ANALOG COMPUTING ARRAYS

A Dissertation
Presented to
The Academic Faculty

By

Matthew R. Kucic

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Electrical Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
December 2004

ANALOG COMPUTING ARRAYS

Approved by:

Dr. Paul Hasler, Advisor
Asst. Professor, School of ECE
Georgia Institute of Technology

Dr. Martin Brooke
Professor, School of ECE
Georgia Institute of Technology

Dr. David Anderson
Professor, School of ECE
Georgia Institute of Technology

Dr. Alan Doolittle
Professor, School of ECE
Georgia Institute of Technology

Dr. Phillip Allen
Professor, School of ECE
Georgia Institute of Technology

Dr.Brad Minch
Professor, School of ECE
Olin College

Date Approved: August 2004

To my advisor Paul Hasler who convinced me of graduate studies and my wife Kelly Kucic who convinced me to stick with graduate studies.

ACKNOWLEDGEMENTS

I wish to thank my colleagues in the Integrated Computational Electronics lab for their encouragement and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Analog Computing Arrays (ACAs) provide a computation system capable of performing a large number of multiply and add operations in an analog form. This system can therefore implement several computation algorithms that are currently realized using Digital Signal Processors (DSPs) who have an analogues accumulate and add functionality. DSPs are generally preferred for signal processing because they provide an environment that permits programmability once fabricated. ACA systems propose to offer similar functionality by providing a programmable and reconfigurable analog system. ACAs inherent parallelism and analog efficiency present several advantages over DSP implementations of the same systems.

The computation power of an ACA system is directly proportional to the number of computing elements used in the system. Array size is limited by the number of computation elements that can be managed in an array. This number is continually growing and as a result, is permitting the realization of signal processing systems such as real-time speech recognition, image processing, and many other matrix like computation systems.

This research provides a systematic process to implement, program, and use the computation elements in large-scale Analog Computing Arrays. This infrastructure facilitates the incorporation of ACA without the current headaches of programming large arrays of analog floating-gates from off-chip, currently using multiple power supplies, expensive FPGA controllers/computers, and custom Printed Circuit Board (PCB) systems. Proof of the flexibility and usefulness of ACAs has been demonstrated by the construction of two systems, an Analog Fourier Transform and a Vector Quantizer.

CHAPTER 1

ANALOG COMPUTING ARRAYS

1.1 Analog Computing Arrays Benefits

Analog Computing Arrays (ACAs) provide a computation system capable of performing a large number of multiply and add operations in an analog form. This system can therefore implement several computation algorithms that are currently realized using Digital Signal Processors (DSPs) who have an analogues accumulate and add functionality. DSPs are generally preferred for signal processing because they provide an environment that permits programmability once fabricated. ACA systems propose to offer similar functionality by providing a programmable and reconfigurable analog system. ACAs inherent parallelism and analog efficiency present several advantages over DSP implementations of the same systems as demonstrated in Fig. 2. ACAs can be used to perform similar computations consuming orders-less power for the same computational functionality [51] or to perform similar computations in less time; permitting real-time computations. ACAs can also be used to perform similar computations using less area, such as when replacing multiple DSPs.

The computation power of an ACA system is directly proportional to the number of computing elements used in the system. Array size is limited by the number of computation elements that can be managed in an array. This number is continually growing and as a result, is permitting the realization of signal processing systems such as real-time speech recognition [51, 52, 30], image processing [22, 21], and many other matrix like computation systems. This research will attempt to provide manageability, as defined in latter sections, of at least one million floating-gate elements.

This research seeks to provide a systematic process to implement, program, and use the computation elements in large-scale ACAs. Once the system is fully developed it can be easily integrated on-chip in future systems. This infrastructure will facilitate the incorporation of ACA without the current headaches of programming large arrays of analog floating-gates from off-chip, currently using multiple power supplies, expensive FPGA controllers/computers, and custom Printed Circuit Board (PCB) systems. Proof of the flexibility and usefulness of ACAs has been demonstrated by the construction of two systems, an Analog Fourier Transform and a Vector Quantizer.

1.2 ACA Approach

The term Analog Computing Array (ACA) is used to describe a two-dimensional matrix of analog computation blocks [34]. The computation block most commonly used in our ACA systems performs a weighted multiplication then sum and is derived from floating-gate elements. Unlike mixer multiplication, where two signals are multiplied or mixed, we instead multiply a signal by a stored analog value (gain term). There are two terms that are interchangeably used for this stored value; weight, inherited from the neuromorphic field and coefficient, inherited from the DSP field. This weight, stored in each computation cell, can be individually addressed and programmed to any desired analog value [35]. This analog weight alters the computation in each cell providing a basis for programmable analog systems. When programmable computational elements are used in parallel the result is a powerful analog computing system as compared to the digital only counterparts as shown in Fig. 1. The array additionally can be scaled post-layout via programming to provide only the desired amount of computation power. This is achieved by fabricating more computation elements than needed and programming the unused cells off so they do not consume power. The power consumed is therefore optimized for the computation task. This also provides a method for overcoming fabrication defects when on the tester or systems that are fault tolerant in the field.

This technology was inspired from the development of analog matrix-vector computations used in neural network implementations [45], and research in data flow architectures

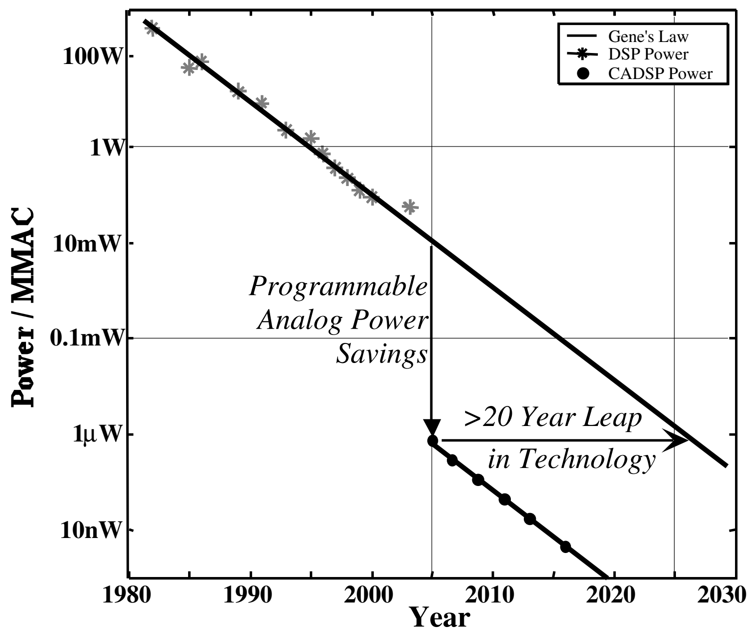


Figure 1. Motivation for ACAs for signal processing. This graph shows the computational efficiency (computation / power consumption) for DSP microprocessors, the extrapolated fit to these data points (Gene's Law [11]) , and the resulting efficiency for a programmable analog system. Two critical aspects for a competitive analog approach are dense analog programmable and reconfigurable elements, and an analog design approach that scales with digital technology scaling. The typical factor of 10,000 in efficiency improvement between analog and digital signal processing systems enables using low-power computational systems that might be available in 20 years.

of parallel processing [40]. Current digital signal processing architectures handle incoming data in chucks, fetching stored coefficients from remote memory and then serially applying these coefficients to the incoming data through some operation as demonstrated in Fig. 2. ACAs take a fundamentally different approach to processing data; instead using a parallel implementation to process the data in real-time and storing the coefficients locally in each computing block, removing the need for a fetch operation. In some cells the coefficient is actually stored in the computational element itself, made possible by using analog floating-gate devices [28]. We have termed this concept Computing in Memory, referring to the fact that the actual memory element performs the computation.

The use of floating-gate devices as a memory element is not new in the circuits community [32]. Since their discovery much effort has gone into making them a viable non-volatile memory element which we now find in wide array of products such as digital cameras and

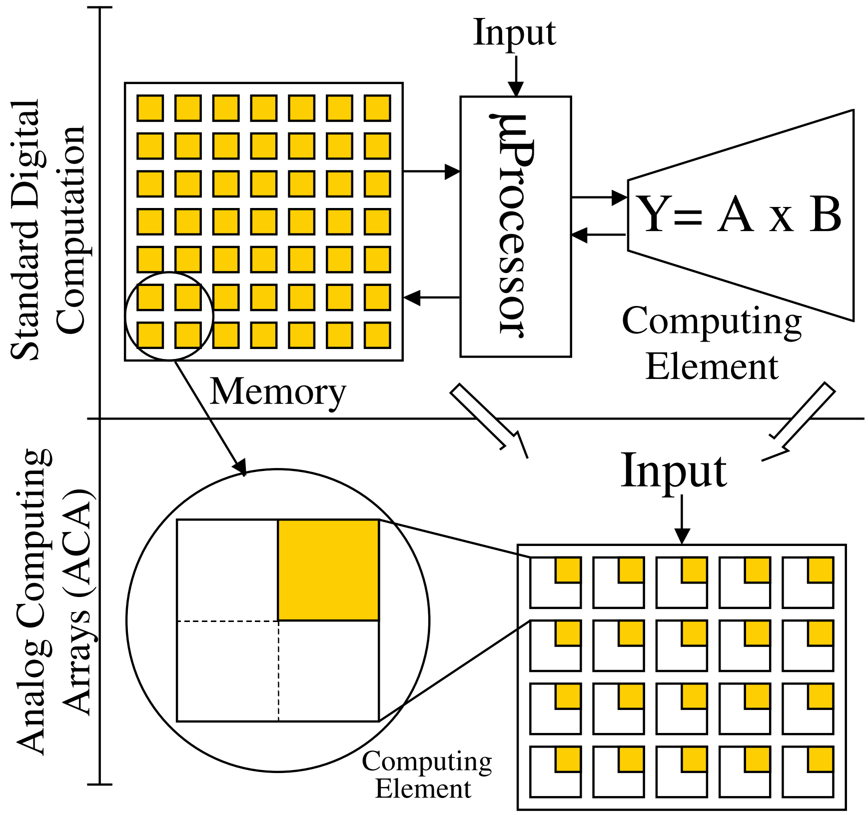


Figure 2. Illustration of computing in floating-gate memory arrays. A typical system is an array of floating-gate computing elements, surrounded by input circuitry to pre-condition or decompose the incoming sensor signals, and surrounded by output circuitry to post-process the array outputs. We use additional circuitry to individually program each analog floating-gate element.

mp3 players. More recently the beneficial use of floating-gates in analog circuits has been realized and it has been published that not only can they be used as memory devices but function as programmable, compact, computational elements [36]. Floating-gate devices have been recently touted as almost a magical elements in analog circuits, storing analog values [2, 14],and performing computations [28].

1.3 ACA Architecture

The memory cells in this architecture may be accessed individually, for readout or programming, or they may be used simultaneously for full parallel computation in applications such

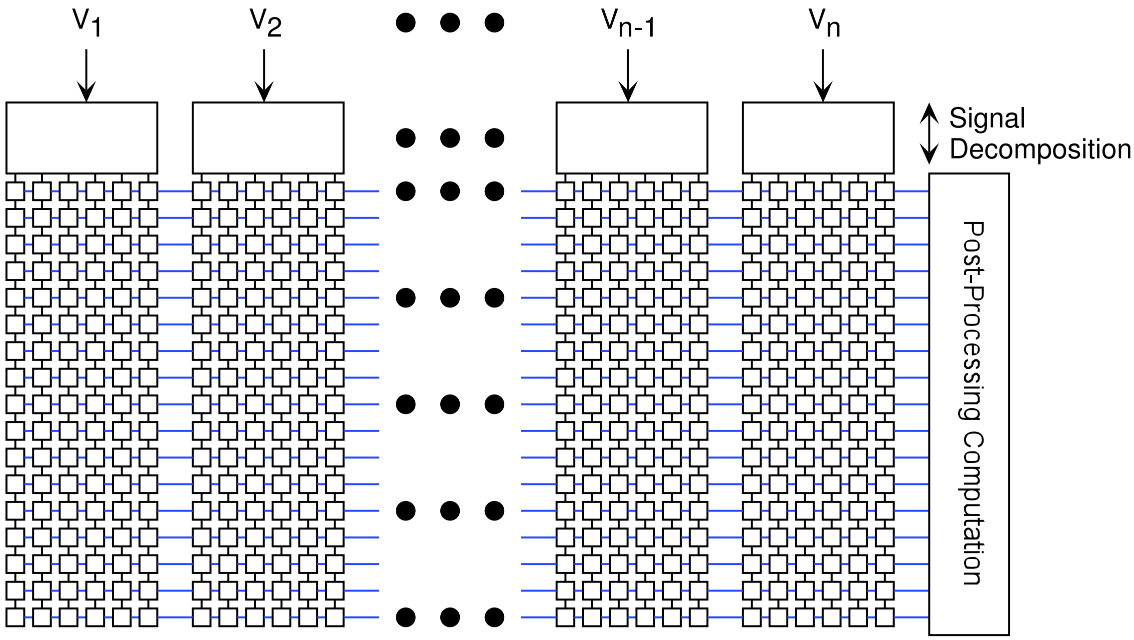


Figure 3. This demonstrates the computing array concept in visual block-diagram form. Before the signal is passed to the array blocks decompose the signal down into orthogonal channels. The array performs a fully-parallel multiplication on each channel and then sums up the results. Additionally, many separate multiply and sums are performed all in parallel resulting in several rows of weighted-sum outputs. These rows are then post-processed with a Winner-Take-All [41] or other compression circuit, or possibly just digitally encoded and sent to the next stage. The ability to expand or reduce the amount of computation is evident in this diagram.

as matrix-vector multiplication or adaptation. This powerful parallel computation is provided with the same circuit complexity and power dissipation as the memory that is needed to just store the coefficient digitally at 4-bit accuracy in non-ACA signal processing systems. This technology can be integrated in a standard digital CMOS process [46] where only on poly layer is available or in standard double-poly CMOS processes, both available from MOSIS and tested with this technology.

The core computational array is surrounded by pre- and post- processing blocks. Preprocessing blocks operate on the incoming signal by breaking the signal down into channels or some other orthogonal unit of information and then performing additional operations on each channel. Pre-processing blocks are also constructed like the core block in a fashion that permits them to be arrayed and also enabled-disabled post fabrication, similar to the core computation array. The pre-processing blocks then send the channels into the core computation array. Post-processing blocks perform functions such as winner-take all [41] as in the vector quantizer system [30] or, may simply convert the processed signal into a digital form for the next stage of a larger system, such as further processing in a DSP. Using the computation array together with pre- and post-processing blocks allows the implementation of scalable, programable, real-time computationally-intensive low-powered systems.

1.4 ACA History

This work began in 1998 when the design for a programmable filter was discussed and later fabricated in 1999 [37]. This system design became possible after the development and test of the Capacitivly Coupled Current Conveyor (C^4) [26] and programability of the floating-gate multiplier [37]. This filter provides a way to weight frequency bands of an arbitrary signal or can be programmed to extract a single channel out of the signal. This was the first ACA system to exploited the benefits of large array of floating-gate elements for use in analog computation. After it's construction the need for a automated system to rapidly and accurately program large arrays of floating-gate elements for the system to be useful became evident. The first attempts to provided a methodology to rapidly and accurately program these arrays started in 1999 and was used to program four coefficients in a recurrent connection network [47]. At first, these four elements were hand-programmed using knobs on a voltage meter. Shortly thereafter work began on a computer-controlled physically-based algorithm that used test equipment controlled via GPIB to program the coefficient arrays [53]. Array sizes in fabricated systems quickly grew to around 64 elements and following the control system was developed on a wire-wrap board that contained the DACs, level-shifters, and current measurement circuitry controlled by a PIC that communicated with MATLAB via a serial cable. Code was written on the PIC to provide the serial interface, set the voltages on the SPI controlled Maxium DAC chips, and provide the accurately timed programming pulses. A PCB version of this board was soon designed and

20 boards built that became the standard for programming floating-gate elements. This not only provided a self-contained setup but allowed for more accurate and rapid delivery of the necessary programming pulses used to program elements in the system than could be currently obtained with the bench setup. Using this PCB we have been able to program computing arrays using more than 2,000 coefficient elements. This board was used up until a redesign by another colleague that used an FPGA development board running a NIOS processor core instead of a PIC controller once these boards became available to the ICE group. This permitted more of the computation and calculations to be performed closer to the chip, an therefore quicker as the need for MATLAB performing the computations was reduced.

