solution of these sample problems, a digital computer may not need to perform computations in floating point. However, the analysis is still valid if the computations could be performed in fixed-point. This is because the custom ASICs mentioned above which have efficiencies in the range of 10 pJ/FLOP are fixed point devices. Secondly, our comparisons with microprocessors are biased in favour of the microprocessor due to the assumption that all FLOP take the same number of clock cycles. For example, many microprocessors can pipeline multiplication operations such that one operation is performed each clock cycle, however, very few can complete a division operation each clock cycle. In this analysis, all FLOPs are treated equally, regardless of its true complexity.

8.2.4 Power Overhead From Data Conversion

Using an energy per conversion per level of 1 pJ (a typical value for high-performance ADCs), measuring 80 state variables over 1 s with 10-bit resolution at 2 Msamples/s, leads to $80 \times 2^{10} \times 2 \times 10^{6} \times 1\,\mathrm{pJ} = 163\,\mathrm{mW}$. However, 10 bits may be an overkill. With the above taking place with only 8 bits, the resulting power dissipation is approximately 40 mW, or about 33 % of the chip's power consumption. This increases the energy consumption numbers calculated above by 33 %.

8.2.5 Comments on Sample Problems

Stochastic differential equations are a class of differential equations that are solved efficiently on the analog computer. The inclusion of high frequency noise greatly increases their computational load on a digital computer. However, the speed of the analog computer is unchanged by it. Further, instead of the exact solution being important, some statistical summary (mean, rms, or probability density function) is usually the goal of the simulation. This means that the moderate accuracy of the analog computer is likely to be adequate in many cases.

There are some limitations to the ability of the analog computer to predict the effects of high-frequency noise. The finite bandwidth of the memoryless blocks, and the presence of higher-frequency poles and zeros in the integrators' frequency response limit the range of frequecies over which the noise behaviour of the system can accurately be simulated. One way to extend this range is to lengthen the time constant of the integrators, thereby increasing the ratio of the bandwidth of the system to the unity-gain frequency of the integrators. This has the consequence of proportionally lengthening the simulation duration and decreasing the power efficiency of the analog computer. That is, once the input noise bandwidth has reached the bandwidth of the analog computer's memoryless blocks, to double the relative frequency of noise that can be simulated, the time constants of the integrators must also be doubled, causing the simulation to take twice as long, and reducing the power efficiency of the analog computer by a factor of 2. This is the same relative performance degradation that a digital computer suffers when it must take twice as many time steps, which is the case when the bandwidth of the noise is increased by two.

8.3 Computation Speed

When only about half of the blocks on the analog computer are used, it solves differential equations at a rate equivalent to a digital computer performing operations at a rate of as much as 14 GLOPs/s. Desktop personal computers, which seldom perform more than 1 FLOP per clock cycle, do not perform operations at this rate.

Chapter 9

Suggestions for Future Work

9.1 Circuit Enhancements

A revision to this analog computer chip should have blocks that meet more stringent specifications. In particular, the following need to be addressed:

- Offsets of all blocks need to be reduced. Small current-output DACs could be added to cancel static offsets in the blocks.
- Use of class-AB ports. A wider dynamic range of signals could be achieved by using class-AB circuits.
- Power-down mode for functional blocks.
- Reduced area of nonlinear blocks.

A wider range of blocks should appear on a future analog computer. In particular:

- Trigonometric functions.
- A digital logic block and flip-flops. This would allow for mixed-mode simulation.
- On-chip noise generators with the provision for controlling the frequency spectrum of the noise.

One of the most critical modifications to the present chip would be to correct the problem of the unused SRAM cells powering up in the ON state and shorting the inputs of blocks to ground. The simplest way to correct this would be to modify the layout of the existing SRAM/switch cell by removing the VIAS that connect the M5 wires down to the actual input side of the CMOS switches. The unused cells could easily be changed to use the modified layout and schematic.

9.2 System Modifications

The computation environment of a subsequent design should be modified to have:

- A parallel digital interface to the digital computer.
- A larger number of analog inputs and analog outputs connecting the digital computer and analog computer.
- On-chip DACs and ADCs with adequate memory on-chip.
- A more direct interface with the digital computer, perhaps incorporating the AC chip with the above modifications on a PCB that can plug into the digital computer's PCI bus.