Analog Computing Arrays are quickly becoming a viable solution to many computationally intensive problems. It is believed that high density analog computing arrays will be an important option for designers who want to implement advanced signal processing algorithms for embedded and ultra low-power systems. Since initial attempts, ACAs have gone through several revisions, at each increasing the manageable number of computing elements which, translates to available computing power. These systems have now evolved over the last several years to the state where industry is becoming excited by their prospect. The ability to provide non-traditional analog computing in a low-power reconfigurable design has lead to the formation of a company by two other graduate students, my advisor and myself, that has resulted in series-A funding from a top venture capitalist.

Presented is the attempt to move the entire control system on-chip providing a single-chip ACA system, permitting system designs with over one million programmable elements that could be programmed in one second. We seek to be able to program this large number of elements in a relatively short period of time (1-5 seconds) while obtaining the accuracy needed for the overall system. Speeds in this order must be obtained if these computation arrays will be used in large scale production, such as a commercialized product. Further the desire is to move all controlling structures on-chip, include the need to charge-pumps to provide injection and tunneling voltages. The ultimate goal is a fully integrated platform for using the ACA concepts without the need to understand floating-gate programming physics.

CHAPTER 2

PROGRAMMABLE FILTER

2.1 Filter Concept

A tremendous amount of the initial research in developing this ACA scheme has gone into the space of programmable and adaptive analog filters based upon computing arrays. After the development of the C^4 an it's small size, the idea to tile several of these filters on one chip was envisioned. Figure 4 graphically demonstrates the top-level description of the Programmable Analog Filter. This figure shows four band taps that can be expanded to as many as needed. Each tap consists of a programmable bandpass filter and a weighted multiplier. The input signal is taken as a voltage, allowing the signal to be broadcast to the multiple bandpass filters or band tap columns. The multiple bandpass filters produce a frequency decomposition of the incoming signal into their programmed bands. A transistor-only circuit model of the AutoZeroing Floating-Gate Amplifier(AFGA) [18], which is referred to as the C^4(Capacitively Coupled Current Conveyer) [38, 37, 39], is used to achieve a broadly tuned bandpass response. By adding feedback between the stages, the filter's roll-off response can be sharpened if desired. Also, the filter can be cascaded into a multi-ordered filter increasing the role-off. The output of each bandpass filter is a voltage that is then broadcast to several weighted multiplier arrays. Therefore, with one input signal, multiple band-weighted versions of the original signal are available as outputs. The output of each weighted multiplier is a current allowing simple addition via KCL to assemble the final output signal from each band-weighted product. This results in a computation similar to a DSPs Fast Fourier Transform (FFT), band-weighting, and Inverse FFT.

The weighting in this filter is performed using floating-gate transistors [16] in a multiplier configuration. The benefits of using a floating gate for the weighting are small size and circuit simplicity. Also, it is inherently non-volatile. Hence, this computational memory element retains its value even when power is not applied to the device, and eliminates the

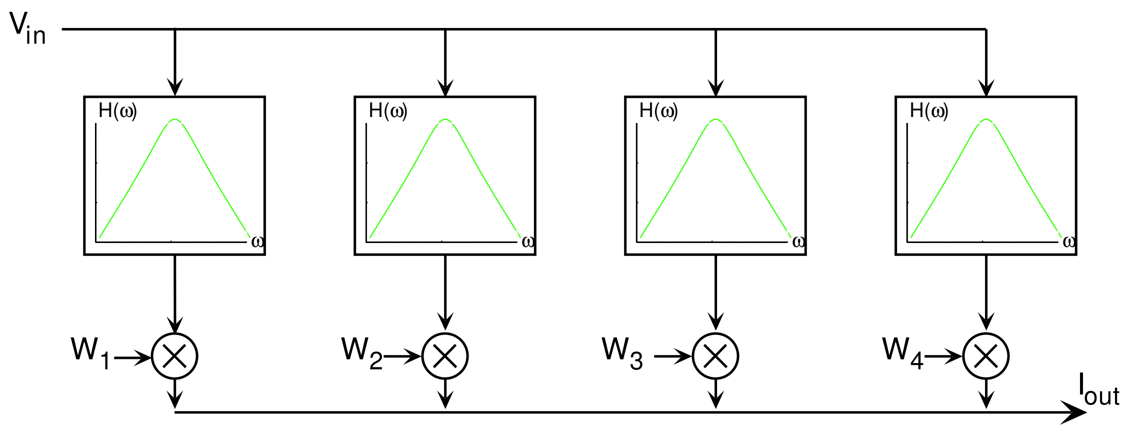


Figure 4. Top level representation of the programmable analog filter. The incoming signal is separated into frequency bands not by computing a DFT algorithm, but by a series of bandpass filters. With this topology it is easy to divide the frequency spacing exponentially, instead of linearly as in typical DFT algorithm.

need for separate non-volatile memory cells with Analog-to-Digital and Digital-to-Analog circuitry to store and reproduce the actual analog weight. Also, because the actual memory element is being used as part of the computation it allows for extremely high chip density which is desired to realize chips with large number of band taps or chips with several band weighted outputs. Further, this system only needs to operate at the incoming data speed and not 2 times or more the signal frequency to avoid aliasing.

First, the frequency response of a single C^4 bandpass element was examined. Demonstrated in Figure 5 is this programmable filter's operation. This bandpass response was obtained by adjusting the bias voltages, V_τ and $V_{\tau p}$, such that the high-pass and low-pass corner frequencies are nearly equal. This response shows that gain can be obtained through this system while filter taps overlap and the sum after multiplication adds multiple copies of a frequency.

All the filter's frequency taps were set to identical bias voltages, and the floating-gate weights were programmed to generate an interesting bandpass response. With this topology and constant bias voltages, many arbitrary filters of second order can be programmed; using different topologies and using feedback connections will allow any desired filter function. A 15% difference in corner frequencies was found in the experiment due to mismatch in the

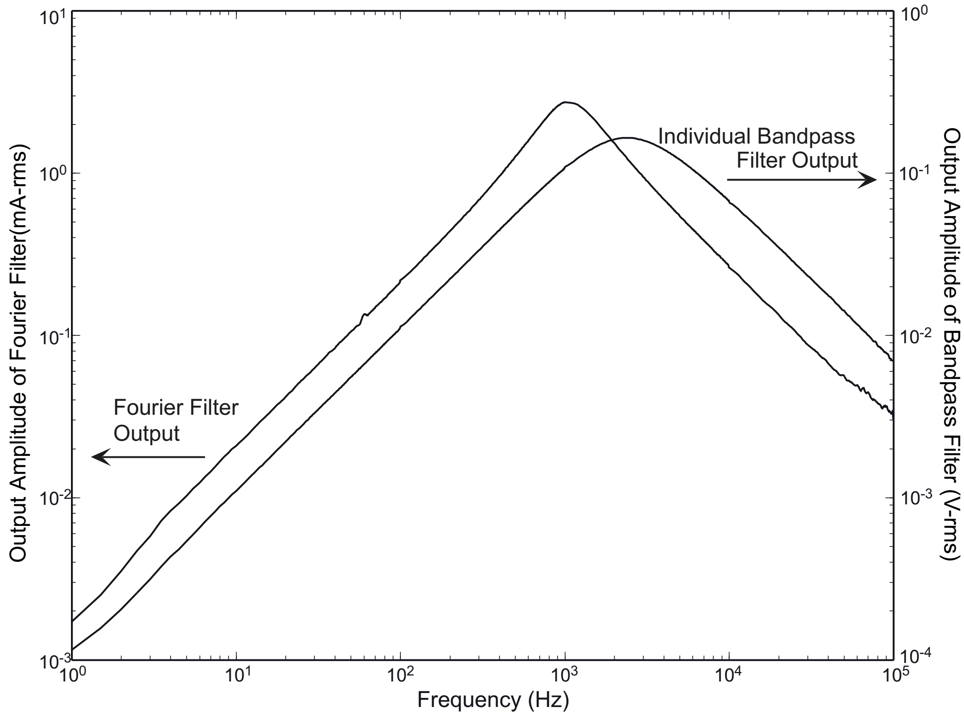


Figure 5. Frequency response of programmable bandpass filter. Frequency response for a single bandpass filter, and for the array bandpass filters with programmed weighting function. The Poly tilt frequency line set with zero difference ; therefore all corner frequencies should be identical aside from mismatch. The result of this programmable filter is a tighter bandpass filter, with a corner frequency roughly half of the original corner frequency.

bias transistors. This current chip has a linear change in bias voltages, because neighboring biases were connected with resistive poly connections. The weights were programmed to the following pattern:

Table 1. Normalized weights for filter shown in Figure 5

Position	1	2	3	4	5	6	7	8	9	10
$W^{+}(\mu A)$	1.56	0.85	1.23	2.38	0.59	1.55	1.06	0.62	2.39	0.90
$W^{-}(\mu A)$	1.11	0.99	1.24	0.73	1.45	1.11	2.08	0.60	0.64	0.75
$W(\mu A)$	0.45	-0.14	-0.01	1.65	-0.86	0.44	-1.02	0.02	1.75	0.15

The floating-gate values did not change throughout the experiment. Typically, a small initial change is noted the first 24 hours due to detrapping (approximately an identical 10mV for each device), after which the gate charge remains constant over the entire duration of these experiments. Since differential weights are used any constant detrapping will

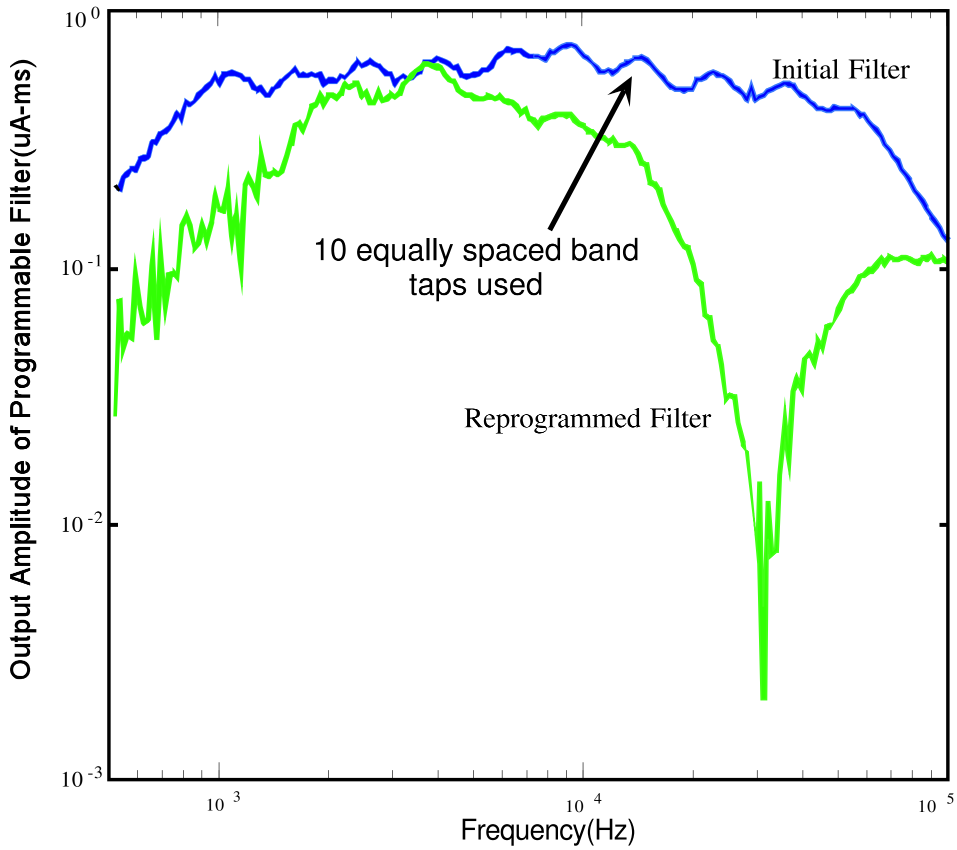


Figure 6. Frequency response of programmable filter. This filter has 10 bandpass elements exponentially spaced in frequency. The ripples on both curves show the location of these bandpass elements. Shown is an initial programmed frequency response, where the weights are nearly equal, and a second programmed frequency response to program an additional notch in the filter's response.

not affect the filter's transfer function.

Figure 5 shows experimental measurements from our programmable Fourier filter. Shown is the frequency response of individual bands multiplied by their weighted outputs for constant effective weights. The result of this programmable filter is a tighter bandpass filter, with a corner frequency roughly half of the original corner frequency. The floating-gate devices used for the multiplication were built with W/L = 10; therefore the devices were biases near threshold with an average bias current of 1μA (total current was 20.58μA). It was found that dynamics of the multipliers did not affect the filter transfer function, and that the harmonic distortion is limited by the multipliers, and not the bandpass filters.

Figure 6 shows the frequency response for a programmable filter with 10 bandpass elements that are exponentially spaced in frequency, instead of to nearly identical frequencies. For the initial frequency response an exponential spacing in the bands with nearly identical weight values at each tap where used. The exponential spacing of these bandpass elements is evident from the ripples on the initial frequency response. Also shown is a second programmed frequency response, where an additional notch is programmed in the spectrum of this initial filter simply by adjusting the weightings of each band. In newer versions the corner frequency parameters of each bandpass element is programmable is set with floating-gate as a bias element. Therefore, we can build programmable filters utilizing arbitrary spacing between each bandpass filter tap.

The combination of the Analog computing array with the C^4 used as a pre-processing block, resulted in a fully programmable analog filter. This filter concept has been advanced well beyond what is presented here by several colleges in the ICE lab. They have taken this initial work and a fully characterized these programmable filters for parameters like speed, SNR, and obtainable Q peaks. This FFT - multiple - IFFT like operation in an analog system has many applications such as Adaptive channel equalization (ACEQ). An ACEQ system has been designed and fabricated as a collaborative project by two other colleagues and myself takes this filter concept one step further by permitting the weight in the multiplication to adapt to the correct value without explicitly programming them.

2.2 Capacitively Coupled Current Conveyor (C^4)

The Capacitively Coupled Current Conveyor (C^4) is a transistor-only version of the autozeroing floating-gate amplifier (AFGA) [15, 28]. A subthreshold transistor is used to model the behavior of an electron-tunneling device, and another subthreshold transistor is used to model the behavior of pFET hot-electron injection in the AFGA. Analytical models have been derived that characterize the amplifier and that are in good agreement with experimental data. This circuit operates as a bandpass filter, and behaves similarly to the AFGA

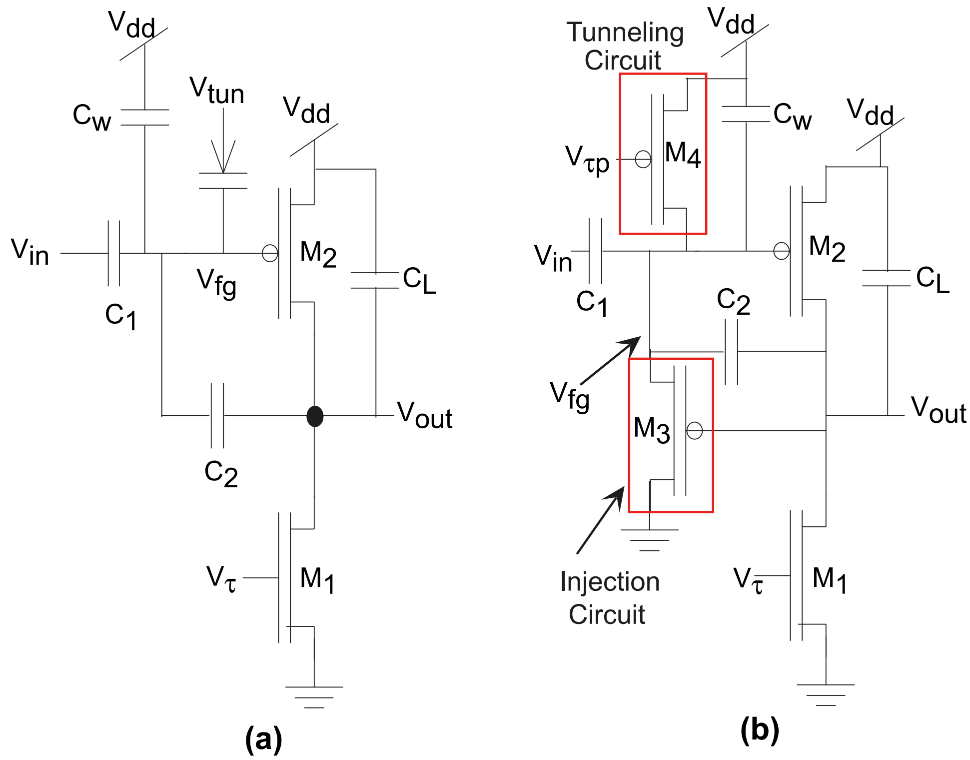


Figure 7. The ratio of C_2 to C_1 sets the gain of both inverting amplifiers. The capacitances, C_w and C_L, represent both the parasitic and the explicitly drawn capacitances. (a) An autozeroing floating-gate amplifier (AFGA) that uses pFET hot-electron injection. The nFET is a current source, and it sets the current through the pFET. Steady state occurs when the injection current is equal to the tunneling current. Between V_{tun} and V_{fg} is the symbol for a tunneling junction, which is a capacitor between the floating-gate and an *n*well. (b) The all-transistor circuit version of the AFGA. M4 represents the tunneling junction in the AFGA, and M3 represents the injection current (gate current) from M2 in the AFGA (Fig. 7a).

but with different operating parameters. Both the low-frequency and high-frequency cut-offs are controlled electronically, as is done in continuous-time filters. The C^4 circuit has a low-frequency cutoff at frequencies above 1Hz due to the minimum current limitations of a MOSFET transistor. This curcuit provides a compliment to the operating regimes of the AFGA that using tunneling currents has a low frequency corner in the range of much less than 1Hz to about 100Hz.

Figure 7a shows the Autozeroing Floating-Gate Amplifier (AFGA). The AFGA uses complementary tunneling and pFET hot-electron injection to adaptively set its DC operating point. The tunneling and hot-electron injection processes adjust the floating-gate charge such that the amplifier's output voltage returns to a steady-state value on a slow time scale. The modulation of the pFET hot-electron injection by the output voltage provides the correct feedback to return the output voltage to the proper operating regime. It can achieve a high-pass characteristic at frequencies well below 1Hz. Figure 7b shows an all-transistor model of this circuit. The investigation of the C^4 circuit was initially undertaken as a way to give an intuitive circuit model and permit circuit simulation of AFGA operation, but has important circuit applications itself. This circuit on it's own is useful for applications requiring adaptation or low corner frequencies above 10Hz, such as audio-band or even IF processing.

To clarify the AFGA's behavior, this circuit was constructed using transistor elements that have identical behavior to the tunneling and injection processes. The open-loop amplifier, which consists of a pFET input transistor, **M2**, and an nFET current source, **M1**, does not change in this all transistor version. The pFET current source, **M4**, serves the same function as the tunneling junction; the tunneling junction was used as a constant current source able to supply extremely small currents, and is a fairly good model in several applications. The second pFET models the hot-electron injection current dependence; for pFET hot-electron injection, the injection current increases for decreasing drain voltage and gate voltage. The **M3** pFET models the drain effect with some difference in parameters; the gate dynamics are similar to the source-degenerated pFETs [15], but it has little effect in either circuit in Figure 7. The two transistor configuration in Fig. 7, **M2** and **M3**, is found in several current mode circuits, but it appears this voltage mode circuit has not been previously considered.

The circuit in Fig 7b and the resulting circuit family of capacitor circuits based on the AFGA have several circuit applications. The first application is in circuits where the adaptation rate needs to be faster than can easily be achieved by tunneling / injection currents.

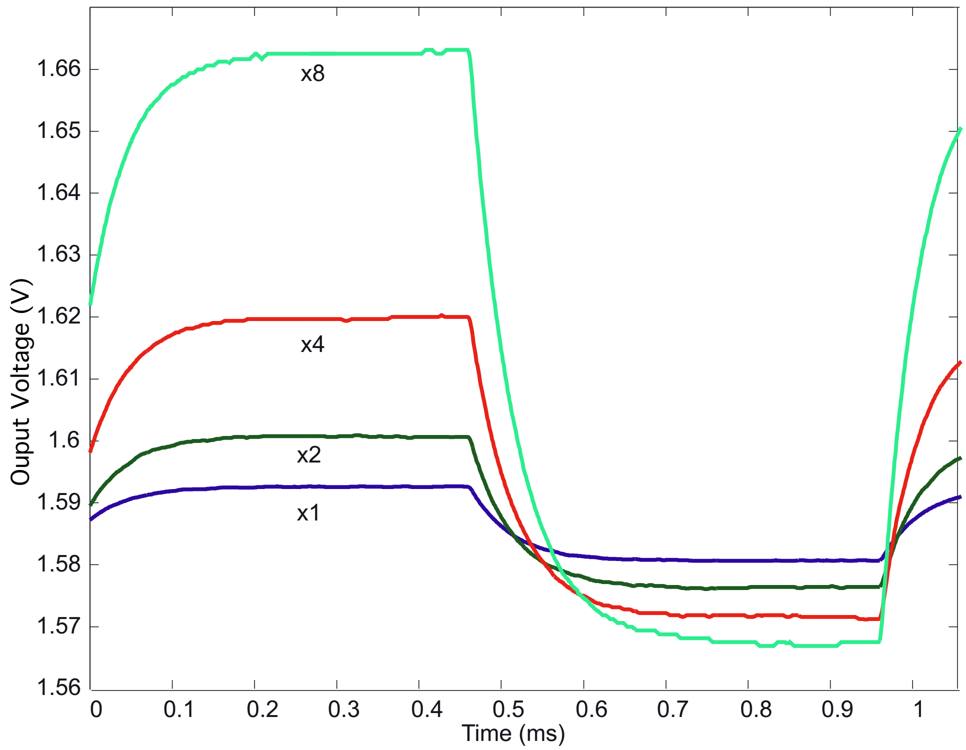


Figure 8. Measured time-domain behavior for the circuit shown in Fig. 7b. Measured output voltage for a step input at four different amplitudes for a fast time response.

The low-frequency cutoff of this circuit is at frequencies above 1Hz, and therefore complements the operating regimes of AFGAs. The second application is in chips requiring very low power supplies, particularly in battery applications. AFGA circuits require higher supplies than the process rating, primarily because the hot-electron effects that are used for the AFGA are in fact what sets the process limits. The experimental measurements were taken at 3.0V, except for dynamic range measurements which were taken with a 1.5V supply. Presently the circuit in Fig. 7 is being used in coupled arrays of bandpass filters, high-speed adaptation circuits, and in circuit models of neural computation.

The following section will discuss the all-transistor circuit model of the AFGA. The relevant equations will be derived, many of which are similar to AFGA equations [15] with different parameters. The following sections consider the time-domain, frequency domain, linear range, noise, and dynamic range of this circuit. Measured data for this circuit that was fabricated in a 1.2μm *n*well process is presented. In addition, a couple of effects not

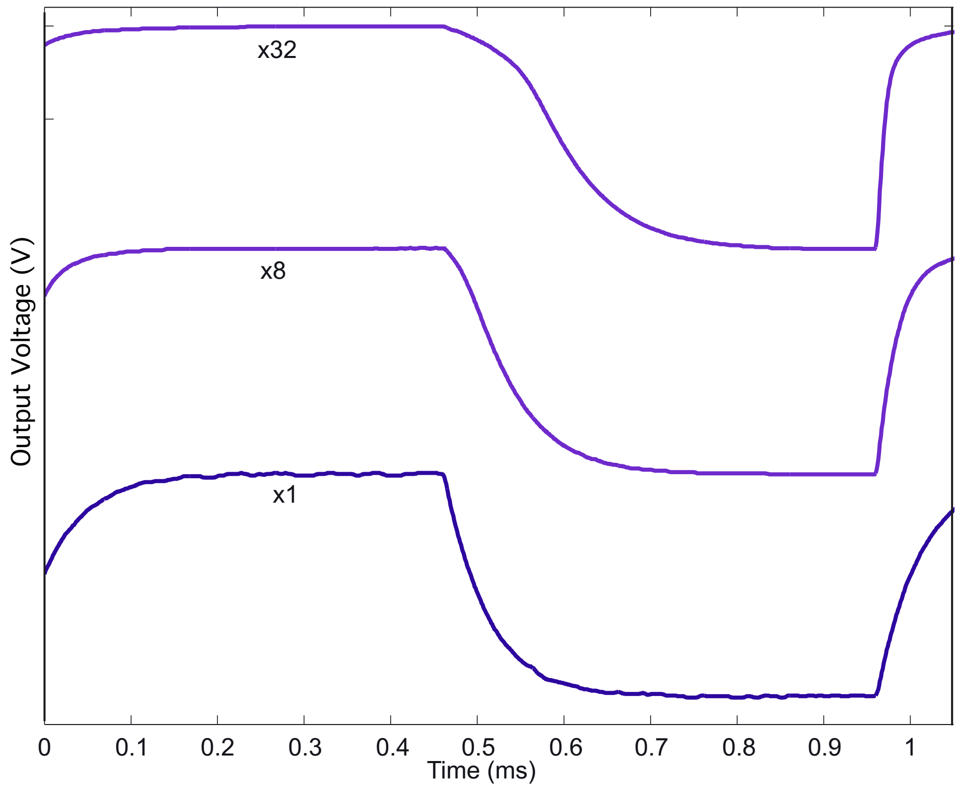


Figure 9. The curves are normalized from Fig. 8 to emphasize the change in shape as we apply larger input amplitudes. The smallest amplitude input step results in nearly symmetric output response, but the larger amplitude inputs result in asymmetric responses due to the second-order nonlinearities. Step input at even a larger amplitude showing clearly showing the distortion at high frequencies.

seen in the AFGA because of the different parameters to this circuit will be considered.

2.3 Basic Circuit Equations

To model the AFGA or the all-transistor equivalent circuit, two equations are written governing the autozeroing floating-gate amplifier behavior around an equilibrium output voltage. For subthreshold operation, the change is described in the nFET or pFET channel current for a change in gate voltage, ΔV_g, and source voltage, ΔV_s, around a bias current, I_{so}, as [15].

$$\text{nFET} : I_n = I_{so} \exp\left(\frac{\kappa_n \Delta V_g - \Delta V_s}{U_T}\right),$$

$$\text{pFET} : I_p = I_{so} \exp\left(\frac{-\kappa \Delta V_g + \Delta V_s}{U_T}\right), \tag{1}$$

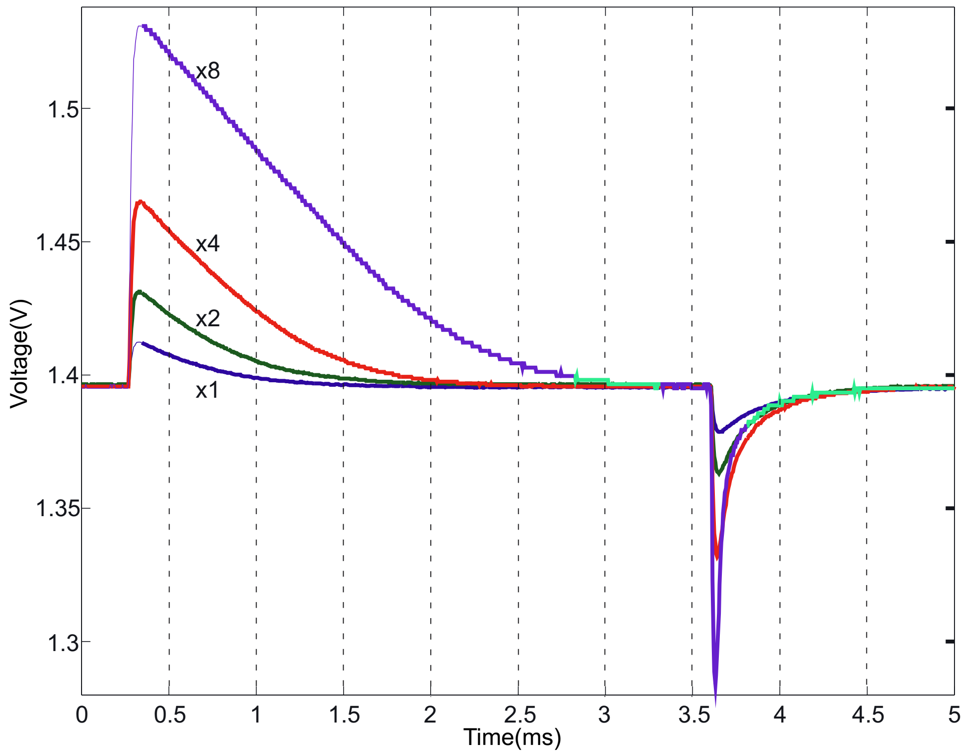


Figure 10. Measured time-domain behavior for the circuit in Fig. 7. Measured output voltage for a step input at four different amplitudes for a slow time response. Measured output voltage for a low-frequency step input at four different amplitudes. The large signal asymmetry of this circuit is visible in this plot. This is a result of the constant current source providing a linear recovery seen in the up-going step and the pFET transistor-feedback providing an exponential recovery seen in the down-going step.

where κ_n, κ is the fractional change in the nFET, pFET surface potential due to a change in ΔV_g, and U_T is the thermal voltage, $\frac{kT}{q}$. We obtain the first equation by applying Kirchoff's current law (KCL) at the floating gate, V_{fg}:

$$C_T \frac{dV_{fg}}{dt} = C_1 \frac{dV_{in}}{dt} + C_2 \frac{dV_{out}}{dt}$$

$$+I_L \left(1 - e^{-(\kappa \Delta V_{out})/U_T}\right) \tag{2}$$

where C_T is the total capacitance connected to the floating gate ($C_T = C_1 + C_2 + C_w$), and I_L is the bias current set by **M4** that sets the adaptation rate. The second equation is obtained by applying KCL at the output node and assuming sufficiently large open-loop gain:

$$C_o \frac{dV_{out}}{dt} = C_2 \frac{dV_{fg}}{dt} + I_H \left(e^{-\kappa \Delta V_{fg}/U_T} - 1\right), \tag{3}$$

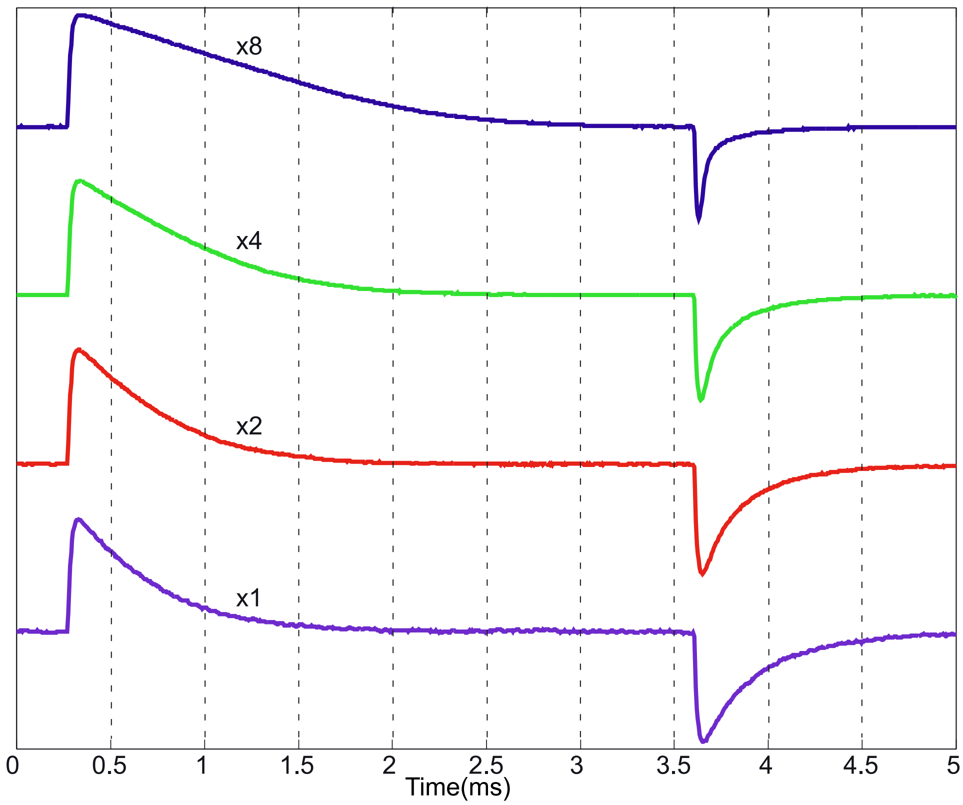


Figure 11. The curves are equal amplitude to see change in shape as we apply larger input amplitudes on the low frequency corner. In both the high frequency and low frequency step, the smallest amplitude input step results in nearly symmetric output response, but the larger amplitude inputs result in asymmetric responses due to the second-order nonlinearities.

where C_o is the total capacitance connected to the output node ($C_o = C_2 + C_L$), and I_H is the bias current set by **M1** that sets the low-pass filter rate. The voltages on the gates of **M1** and **M4**, V_τ and $V_{\tau p}$, set the two bias currents, I_H and I_L, respectively. These equations are identical to the AFGA case, but typically the equivalent term to $\exp(V_g/U_T)$ in (2) is neglected because of the large difference in timeconstants for the AFGA.