- An inter-macroblock switching scheme that allows only a section of a global wire to be used for a particular connection. This would increase the number of inter-macroblock connections a given number of global wires could make.

9.3 Future Applications

One of the most promising application areas for future work is the solution of stochastic differential equations. Many interesting problems that are very time consuming to solve digitally require the solution of low-order equations. Other application areas that warrant more investigation include:

- Using the analog computer to implement the control algorithms of chemical reactions.
- The solution of PDEs using the method of characteristics.
- The solution of larger PDEs by connecting arrays of chips together.
- The solution of nonlinear programming problems.

Appendix A

Summary of Programming Data for the Analog Computer

A.1 Functional Blocks

A.1.1 Integrator's Programming Data

Each integrator accepts a 16-bit word consisting of the following:

- b0: Infinity mode. 0 selects dynamic offset cancellation. 1 selects static behavior.
- b1-b2: Input and output range: 00: 100 nA. 10: 1 μA. 01: 20 μA.
- b3-b4: Tuning current range: 00: 100 nA. 10: 1 μA. 01: 20 μA.
- b5: DAC out?. 1 routes the DAC's output to off-chip for possible measurement.

- b6-b15: DAC input.

A.1.2 VGA / 2-Input Multiplier's Programming Data

Each VGA / 2-input multiplier accepts a 16-bit word consisting of the following:

- b0-b1: Input range: 00: 100 nA. 10: 1 μA. 01: 20 μA.
- b2-b3: Output range: 00: 100 nA. 10: 1 μA. 01: 20 μA.
- b4: Mult: 0: circuit acts as a VGA. 1: circuit acts as a 2-input multiplier.
- b5: DAC out?. 1 routes the DAC's output to off-chip for possible measurement.
- b6-b15: DAC input.

A.1.3 Fanout Block's Programming Data

Each Fanout accepts an 8-bit word consisting of the following:

- b0-b1: Input range: 00: low. 10: medium. 01: high.
- b2-b3: Output range for output 1: 00: low. 10: medium. 01: high.
- b4-b5: Output range for output 2: 00: low. 10: medium. 01: high.
- b6-b7: Output range for output 3: 00: low. 10: medium. 01: high.

A.1.4 Exponential Block's Programming Data

Each Exponential block accepts a 2-bit word consisting of the following:

- b0-b1: Input range: 00: low. 10: medium. 01: high.

A.1.5 Programmable Nonlinear Block's Programming Data

Each pair of programmable nonlinear blocks is programmed by the following data:

- word0, b0-b1: Input range for x_1: 00: low. 10: medium. 01: high.
- word0, b2-b3: Input range for x_2: 00: low. 10: medium. 01: high.
- word0, b4-b5: Output range for y_1: 00: low. 10: medium. 01: high.
- word0, b6-b7: Output range for y_2: 00: low. 10: medium. 01: high.
- word0, b8-b9: Output range for y_3: 00: low. 10: medium. 01: high.
- word0, b10-b11: Single block function type for y_1 (when applicable).
- word0, b12-b13: Single block function type for y_2 (when applicable).
- word0, b14-b15: Combined block function type (when applicable)
- word1, b0: Combos? 1 indicates that the two nonlinear blocks will be used in tandem, rendering word0, b10-b11 irrelevant. 0 indicates that each nonlinear block implements a separate single input function, rendering word 0, b14-b15 irrelevant.
- word1, b1: Cal DAC0? 1 routes the output of DAC0 to off-chip.
- word1, b2-b3: Range for DAC0: 00: low. 10: medium. 01: high. DAC0 sets c_1.

- word1, b4-b13: DAC0 input.
- word1, b14: Cal DAC1? 1 routes the output of DAC1 to off-chip.
- word1, b15 - word2, b0: Range for DAC1: 00: low. 10: medium. 01: high. DAC1 sets c_2.