2.4 Time-Domain Modeling

For short-timescale dynamics, I_L is negligible; this assumption is equivalent to ignoring the tunneling and injection currents for the AFGA. Combining (2) and (3) with this simplification results in the following equation for ΔV_{fg}:

$$\tau_h \frac{dV_{fg}}{dt} = A_h \tau_h \frac{dV_{in}}{dt} + \left(e^{-\kappa \Delta V_{fg}/U_T} - 1\right), \qquad (4)$$

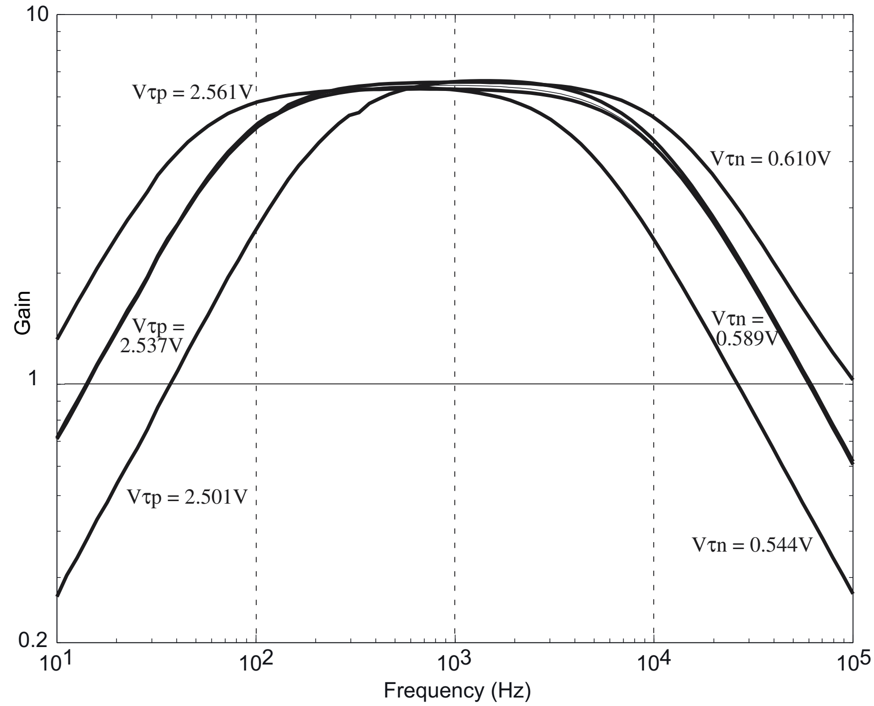


Figure 12. Frequency Response for the circuit in Fig. 7b. (a) Gain response for three different values of V_τ and three different values of $V_{\tau p}$. We can independently change the high-frequency corner with V_τ and can independently change the low-frequency corner with $V_{\tau p}$. The passband gain of this circuit is roughly 6.4.

where we define $\tau_h = \left((C_T C_o - C_2^2)/U_T\right)/(\kappa C_2 I_\tau)$, and $A_h = C_1/\left(C_T - (C_2^2/C_T)\right)$. The gain from input to output due to capacitive feedthrough is A_h. This equation and its solutions are similar to AFGA solutions [20, 23]. As in the AFGA, increasing C_w or C_L without changing C_1 and C_2 decreases the amount of capacitive feedthrough.

For long-timescale dynamics, the floating-gate voltage is fixed by the high gain amplifier formed by **M1** and **M2**. Therefore, (2) simplifies to

$$C_2 \frac{dV_{out}}{dt} = -C_1 \frac{dV_{in}}{dt} + I_L \left(e^{-(\kappa \Delta V_{out} - \Delta V_{fg})/U_T} - 1\right) \tag{5}$$

If an input voltage step is applied such that ΔV_{out} moves to $\Delta V_{out}(0^+)$, then how ΔV_{out} varies with time (t) can be modeled immediately following this step:

$$\Delta V_{out}(t) = V_{inj} \ln\left(1 + \left(e^{\Delta V_{out}(0^+)/V_{inj}} - 1\right) e^{-t/\tau_l}\right), \tag{6}$$

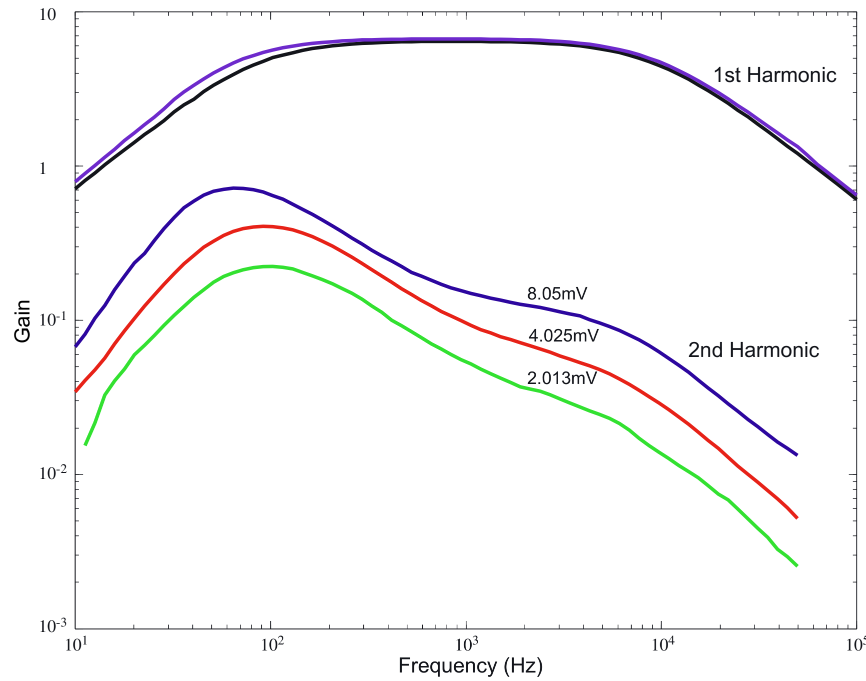


Figure 13. Gain response of the first and second harmonics for three different input levels and V_τ = 0.589V and $V_{\tau p}$ = 2.537V. Doubling the amplitude results in gain doubling, as expected for second harmonic distortion. The distortion peak is near the lower frequency corner. Linearizing the low frequency corner should be considered to reduce distortion.

where $\tau_l = C_2 V_{inj} / I_L$; note that $\Delta V_{out} \rightarrow 0$ as $t \rightarrow \infty$. A similar approximation is obtained for fast inputs, which is similar to expressions for the AFGA [23].

Figure 8 and Fig. 10 illustrate the short- and long-timescale behavior by showing the output-voltage response to a step input. Figure 11 shows that output voltage returns to steady state. As amplitude increases, the waveform becomes more asymmetric and nonlinear. This trend follows (6) and typical AFGA behaviors [23]. Figure 9 shows this circuit's measured output-voltage response to square-wave inputs. As in the low-frequency case, the high-frequency response of the AFGA is asymmetric: the downgoing step response approaches its steady state linearly with time, and the upgoing step response approaches its steady state logarithmically with time.

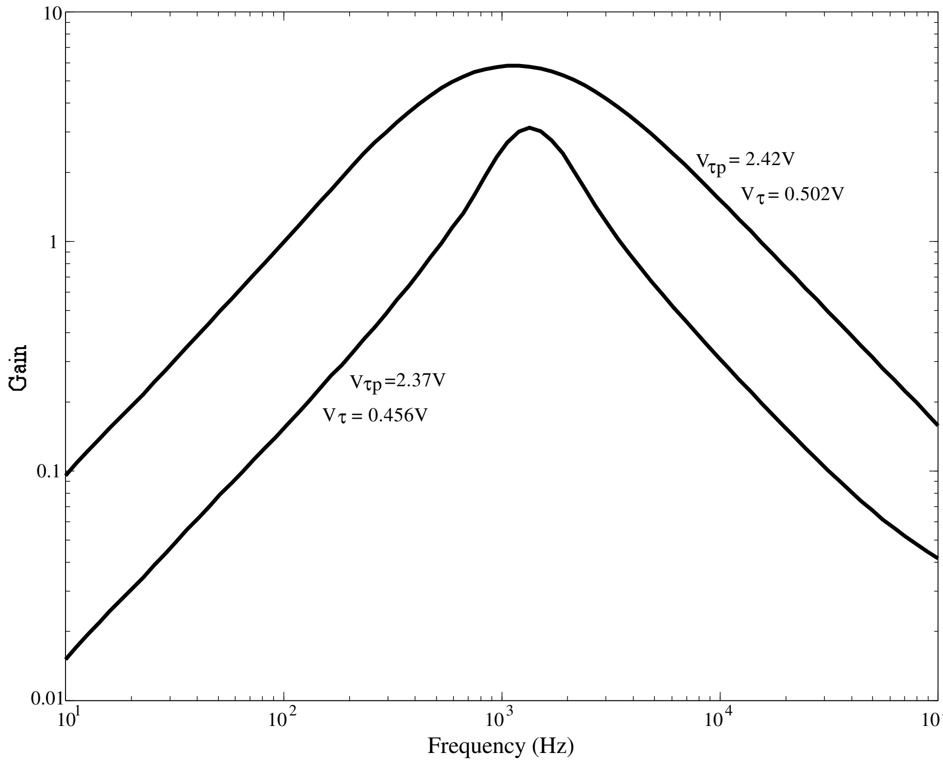


Figure 14. Frequency response of the Bandpass circuit when τ_l is near τ_h. When symmetrically decreasing τ_l and τ_h, the center frequency remains nearly unchanged, but the bandpass gain decreases.

2.5 Frequency Response

Like the AFGA, this circuit's transfer function is bandpass, with the low-frequency cutoff set by the equilibrium currents from **M3** and **M4**, and the high-pass cutoff independently set by the equilibrium pFET, **M2**, and nFET, **M1**, channel currents. Figure 12 shows the measured AFGA frequency response for various inputs and V_τ, $V_{\tau p}$ bias voltages. By taking the Laplace transform of (2) and (3), the following transfer function can be obtained:

$$\frac{V_{out}(s)}{V_{in}(s)} = -\frac{C_1}{C_2}\frac{1 - A_h\tau_h s}{1 + \tau_h s + \frac{1}{\tau_l s}}, \tag{7}$$

where τ_l, τ_h, A_h are as defined previously.

To simplify (7) when $\tau_l \gg \tau_h$, the time constants are sufficiently separated to form an amplifier region. In this regime, V_τ and $V_{\tau p}$ independently alter the corner frequencies; this

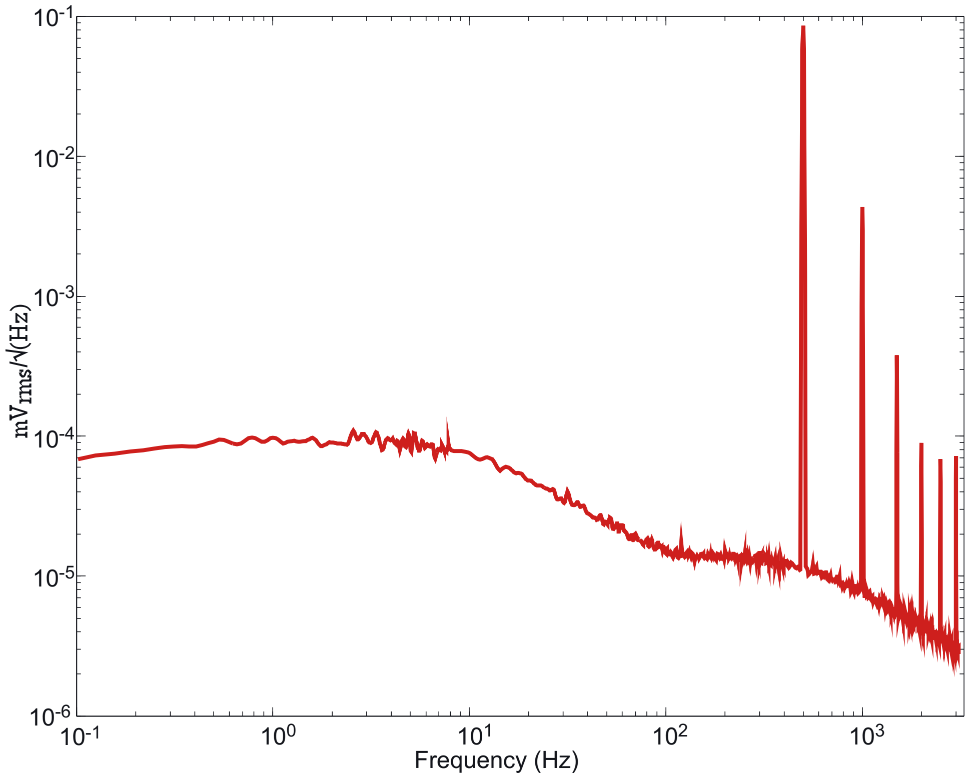


Figure 15. Spectrum of the C^4 voltage for a sinusoidal input. The total harmonic distortion was -26dB below the fundamental frequency, and is primarily dominated by second harmonic distortion; this location was the location of maximum second-harmonic distortion for the high-frequency corner. Like the AFGA, -26dB corresponds to the linear range of this amplifier. The total output noise coming from this amplifier was 0.41mVrms. From this graph, the resulting dynamic range is 47dB. The circuit was operating on a 1.5V supply for this experiment.

region of operation closely matches the AFGA frequency response [23]. At low frequencies, the circuit in Fig. 7 behaves as a high-pass filter. Approximating (7) as

$$\frac{V_{out}(s)}{V_{in}(s)} = -\frac{C_1}{C_2}\frac{s\tau_l}{1 + s\tau_l}. \tag{8}$$

The corner frequency is set by $V_{\tau p}$. At high frequencies, the circuit in Fig. 7 behaves as a low-pass filter. In this case, approximating (7) as

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2}\frac{1 - A_h\tau_h s}{1 + \tau_h s}. \tag{9}$$

The nFET current source, **M1** sets the bias current and sets the resulting corner frequency. At frequencies much higher than the integrating regime, this circuit exhibits capacitive feedthrough (not seen in Fig. 12, but seen at high frequencies in Fig. 14 for the low gain case) which can be reduced by an increase in either C_w or C_L.

The circuit in Fig. 7 can also operate as a bandpass filter with a narrow passband, that is, V_τ and $V_{\tau p}$ can effect the entire transfer function. In an AFGA the corners are typically very far apart. Figure 14 shows the frequency response for two values τ_l and τ_h that are close together; this experiment shows this bandpass behavior.

It would be useful to calculate this circuit's dynamic range and compare it to the AFGA. Figure 15 shows the spectrum of the circuit's output voltage for a sinusoidal input. The total harmonic distortion was -26dB below the fundamental frequency, and is primarily dominated by second harmonic distortion; this location was the location of maximum second-harmonic distortion for the high-frequency corner. Like the AFGA, -26dB corresponds to the linear range of this amplifier. Assuming thermal noise, the total output-noise power is identical to the AFGA [23]:

$$\hat{V}_{out}^2 = \left(\frac{C_T}{C_2 g_m}\right)^2 \int_0^{\frac{2\pi}{\tau_h}} \frac{\hat{i}_o^2}{1 + (\omega\tau_h)^2} \frac{df}{\Delta f} = \frac{qU_T}{\kappa} \frac{C_T}{C_2 C_o}. \tag{10}$$

The total output noise coming from this amplifier was 0.41mVrms. The total output-noise power is roughly proportional to C_w, and is inversely proportional to C_L. We define dynamic range, DR, as the ratio of the maximum possible linear output swing to the total output-noise power. In the case where the high-frequency cutoff sets the distortion, the dynamic range is also identical to the AFGA case [23]:

$$DR = \frac{V_{Lo}^2}{2\hat{V}_{out}^2} = \frac{\kappa}{2q} V_{Lo} C_o. \tag{11}$$

From this graph, the resulting dynamic range is 47dB. As in the AFGA circuit, the linear range can be increased by increasing C_w, and the dynamic range can be increased by increasing C_w and C_L [23, 15]. These circuits have been simulated using Cadence [43].

CHAPTER 3

DIBL FOR EXTENDING LINEAR RANGE OF C^4

A Drain Induced Barrier Lowering (DIBL) device can be used to increase the linear range of the C^4 circuit. To see why this is, let us examine a common source nFET amplifier with a pFET load current source. The gain of this device can be written as

$$A_v = -\frac{\kappa(V_{o,nFET}||V_{o,pFET})}{U_T} \tag{12}$$

Assuming the nFET has a sufficiently large channel length we can ignore it's Early effect and conclude the gain A_v is simply $\kappa V_a/U_T$. If we want to trade off gain for increased linearity we would want to reduce the Early voltage or increase the early effect. This can be accomplished through the use of a DIBL device as explained.

3.1 The MOSFET Relationship of Channel Current to Drain Voltage

When a subthreshold MOSFET saturates (at approximately $V_{ds} > 4U_T$), we model the current through the channel as

$$I = I_o(e^{(\kappa V_g - V_s)/U_T})e^{V_{ds}/V_A} = I_{sat}e^{V_{ds}/V_A}, \tag{13}$$

where I_o is analogous to the reverse saturation current in a BJT, and I_{sat} is the ideal drain current in saturation, assuming no dependence on drain voltage. Fig. 16 shows that real devices can show a significant drain-voltage dependence. The effect of the drain voltage modulating the channel's length is known as channel-length modulation or the Early effect, where V_A (=1/λ) is the Early voltage. Although we will focus on MOSFETs, we could similarly express the collector current in a BJT as

$$I = I_o e^{(V_b - V_e)/U_T} e^{(V_c - V_e)/V_A}. \tag{14}$$

The lowest curve in both parts of Fig. 16 shows drain current versus drain voltage for a MOSFET with the minimum channel length allowed by the process' design rules. The

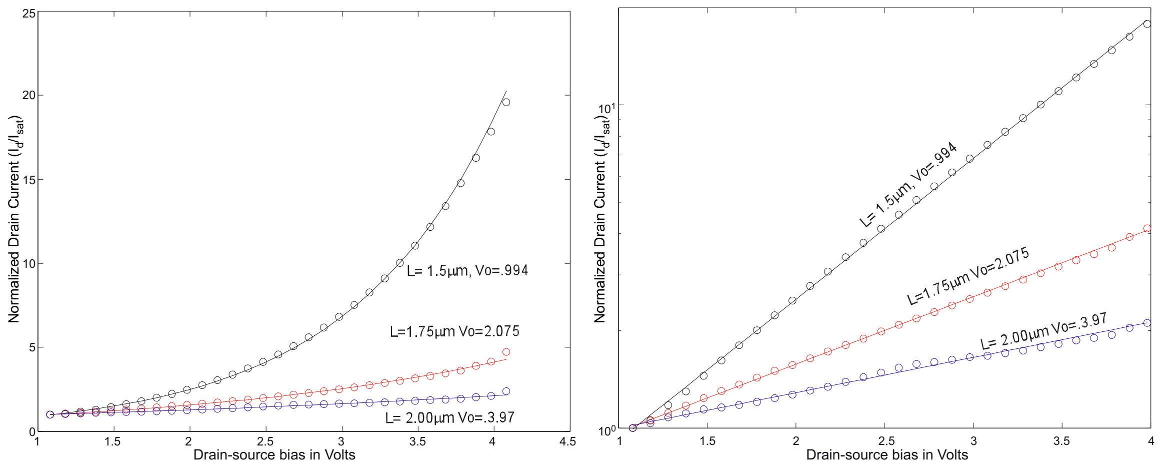


Figure 16. Empirical measurements of drain current versus drain voltage for three different channel-length nFETs in a 2.0 μm process at a gate voltage 0.518V above substrate. In the top plot, the linear scale shows that the drain current's dependence upon V_{ds} is strong and nonlinear for devices shorter than minimum length. On a logarithmic current scale (the bottom plot), straight lines validate an exponential representation of drain current versus drain voltage.

weak dependence of its channel current on drain voltage is adequately described by the classic model of the Early effect:

$$I = I_{sat}(1 + \frac{V_d}{V_A}) \Rightarrow \frac{\Delta I}{\Delta V_d} = \frac{I}{V_A}. \tag{15}$$

Unfortunately, this classic model does not adequately describe the drain voltage dependence when the channel length is reduced below the processes minimum stated channel lenght because the dependency of I_d upon V_{ds} is exponential. Therefore we forgo the classic model for the exponential term e^{V_{ds}/V_A} in (13). Now that we have supplied empirical verification of our model, let's examine its physical basis.

3.2 Drain-Induced Barrier Lowering (DIBL)

Fig. 17 shows the cross section of an n-channel MOSFET as well as its energy band diagram (potential versus length along the channel). For a short-channel device, a decrease in the drain's electron potential with respect to the channel (i.e. an increase in V_d) results in a significant drop in the work function between the source and the channel. Since the current density increases exponentially with a decreasing source-to-channel barrier, the

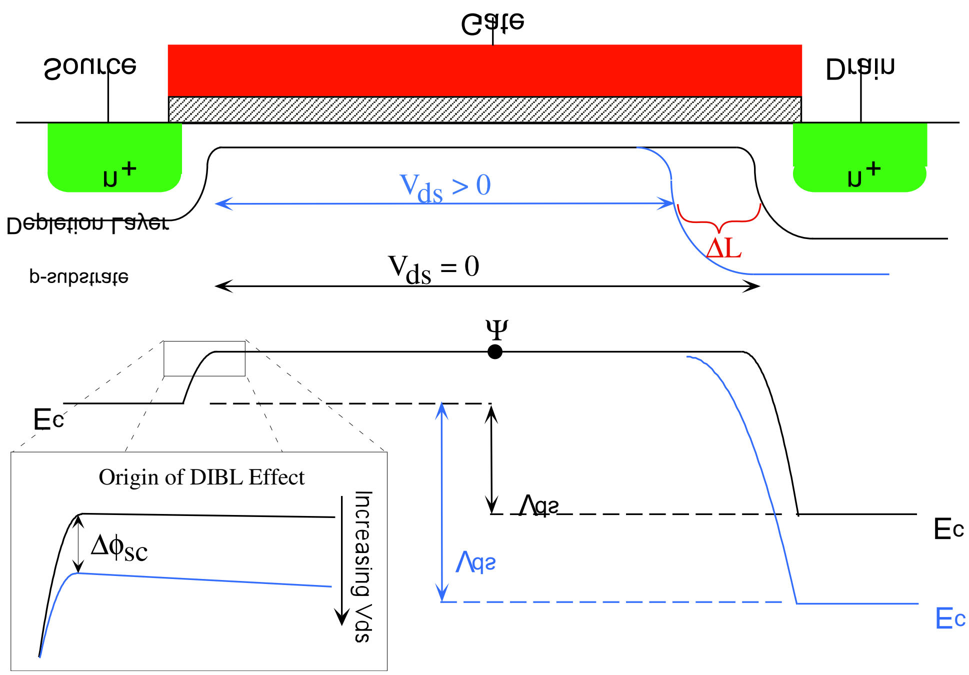


Figure 17. Cross section and energy band diagram of a MOSFET. As V_{ds} increases, the channel grows shorter; therefore, the current increases. Also, a drop in electron potential at the drain of an nFET induces a small drop in the electron potential of the channel at the source-channel interface. (The picture of the source barrier is enlarged.) The current density through the channel increases as an exponential function of the source barrier reduction.

channel current is an exponential function of the drain voltage, much as it is in the floating-gate transistor. Both processes include barrier lowering at the source due to movement of the drain (although they happen for different reasons). Therefore, we can use (13) to model this phenomena: V_A is a parameter that can be extracted empirically. For long MOSFETs, a large V_A is associated with an exponential curve whose curvature is very small. In fact, the bend in the curve is so small that the slope of $\Delta I/\Delta V_{ds}$ does not change noticeably for the operating range of the device; therefore, the exponential curve appears linear. In summary, the channel current's dependence on drain voltage looks linear for long-channel MOSFETs, while for MOSFETs with very short channels, the drain current varies exponentially with drain voltage in saturation. Nevertheless, we can use the same expression and a single parameter to model both short-channel circuit effects.

In Fig. 18 (b) we show empirical evidence of the short-channel MOSFET's exponential

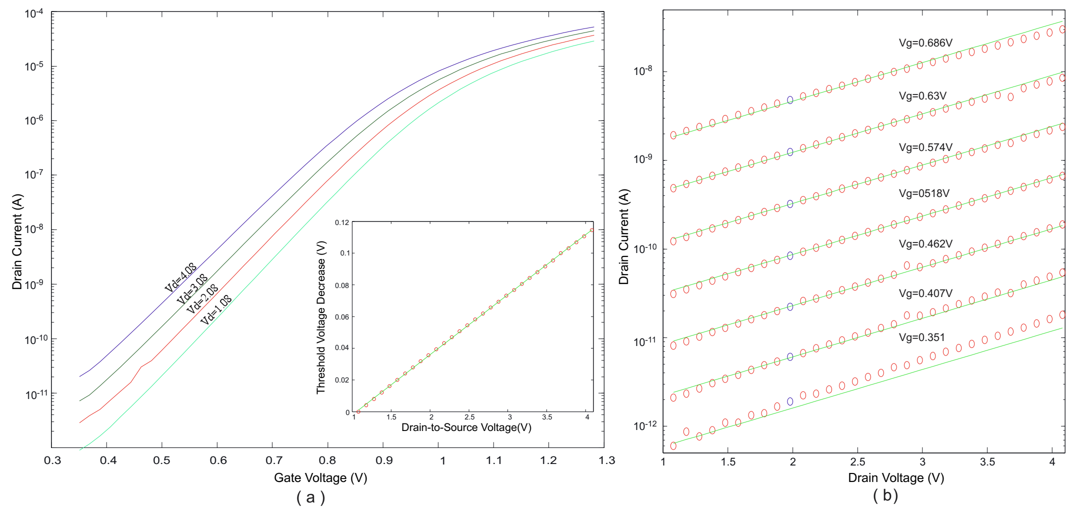


Figure 18. Measured data from a short-channel MOSFET showing substantial DIBL. (a) Linear curve fit to the threshold voltage shift with drain voltage: slope = 0.0379 V/V (b) Current versus drain voltage for several applied gate voltages. This data was taken from an L=1.75μm pFET built in a 2μm CMOS process. The DIBL FET results in an exponentially increasing current due to linear changes in drain voltages for subthreshold biases; a similar effect is seen for above-threshold currents. Good agreement is seen to a single curve fit of this data to the modified nFET equations, with $\kappa = 0.614$ and $V_A = 1V$. We can use this device to increase the linear range of on-chip FET circuits without resistors. Single curve fit to experimental data: $I = (5.18 \times 10^{-17}) \times e^{0.614 V_g/U_T} e^{V_d/1.0V}$

dependence of I_d upon V_{ds} over several distinct gate voltages, just as we did for the floating-gate dev Since both phenomena involve source barrier lowering, we should expect that DIBL also manifests itself in threshold voltage reduction. Fig. 18 (a) shows that applying a change in drain voltage noticeably shifts the curve of drain current versus *gate* voltage; in other words, the threshold voltage shifts. The curves are equally spaced for equal changes in drain voltage; hence, the threshold voltage drop is linear. Therefore, we can infer that moving the source barrier is reduced by a linear factor of the drain potential decrease. The inset plot accentuates the linear decrease in V_T with an increase in V_{ds}.

We have discussed how a decrease in channel length entails an increased drain voltage dependence. We show this effect explicitly in Fig. 19, where we have plotted Early voltage versus effective channel length for nFETs and pFETs in the same 0.5μm CMOS process. For this plot we have estimated the effective channel length to be 0.35μm less than the drawn gate length. Both curves show that current to drain voltage *in*dependence (i.e. Early

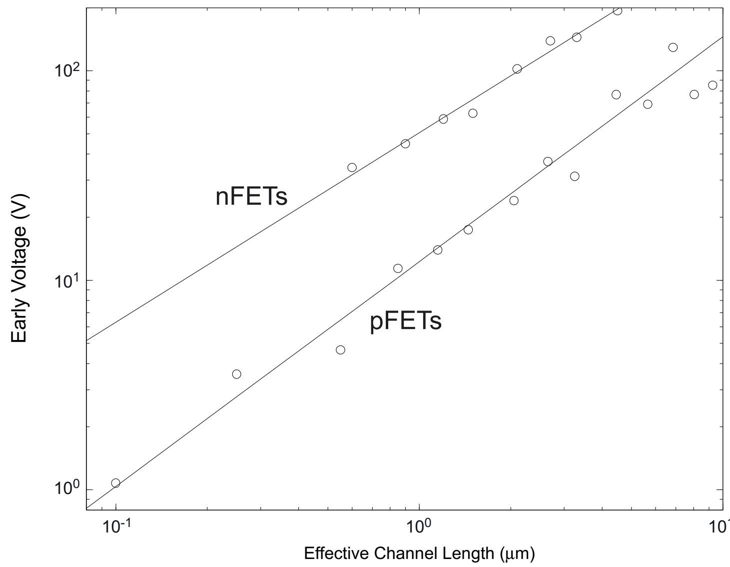


Figure 19. Measured dependence of Early voltage on effective channel length in a $0.5\mu m$ process for both nFETs and pFETs. The pFET data includes devices at and below the minimum allowed channel length for the processes.

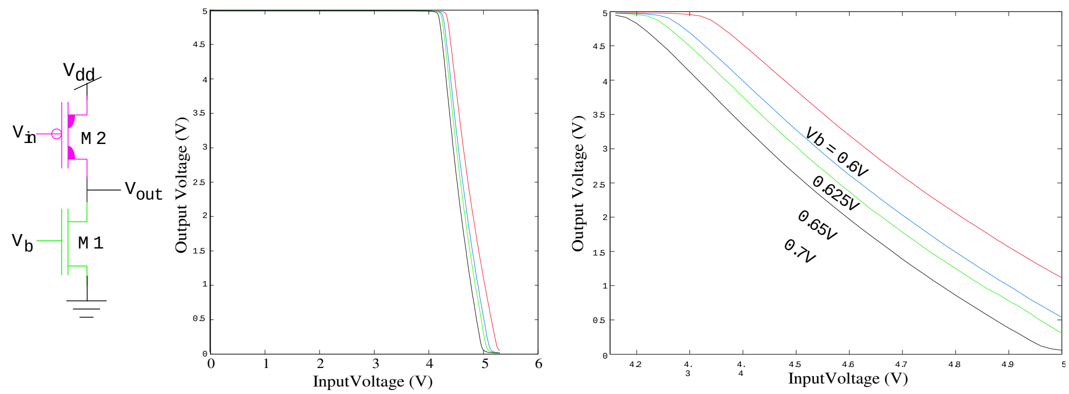
voltage) is a linear function of effective channel length. The difference in the slopes of the two curves signifies different doping profiles of the devices.

3.3 DIBL devices in amplifiers

We will now show how linear operation in the exponent provide a powerful technique to develop circuits with saturated MOSFETS operating in the subthreshold regime, or BJT circuits biased in the active regime. We can reforulate our earlier model, (13), as

$$I = I_o e^{(\kappa V_g - V_s)/U_T} e^{V_d/V_A} = I_{sat} e^{V_{ds}/V_A} \tag{16}$$

where $A_v = V_A/U_T$ is the maximum voltage gain of a subthreshold transistor. If we build a circuit where we fix the channel current, we would predict a linear dependence between V_g, V_d, and V_s, even though we start with a nonlinear equation, throughout the saturation operating region. For example, consider the high-gain amplifier circuit shown in Fig. 20.