A.2 Programming Summary for Interblock Conncections

The following signals are used to program the state of the chip's switches, and the memory in each of the digitally programmed functional blocks:

- CLK: Active low signal used for programming the states of the switches.
- WR_EN: Write enable signal for programming switch states.
- RST: Reset signal for resetting the chip.
- a[0]: A signal which is pulsed to latch data into the memory of the functional block whose address is specified by a[1:5].
- a[1:15]: 16-bit address word.
- d[0:15]: 16-bit data word.

In more detail, the function of the a[0:15] is as follows:

- a[14] specifies whether the rest of the address word is referring to a location inside a macroblock (either internal switch states, or functional block memory) or adjacent to the macroblock.

- a[10:13] determine the macroblock to be programmed, either internally, or adjacent to it. a[10:11] encode the row (00=0, 01=1, 10=2, 11=3) and a[12:13] encode the column in the same way.

- a[1:9] encode the address of a functional block to be programmed, and the row and column address within switch memory. For the latter, a[1:3] determines the index of the block of eight rows in the memory to be programmed and a[4:9] determines the column (word line) to be programmed. In the former, a[1:5] determine which function block to program inside a macroblock. How these bits are interpreted depends on the levels of CLK, WR_EN, and a[0] signals.

- a[15] is used to program the column of switches that contains the global outputs of a macroblock.

Two different blocks of logic process WR_EN, a[0], a[10:14] and ultimately generate an a[0] signal, and WR_EN signals for the macroblock. The first block, GLOBAL_ADDRESS_DECODER, has one instance on the chip, while the second MB_DECODER appears 16 times, once for each macroblock.

The GLOBAL_ADDRESS_DECODER block decodes a[10:13] into two sets of four signals that, through one-hot encoding, specify the row and column of the macroblock to be programmed. It also produces global and local versions of the

WR_EN. These signals are applied to the WR_EN terminal of the memories adjacent to, and internal to a given macroblock. During a reset, if WR_EN is high, both global and local versions of the signal go high. The GLOBAL_ADDRESS_DECODER's behaviour can be summarized by the following truth tables:

a[10]	a[11]	row_sel[0]	row_sel[1]	row_sel[2]	row_sel[3]
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Table A.1: Truth table for GLOBAL_ADDRESS_DECODER's row_sel.

a[12]	a[13]	col_sel[0]	col_sel[1]	col_sel[2]	col_sel[3]
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Table A.2: Truth table for GLOBAL_ADDRESS_DECODER's col_sel.

WR_EN	a[14]	RST	WR_EN_G	WR_EN_L
0	X	X	0	0
1	0	0	0	1
1	1	0	1	0
1	X	1	1	1

Table A.3: Truth table for GLOBAL_ADDRESS_DECODER's write enable signals.

Signal row_sel[i] is applied to the MB_DECODER blocks for all macroblocks in the i^{th} row while signal col_sel[i] is applied to the MB_DECODER blocks for all macroblocks in the i^{th} column of the macroblock array. WR_EN_G and WR_EN_L denote global and local write enable signals, which are applied to every MB_DECODER block. The MB_DECODER block gates a[0], WR_EN_G and WR_EN_L in the following way:

SIG_TO_GATE	col_sel	row_sel	RST	GATED_SIG
A	X	X	1	A
X	0	0	0	0
X	1	0	0	0
X	0	1	0	0
A	1	1	X	A

Table A.4: Truth table for the MB_DECODER.

Only when a particular macroblock is selected via a[10:13] can its write enable signals and a[0] be raised. a[14] determines if the write enable signal controls the internal write enable or the external one.

When a[15] is low, the word lines in the external memory of a macroblock can be selected by a nor based decoder taking a[4:9] as its input. When a[15] is high, the word line decoder for the external memory is disabled and a[15] controls the word line for the SRAM cells controlling the states of the switches that connect global output wires to off-chip.

A.2.1 Reset

When the RST signal is raised, the chip's input signals a[0] and WR_EN are applied to all macroblocks, independent of the levels of a[10:14]. Within each macroblock, a default data sequence is applied to the memory in all functional blocks. When a[0] goes high, address lines a[1:5] and the lines that normally propagate their complements, all go high, causing every block's memory to be programmed by the default data. If *clk* is lowered while RST and WR_EN are high, all of the switch memory is reset such that all switches are open.