(a) (b) (c)

Figure 20. Amplifier Transfer characteristics with a DIBL pFET device. (a) The inverting amplifier with a DIBL pFET as its current source. The DIBL device has a low output resistance (i.e. a low V_A). (b) Voltage transfer function of the amplifier in (a) for various bias currents. Because of the low output resistance, the gain is low; therefore, the input range over which both FETs are saturated is much greater. (c) An enlarged picture of (b). Although the input range is very large (almost a volt), the output function is (approximately) linear over the entire range. The only limitation on this circuit's linearity is device saturation.

The special symbol for the pFET denotes that it is a DIBL device; i.e. it has an ultra-short channel. We use a long-channel nFET current source; therefore its Early effect is negligable. We solve for

$$U_T ln\frac{I_{bias}}{I_o} = \kappa V_{in} + \frac{V_{out}}{A_v} \tag{17}$$

and therefore the gain from V_{in} to V_{out} is $\kappa V_a/U_T$. This gain is a constant for subthreshold currents. Figure 20 demonstrates that we get a nearly constant gain over a wide swing of input voltages, as predicted from theroy.

3.4 Differential Version of C^4 With DIBL

Figure 18b shows that the channel current through this pFET is an exponential function of both gate and drain voltage for a very short channel-length device. A device that exhibits this exponential relationship between channel current and drain voltage is referred to as a DIBL FET, because Drain-Induced Barrier Lowering (DIBL) causes this effect. The symbol used for the DIBL FET is shown in **??**a.

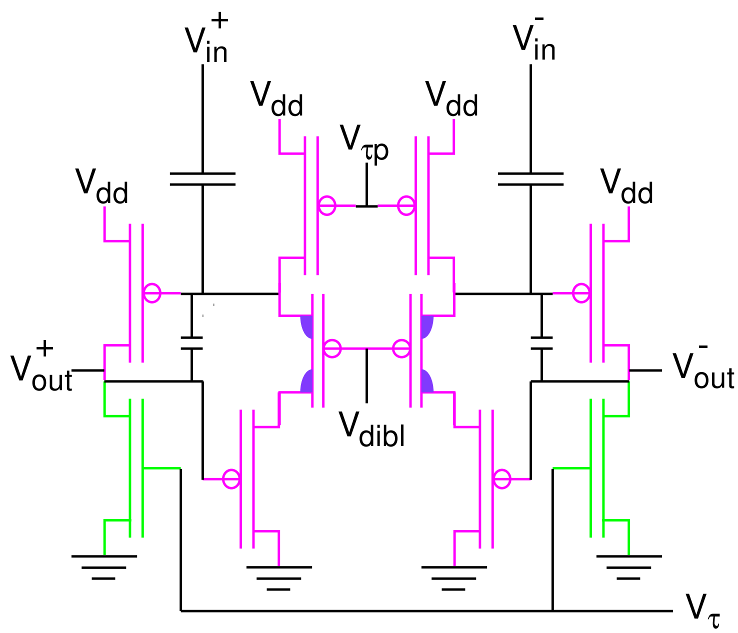


Figure 21. The circuit diagram of a differential version of this bandpass filter. An additional DIBL MOSFET was added (Figure ??) to extend the linear range of this device. The frequency response of this bandpass element is shown in Fig. 8.

We use a differential circuit topology, shown in Figure 21, to eliminate the second-harmonic distortion components and increase the filter's power-supply rejection. Also, these circuits provide the correct output as the multiplier circuits that require balanced differential inputs, as described in Section 3.5. Using the DIBL device effectively increases the linear range of these devices; in the last section we showed that harmonic distortion at the low corner is significant due to the nonlinearities of the adapting pFET transistor. The key problem for linearity for an amplifier with gain is the harmonic distortion due to the transistors setting the highpass response. Adding the DIBL device increases the linear range frin 35mV to nearly 1V. Both the single and differential amplifiers had a gain of 10, and had identical bias settings. The large amplitude (0.8V) sinusoidal signal has roughly -30dB second-harmonic distortion. The second harmonic distortion, shown in the normalized case in Figure 21, is almost completely eliminated (26 dB down) in the differential version; the differential device is limited by third harmonic distortion at its linear range. This 0.8V amplitude sinusoidal signal has roughly -37dB third harmonic

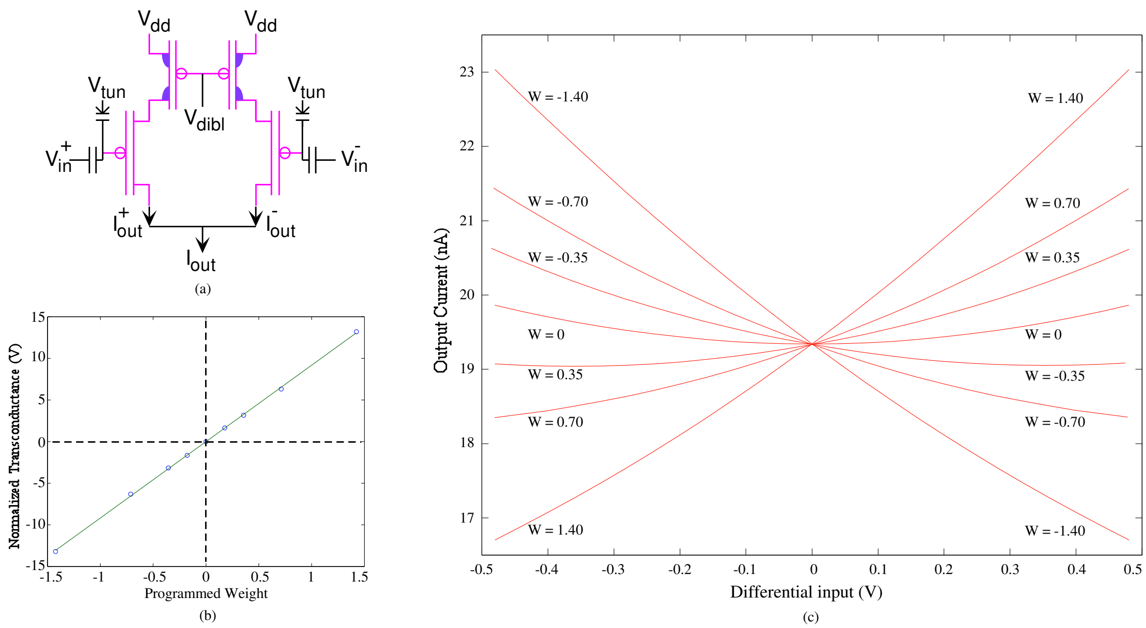


Figure 22. Four-quadrant weighted multiplication using floating-gate devices.. Shown is experimental data of the transfer characteristics of these devices. Between V_{tun} and V_{fg} is the symbol for a tunneling junction, which is a capacitor between the floating-gate and an *n*well. Clearly, there is distortion in these simple multiplier cells. More complex multiplier cells can be used to reduce distortion but at a cost of space.

distortion. The highest harmonic dominates the total harmonic distortion of this amplifier.

This bandpass circuit was originally developed as a transistor-only version of the autozeroing floating-gate amplifier (AFGA) [18, 16, 27]. The close connection to the AFGA allows for direct applications of existing results: 1. the filter's linear (minimum) range can be increased by increasing C_w, 2. A voltage input at the filter's linear range corresponds to -26dB second-harmonic dominated distortion, 3. The total output-noise power is roughly proportional to C_w, and is inversely proportional to C_L, and 4. We can increase the linear range by increasing C_w, and we can increase the dynamic range by increasing C_w and C_L [20].

3.5 Floating-Gate Input-Weight Multiplier

Figure 22 shows the circuit model for our four-quadrant multiplier. This circuit was presented initially as part of a four-quadrant synapse [24]; the DIBL devices enhance the

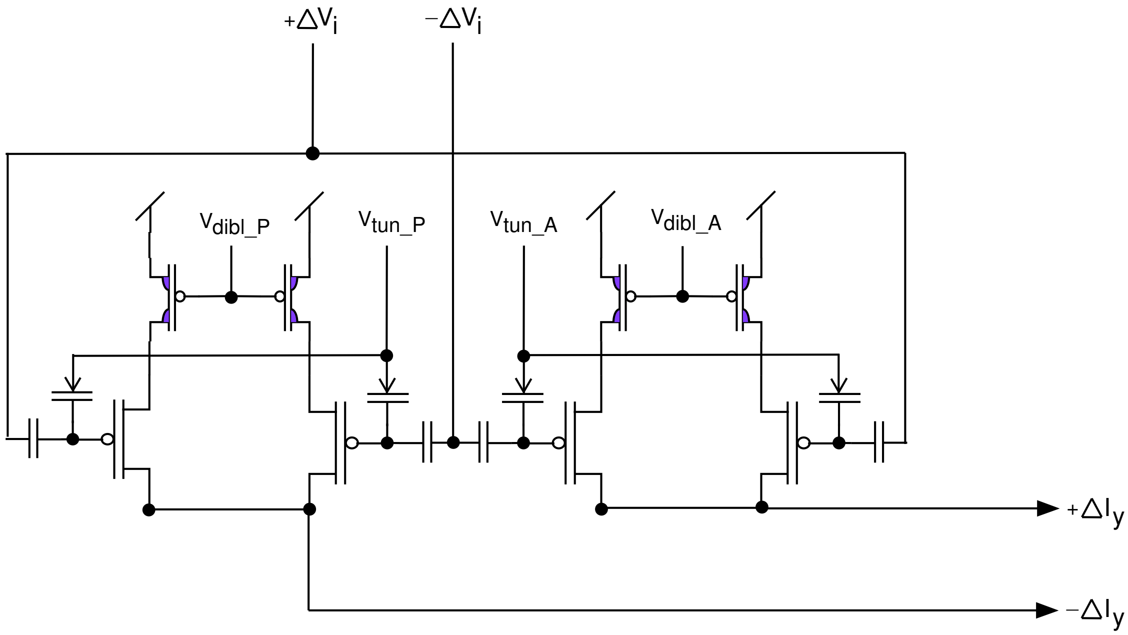


Figure 23. Differential Structure for 4-Quadrant Operation to reduce even harmonic distortion. However, this more than doubles the cell size from the simple 4-quadrant multiplier.

linear range of these synapses while also providing the correct feedback to generate familiar Hebbian learning rules [24, 16, 10]. Figure 22 also shows the measured data from the floating-gate weighted multiplication. A reasonable multiplication is obtained over a 0.5V differential input range for a positive and negative range in weight values. Second harmonic distortion dominates this multiplier as seen from the $W = 0$ curve. Alternatives such as the differential multiplier shown in Figure 23 or current mode approaches [48] can be used to reduce to even harmonics from the current multiplier. Solutions to reducing the distortion come with a space tradeoff affecting density. Therefore, if the application can tolerate the distortion the multiplier in Figure 22 should be used. Offsets due to the inputs and mismatch are not a problem because each weight is explicitly programmed and can be set to eliminate the offsets.

To derive equations to model this multiplier behavior, begin with

$$
\begin{aligned}
I_{out} &= I^{+} + I^{-} \\
I_{ds} &= I_o \frac{W}{L} \exp^{\frac{-\kappa_p V_{fg}}{U_T}} \\
V_{fg} &= \frac{Q_{fg}}{C_T} + \frac{C_{in}}{C_T} V_{in}
\end{aligned}
\tag{18}
$$

If we re-write V_{fg} to have an input around a bias point

$$
V_{fg} = \underbrace{V_{prog} + V_{inbias}} + \frac{C_{in}}{C_T} V_{in} \tag{19}
$$

where $V_{prog} + V_{inbias}$ is the bias point.

The current output is summed using KCL and can be expanded from 18 as

$$
I_{out} = I_{so} \frac{W^{+}}{L^{+}} \exp\left(\frac{-\kappa_p (V_{bias^{+}} + \frac{C_{in}}{C_T} \Delta V_{in^{+}})}{U_T} \right) + I_{so} \frac{W^{-}}{L^{-}} \exp\left(\frac{-\kappa_p (V_{bias^{-}} + \frac{C_{in}}{C_T} \Delta V_{in^{-}})}{U_T} \right) \tag{20}
$$

We can represent the bias term in each floating-gate device as

$$
\begin{aligned}
w^{+} &= \exp\left(\frac{-\kappa_p V_{bias^{+}}}{U_T} \right) \\
w^{-} &= \exp\left(\frac{-\kappa_p V_{bias^{-}}}{U_T} \right)
\end{aligned}
\tag{21}
$$

If we make the assumption the devices are identical, define a term $V_y = (U_T C_T)\,(\kappa_p C_{in})$ as the exponential slope of this element between capacitive input and channel current, and assume $\Delta V_{in^{+}} = -\Delta V_{in^{-}}$ 20 can be re-written as

$$
I_{out} = I_{so} \left(W^{+} \exp\left(-\Delta V_{in}/V_y\right) + W^{-} \exp\left(\Delta V_{in}/V_y\right) \right); \tag{22}
$$

The exponential slope V_y is a direct factor of the capacitive voltage divider into the floating-gate. This voltage V_y is typically around 1V. assuming the inputs are within the input linear range, V_y, then we approximate the exponentials as linear functions:

$$
I_{out} \approx I_{so}(W^{+} + W^{-}) + I_{so}(W^{+} - W^{-}) \frac{\Delta V_{in}}{V_y}, \tag{23}
$$

where W^+ and W^- are the weights corresponding to pFET devices connected to V_{in}^+ and V_{in}^-, respectively. The synapse weight, defined by $W^+ - W^-$, and ΔV_{in} take on both positive and negative values; therefore the change in the output current is a four-quadrant product of the input by the synapse weights for fast timescales.

The circuit shown in 23 gives a four-quadrant multiplication between the input and a stored weight. This synapse couples two source-degenerated(s-d) floating-gate pFETs in a way that subtracts out their common responses to achieve a four-quadrant multiplication. This circuit supplies a differential output unlike the previous multiplier that converted the differential input to a single ended output. This circuit has the added benifit of being a differential system where even order harmonics are reduced are reduced at the output.

Additional pre-distort circuits have been designed for the multiplier to increase it's linear range and convert it to a current mode multiplier [9]. Further current mode version of this multiplier have been examined an determined to provide a 531nW/MHz multiplier that is linear over two decades [6]. This provides 1 million MACs/0.27μW as compared to a commercially available DSP which achieves 1 million MAC/0.25mW.

CHAPTER 4

PROGRAMMING ARRAYS

4.1 Array Configuration of the Floating-Gate Elements

As with any new technology, these floating-gate devices present new challenges that must be addressed before it can be a viable solution in mainstream applications. Many of these headaches are very similar to the digital counterparts of ACAs, EEPROMs [3]. The major similarity is how to modify the information in the device to a desired value. Yet, while similar in physics, the desired application and use of the device lead us to different schemes to alter the stored information. The desire is to deliver a system that will allow anyone to realize the benefits of floating-gates in analog systems, even without a deep understanding of floating-gate devices.

The floating-gate core used in the computational arrays differ from those used in other analog memory circuits such as Epots [13]. Epots are made ”User-Friendly” by the addition of several control circuits around each floating-gate element such that the overall circuit block is an order of magnitude larger than our single floating-gate elements. The benefit of large floating-gate computational arrays is the compactness of each core element and therefore support circuits must be moved to the arrays periphery to maintain a dense repetitive core.

Presented in this chapter is a custom programming board and a physics based algorithm that is able to quickly and accurately program the floating gates in large arrays. Hopefully the user friendly analog programming scheme places analog floating-gates in the toolbox of circuit designers. Eventually making them as easy to understand and easy to use as digital EEPROMs which is currently an accepted technology in modern circuit design [18].

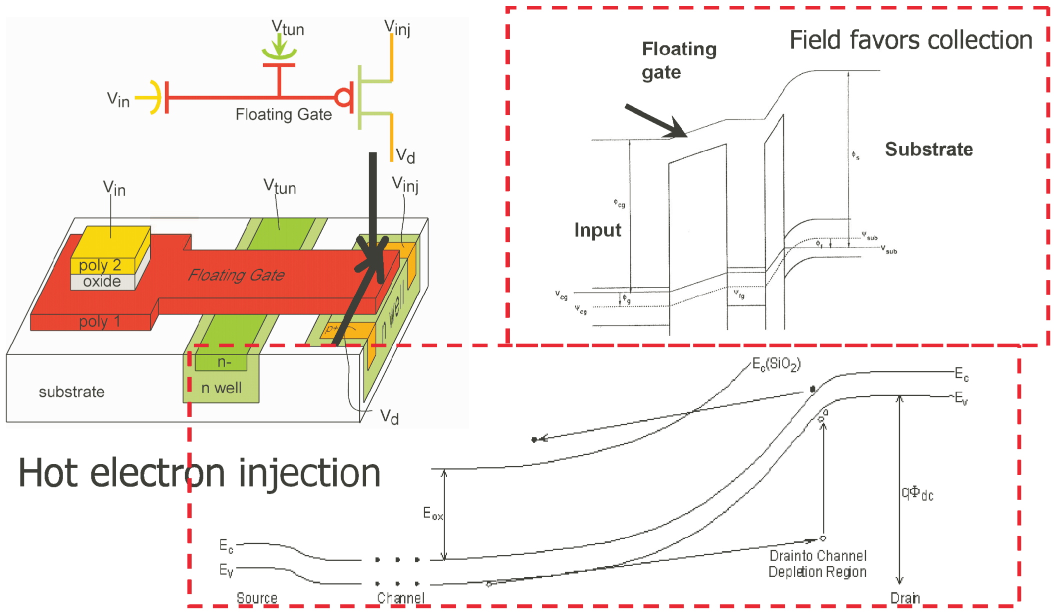


Figure 24. The pFET device is chosen as our floating-gate device due to it's device characteristics. The pFET device has the channel to floating-gate field in the correct direction to collect excited carriers onto the floating-gate. Shown is the schematic representation of the floating-gate device, the cross section, and the injection fields of interest.

4.2 Floating-gate Device Overview

The floating-gate device that we use in our systems is shown in Fig. 24. The pFET device is chosen as our floating-gate device due to it's device characteristics. The pFET device has the channel to floating-gate field in the correct direction to collect excited carriers onto the floating-gate. A nFET device must use a pBASE layer [15] or dual-gate Source Side Injection (SSI) [54] structure to permit hot-electron injection to occur. These are both special process steps that also alter the operation of the FET when it is used in a computation and not just a memory element. The downside to choosing a pFET device for injection is the requirement to reference everything from V_{dd} instead of gnd. This does present some headaches when programming especially when charge-pumps are used to provide the injection voltage.

The pFET floating-gate device shown in Fig. 24 can modeled by

$$I_s = I_{so} \exp\left(\frac{V_s - \kappa_p V_{fg}}{U_T}\right) \exp\left(-\frac{V_{ds}}{V_o}\right) \tag{24}$$

where V_{fg} is the change in the floating gate voltage, κ_p is the fractional change in the pFET surface potential due to a change in V_{fg}, V_o is the Early voltage of the pFET, and U_T is the thermal voltage, $\frac{kT}{q}$.

The floating-gate voltage is made up of

$$V_{fg} = V_1 * \frac{C_1}{C_T} + V_2 * \frac{C_2}{C_T} ... + Q_{stored} \tag{25}$$

where C_T is the total sum of all the capacitances onto the floating-gate that has a voltage V_{fg}. The floating-gate voltage also has a Q_{stored} term which is the charge stored on the floating node.

The charge on the floating-gate can be programmed using one of three methods. First, is electron generation from radiation such as ultra-violet (UV) light or cosmic rays. Second, is tunneling through oxides with high fields. Last, is using electrons with enough energy to overcome a barrier, generally hot electron injection. We avoid the use of UV exposure for programming or erase due to the need for an external source of light and the high cost of a package with a window. The next section on device selection will explain the how and why tunneling and hot-electron injection is used to erase and program the floating-gate charge.

4.3 Device Selection in Arrays

Developing an efficient algorithm for pFET programming requires discussing the dependencies of the gate currents, and the ability to modify a single device with high selectivity. A device is programmed by increasing the output current of a pMOS transistor using hot-electron injection, and decreasing the output current using electron tunneling. In the proposed scheme, devices are programmed using hot-electron injection and are reset by tunneling the devices below the level to which they are to be programmed. This scheme is chosen due to device selectivity of the two different processes and tunneling junction mismatch that effect predictability for modifying floating-gate charge in an array. A time response of one-half a second was used to produce Figure 25 because of the speed of the instruments needed to carefully characterize this measurement [17]. The floating-gate

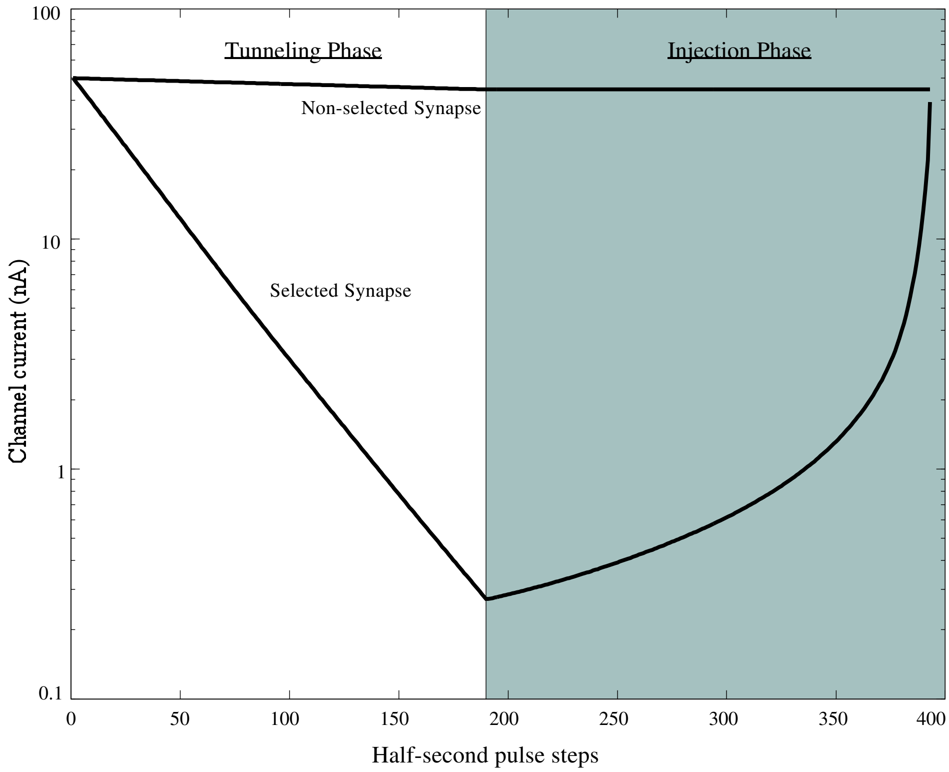


Figure 25. Output currents from two elements on the same row of a floating-gate pFET array, showing 115 tunneling operations followed by 200 injection operations. The floating-gate charge is modified using several 0.5s pulses. [15, 25]

devices easily handle responses in ranges down-to and below micro-seconds by simply increasing the tunneling and drain-to-source voltages used during programming and has been show to work at this rate using the new updated programming board.

Before deciding on a particular programming scheme, the synapses(floating-gate pFET with DIBL device) interaction when coupled into an array was considered. The device interactions are due entirely to the nonlinear dependence of the terminal voltages on the floating-gate current. The tunneling and drain terminals are chosen to be common along a row; therefore when programming one row, the other rows remain unaffected. Finally, how to selectively modify the charge on a particular floating gate without affecting the other floating gates along the same row had to be considered.

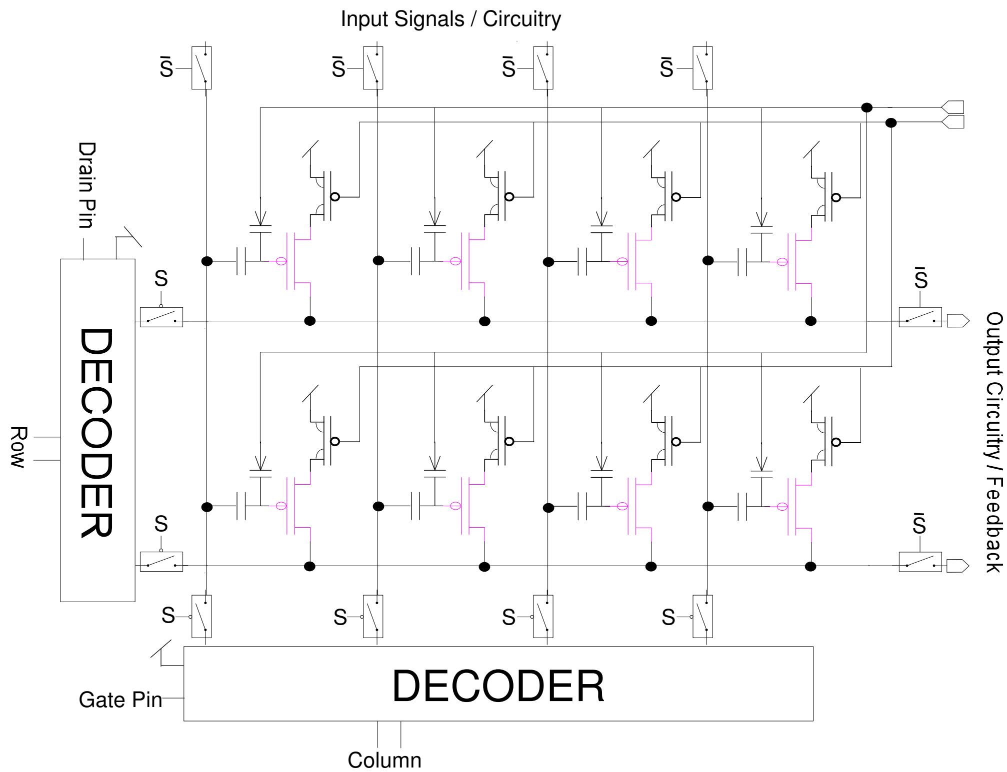


Figure 26. Circuit diagram of chip design to allow dual programming / operation. When the control signal S is 1, then the switches are closed to the decoder circuitry, enabling programming, and the switches are opened to the normal operating circuitry. Both decoders either set their outputs to V_{dd} if 0 or select an output to an external pin. When the control signal S is 0, then the switches are open to the decoder circuitry, and the switches are closed to the normal operating circuitry. The goal is to fit all floating gates used into this array format which is described in Chapter 5.

Figure 25 illustrates that a single floating-gate device along a row can be programmed with minimal effect to its neighbors. Tunneling selectivity along a row in this array is entirely a function of how far apart the a selected floating-gate voltage vs. the non-selected floating gate voltages can be pushed by the gate inputs. This is due to the fact that the amount of tunneled current is based on the voltage across the tunneling capacitor. This voltage across the tunneling capacitor is equal to $V_{fg} - V_{tunnel}$ as can be seen in Fig. 24. As equation (25) shows the floating-gate voltage is made up of several terms including the charge stored on the floating-gate. Exponentially, more tunneling current is obtained for each linear increase across the capacitor because of the probability of electrons tunneling through the barrier [42]. In brief, this is because as the effective barrier height is lowered due to electrostatic influence of an electron approaching the interface and the Fermi-distribution of the electrons in the conduction band. To select a particular synapse, that floating gates control input is brought to as low a voltage as possible, while at the same time bringing all the remaining floating gates control input to has high a voltage as possible. The hope is that the coupling into the floating-node is large enough that the V_{fg} for the selected device is much lower than that of the non-selected devices. Because the floating-gates voltage depends also on the stored charge device selective can change with programmed values, a non-desired attribute. The selectivity ratio for the pFETs in Figure 25 on the same row is roughly 40 for a 5V supply. The tunneling selectivity can be increased by increasing the input voltage steps or by increasing the gate coupling to the floating gate providing better floating-gate control. Generally this input gate coupling is set by the computation and size requirements. Because of the poorer selectivity, tunneling is primarily used for erasing and for rough programming steps.

The initial topology proposed[35] with tunneling junctions combined along rows only, has been simplified in actual implementation by tying all tunneling rows together to consume only one pin on the package. This avoids the need for high-voltage transistor switches to tunnel along an individual row, which may not be available in all process. There is a drawback to using this simplified approach, tunneling can then only be selected at best down to a column of synapses because the gates are shared along the columns. However, this simplified approach works successfully in the overall programming scheme because, tunneling is used primarily for erasing.

Figure 27 shows experimental measurements of pFET injection versus drain-to-channel voltage[15]. Injection current in the transistor is modeled by the following equation:

$$I_{inj} = I_{inj0}\left(\frac{I}{I_{so}}\right)^{\alpha} e^{-\Delta V_d/V_{inj}}, \tag{26}$$

where $\alpha = .93$ and $V_{inj} = 400mV$. For injection to occur in a device there are two controlling parameters, the source-to-drain voltage (ΔV_d) to create the high field needed and the gate voltage to create current the MOS channel(I_{so}). Therefore as shown in Fig. 28, a device in the array is selected to be programmed by lowering the column voltage, containing the gate of the device, to around threshold(optimal injection voltage) and the row voltage, containing the drain of the device, to a voltage to produce injection, while all other rows and columns are tied to V_{dd}. Because both conditions described earlier are then satisfied for only one device, we have the ability to select a device to program out of the array. For larger source-to-drain voltages, exponentially, more injection current is obtained as shown in (26) and verified by Fig. 27. To control the amount of injection in the device, the source-to-drain voltage is modulated by raising or lowering the row's drain voltage. During programming, the system voltage is raised to values that allow injection in that process. Selectivity in this process is also dependent upon the floating-gate charge. If the charge is increased in any device to permit channel current to flow when V_{dd} is applied to the control input gate the device will still inject when the drain voltage is pulsed. All target current calculations for the device are performed with the same drain-to-source voltage as during operation at the specified gate voltage. This allows the device to be programmed for its operational voltages, unlike other programming algorithms that suffer from drain-gate couple changes when coming out of programming mode.

4.4 Floating-gate Array Programming Scheme

The programming scheme for analog cells must be different than that of digital memory due to the difference in their objectives. For digital or binary circuits the desired solution is only interested in resolving two distinct values for each cell, that is one or zero; newer digital systems store information using multiple levels but fundamentally use approaches similar to binary valued cells. Analog programming systems require a continuous range of programmable values. Also in digital systems, design considerations demand a device that can continuously and quickly store and read information. In an analog system we are generally more interested in programming a device once and then reading or, as we use them,

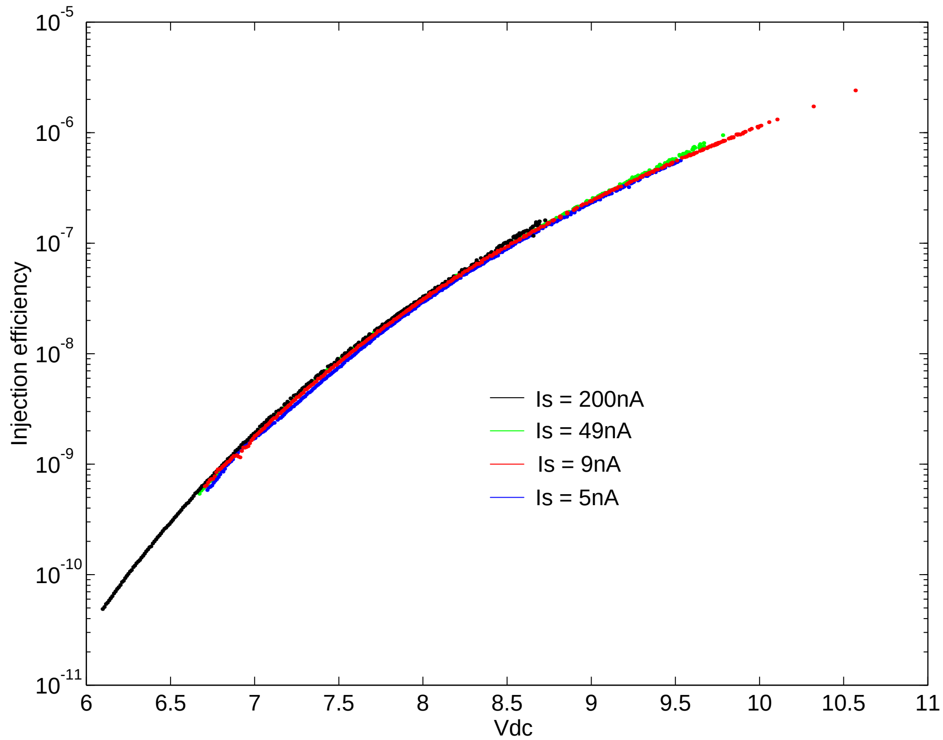


Figure 27. Measured data of pFET injection efficiency versus the drain-to-channel voltage for four source currents. Injection efficiency is the ratio of injection current to source current. The injection efficiencies are nearly identical for the different source currents; therefore, they appear to be indistinguishable on the plot. At drain-to-channel voltages equal to 8.2V, the injection efficiency e-folds (increases by a factor of e) for a 250 mV increase in drain-to-channel voltage. [17]

computing continuously afterwards. Therefore in the analog system, instantaneous programming of each cell is not as critical while accuracy of programming is far more critical. Speed is still relatively important as large arrays are to be programmed, but our approach prioritizes accuracy considerations over speed as evident by the iterative approach.