Bibliography

[1] Granino Arthur Korn and Theresa M Korn. *Electronic Analog and Hybrid Computers.* McGraw-Hill Book Company, 1964.

[2] Glenn Cowan, Robert Melville, and Yannis Tsividis. A VLSI analog computer / math co-processor for a digital computer. In *Proceedings of the ISSCC*, pages 82, 83, 586.

[3] Uri M Ascher and Linda R Petzold. *Computer Methods for Ordinary Differential Equations and Differential Algebraic Equations.* Society for Industrial and Applied Mathematicians, 1998.

[4] Edward K F Lee and Glenn Gulak. A CMOS field-programmable analog array. In *Proc. ISSCC*, pages 186–187, February 1991.

[5] Carver Mead. *Analog VLSI Implementations of Neural Systems.* Addison-Wesley Publishing Company, 1989.

[6] Piotr Dudek and Peter J Hicks. A CMOS general-purpose sampled-data analogue microprocessor. In *Proc. ISCAS*, volume 2, pages 417–420, May 2000.

[7] M I Sobhy and M Y Makkey. A new look at analog computing using switched capacitor circuits. In *Proc. ISCAS*, volume 1, pages 484–487, May 1998.

[8] Andrew Singer and Alan Oppenheim. Circuit implementations of soliton systems. *International Journal of Bifurcation and Chaos*, 9(4):571–590, 1999.

[9] J Chedjou, H Fotsin, P Woafo, and S Domngang. Analog simulation of the dynamics of a van der Pol oscillator coupled to a Duffing oscillator. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 48(6):748–756, June 2001.

[10] Walter J Karplus and Richard A Russell. Increasing digital computer efficiency with the aid of error-correcting analog subroutines. *IEEE Transactions on Computers*, 20(8):831–837, 1971.

[11] Kenneth Kundert. *A Designer's Guide to SPICE and SPECTRE.* Kluwer, 1995.

[12] John H. Mathews. *Numerical Methods for Mathematics, Science, and Engineering.* Prentice Hall, second edition, 1992.

[13] K G Nichols, T J Kazmierski, M Zwolinski, and A D Brown. Overiew of SPICE-like circuit simulation algorithms. *IEEE Proceeds on Circuits, Devices, and Systems*, 141(4):242–250, August 1994.

[14] Jacob K White and Alberto Sangiovanni-Vincentelli. *Relaxation Techniques for the Simulation of VLSI Circuits.* Kluwer, 1987.

[15] Kenneth Kundert, Jacob White, and Alberto Sangiovanni-Vincentelli. *Steady-State Methods for Simulating Analog and Microwave Circuits*. Kluwer, 1990.

[16] M Punzenberger and Christian Enz. A new 1.2 V BiCMOS log-domain integrator for companding current-mode filters. In *Proc. ISCAS*, pages 125–128, May 1996.

[17] Yannis P Tsividis. *Operation and Modeling of the MOS Transistor*. Oxford University Press, second edition, 1999.

[18] Barrie Gilbert. A new wide-band amplifier technique. *IEEE Journal of Solid-state Circuits*, 3(4):353–365, 1968.

[19] Marcel J M Pelgrom, Aad C J Duinmaijer, and Anton P G Welbers. Matching properties of MOS transistors. *IEEE Journal of Solid-state Circuits*, 24(5):1433–1440, 1989.

[20] Barrie Gilbert. A precise four-quadrant multiplier with subnanosecond response. *IEEE Journal of Solid-state Circuits*, 3(4):365–373, 1968.

[21] Roberto Alini, Andrea Baschirotto, and Rinaldo Castello. Tunable BiCMOS continuous-time filter for high-frequency applications. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 27(12):1905–1915, December 1992.

[22] Bernt Oksendal. *Stochastic Differential Equations*. Springer, 2003.

[23] Chihiro Hayashi. *Nonlinear Oscillations in Physical Systems*. McGraw-Hill Book Company, 1964.

[24] Robert Brodersen. System-on-a-chip design specification, architecture and prototyping. ISSCC SOC Short Course, 2003.