Programming a floating-gate element involves being able to adjust multiple control voltages of a single element. As shown in Fig. 28 it is possible to isolate individual elements in a large matrix using peripheral control circuitry. Using the control circuitry, one is able to access and isolate the gate and drain voltage of a single element. This orthogonal isolation allows a single device to be programmed through correctly applied control voltages.

Any circuit containing programmable floating-gate elements must also have various

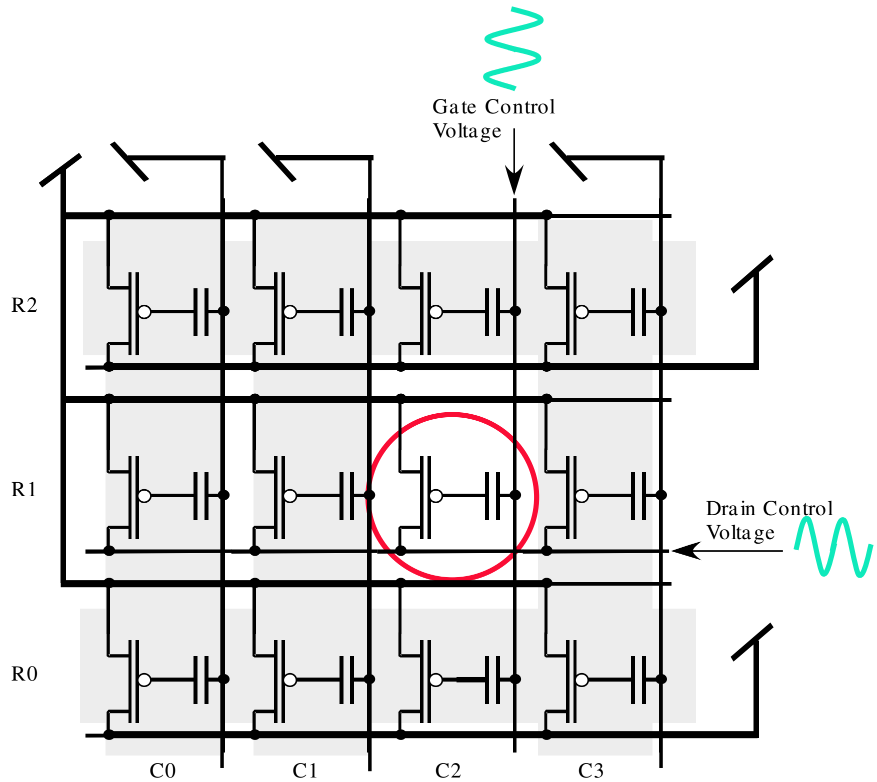


Figure 28. Floating-gate array demonstrating element isolation by controlling the gate and drain voltage of each column and row. Selection of gate and drain voltages are controlled by on-chip mux circuitry.

switching circuitry to access each floating-gate element. The programming method uses gate isolation per column and drain isolation per row. Individual circuit blocks are designed assuming access by rows and columns and the control circuitry is designed accordingly. This method of device isolation ensures that 1.) there will only be sufficient drain to source voltage for injection of elements in a given row, and 2.) the only element in that given row with any current flowing will be in the selected column. Row selection switches the drain lines of a row of circuit blocks to the appropriate control signal while all other drains are connected to a different control line which is typically connected to VDD for the chip. Column selection switches the gate lines of an entire column to an single control line while all other gates are switched to another line which is typically connected to VDD. This configuration ensures that control voltages for the selected element are available externally

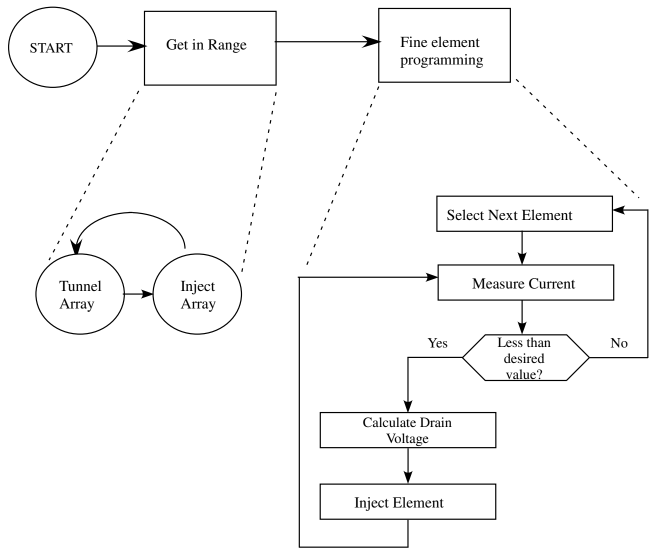


Figure 29. Flow chart of programming algorithm. Devices first erased using electron tunneling. They are iteratively programmed using hot-electron injection. A drain voltage is calculated using a physics based equation and then applied as a rapid pulse to the device.

for injection control, while at the same time elements not being programmed are off during the process. Because of the simplicity of the switching scheme, which requires a single pFET in most cases, or a complimentary set of full transmission gates in the most complex case, we are able to minimize the overall size of individual circuit blocks.

The isolation circuitry, shown in Fig. 26 is made of muxes that switch the drain and gate voltages of the desired element onto a common bus for each signal. All other elements are switched to a separate voltage which ensures that those devices will not inject. The external voltages are routed off-chip and controlled by an external programming board [36]. A typical programming scenario is shown in Fig. 28.

4.5 Floating-gate Programming Algorithm

Using large floating-gate arrays on the order of 1K to 10K elements, it becomes obvious that programming each element by hand would be a very time intensive process. Fig. 29 shows a block diagram of the programming algorithm. The programming algorithm has been automated into a single MATLAB function comprised of three lower level functions.

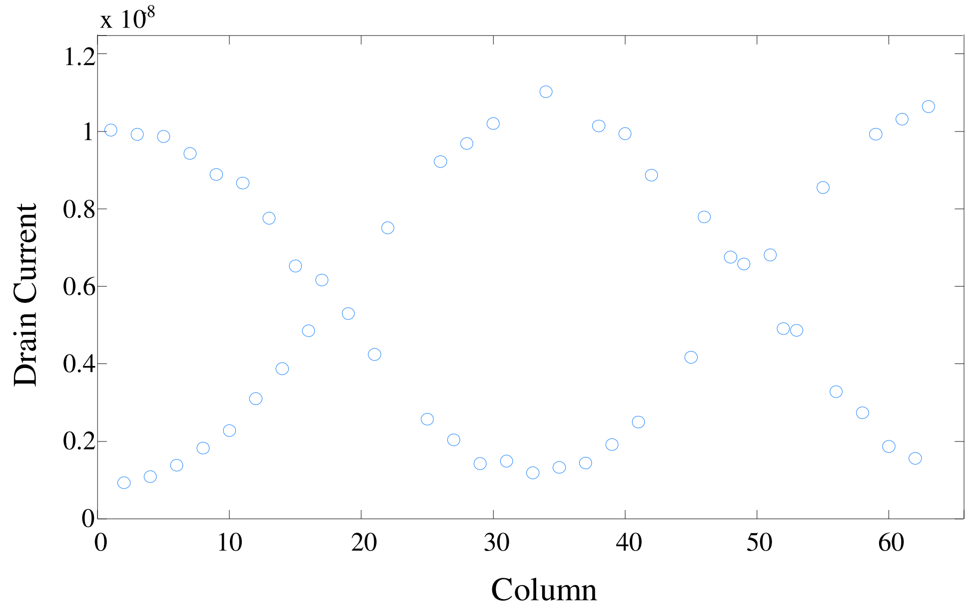


Figure 30. A single row of floating-gate multiplier blocks programmed to cosine coefficients. These blocks are essential to performing analog frequency transform functions. Because the values are arbitrary, one can also set these linearly or to increase or decrease logarithmically.

These functions include:

1. TunnelToRange()

Tunnels entire array until all current levels are below their desired values. This function is required because tunnelling is used for global erase and injection is used for fine programming but only works in a single direction.

2. InjectToRange()

Ensures that all elements have a sufficient current at a given gate voltage to increase the speed of injection. Injection requires drain current. Tunnelling is a global function so some elements may have significantly lower current levels than others. Future revisions will allow isolation of tunnelling as well.

3. InjectArray()

Controls the injection of an entire array by calculating the optimal drain voltage to ensure that each injection pulse brings the element closer to the desired current

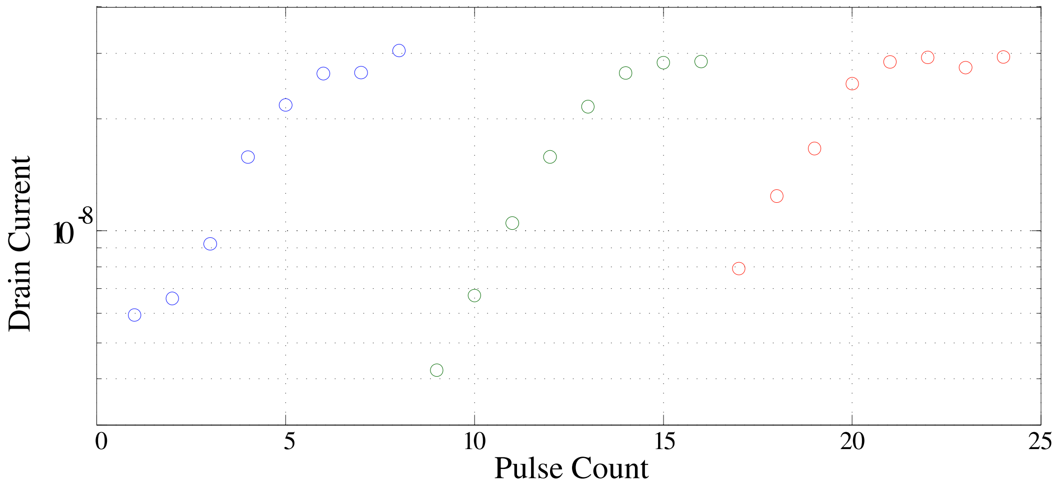


Figure 31. Injection of a single element over 3 different trials. Each trial has a desired current of 30 nA. After the desired current is reached, the gate voltage is increased to lower the current and the programming algorithm is repeated.

without overshooting.

Based on transistor physics we are able to solve for the injection drain-to-source voltage for a desired current. Hot-electron injection in a pFET is described by (27)

$$I_{inj} = I_{inj_0}\left(\frac{I_S}{I_{S_0}}\right)^{\alpha} e^{-\Delta V_d/V_{inj}} \tag{27}$$

where I_{S_0} is the initial current, I_S is the final current, and V_{inj}, I_{inj}, and α are physical device parameters that are obtain from characterization data or can be iterativly fitted. The injection current is also dependent upon the change in the floating-gate voltage V_{fg} [25].

$$C_T \frac{\partial V_{fg}}{dt} = -I_{inj} \tag{28}$$

and the current in the subthreshold transistor modeled by

$$I_S = I_{S_0} e^{-\kappa \Delta V_{fg}/U_T} \tag{29}$$

Solving for I_S we start with

$$\frac{d(I_S)}{dt} = I_{S_0} e^{-\kappa \Delta V_{fg}/U_T}\left(\frac{-\kappa}{U_T}\right)\underbrace{\frac{\partial V_{fg}}{dt}} \tag{30}$$

substituting (28) into (30)

$$\frac{d(I_S)}{dt} = \underbrace{I_{S_0} e^{-\kappa \Delta V_{fg}/U_T}}\left(\frac{-\kappa}{U_T}\right)\left[\frac{-I_{inj}}{C_T}\right] \tag{31}$$

then substituting (29) into (31)

$$\frac{d(I_S)}{dt} = I_S\left(\frac{\kappa}{U_T}\right)\left[\frac{I_{inj}}{C_T}\right] \tag{32}$$

finally substituting (27) into (32)

$$\begin{aligned}
\frac{d(I_S)}{dt} &= I_S\left(\frac{\kappa}{U_T \cdot C_T}\right)\left[I_{inj_0}\left(\frac{I_S}{I_{S_0}}\right)^{\alpha} e^{-\Delta V_d/V_{inj}}\right] \\
\frac{d(I_S)}{dt} &= I_S{}^{1+\alpha}\left(\frac{\kappa}{U_T \cdot C_T}\right)\left[I_{inj_0}\left(\frac{1}{I_{S_0}}\right)^{\alpha} e^{-\Delta V_d/V_{inj}}\right]
\end{aligned} \tag{33}$$

Solve by separation of variables.

$$\begin{aligned}
C_0 &= \frac{U_T \cdot C_T}{\kappa} \\
\int \frac{d(I_S)}{I_S{}^{1+\alpha}} &= \int \frac{I_{inj_0}}{C_0 \cdot I_{S_0}{}^{\alpha}} \cdot e^{-\Delta V_d/V_{inj}} dt \\
\left(\frac{-1}{\alpha}\right)\frac{1}{I_S{}^{\alpha}} &= \frac{I_{inj_0}}{C_0 \cdot I_{S_0}{}^{\alpha}} \cdot e^{-\Delta V_d/V_{inj}} \cdot (t) + A \\
I_S{}^{-\alpha} &= \frac{-\alpha \cdot I_{inj_0}}{C_0 \cdot I_{S_0}{}^{\alpha}} \cdot e^{-\Delta V_d/V_{inj}} \cdot (t) + A \\
A &= I_{S_0}{}^{-\alpha} \quad @ \ t = 0
\end{aligned} \tag{34}$$

resulting in I_S of

$$I_S{}^{-\alpha} = \frac{-\alpha \cdot I_{inj_0}}{C_0 \cdot I_{S_0}{}^{\alpha}} \cdot e^{-\Delta V_d/V_{inj}} \cdot (t) + I_{S_0}{}^{-\alpha} \tag{35}$$

where $C_0 = U_T \cdot C_T/\kappa$ [17, 19].

Rearranging this solution into a single equation for ΔV_d, these equations are simplified into a solution of a single equation for the required drain voltage to reach a desired current.

A change in drain voltage is calculated around a quiescent drain voltage that gives a 10 percent change in current for each injection pulse.

$$\Delta V_d = -\ln\left[\frac{-C_0}{\alpha \cdot I_{inj_0} \cdot (t)}\left[\left(\frac{I_{S_0}}{I_{S_{desired}}}\right)^{\alpha} - 1\right]\right] \cdot V_{inj} \tag{36}$$

The required drain voltage is calculated for each element in the array given their present value of current and the final desired current. These voltage values are then applied to each element in the array [17].

Using floating-gate elements in analog computation requires that each element be programmed accurately to a desired current. Figure 30 shows a row of floating-gate multipliers that have been programmed as a differential cosine weighting function. These elements perform a scale multiplication on a differential input signal.

4.6 Pogramming Speed Issues

With large arrays of floating gates, on the order of 1K to 10K, speed is of paramount importance for this technology to be viable. Programming speed is limited by four factors:

1. Injection pulse width.

2. External current measurement.

3. Serial communications for element selection.

4. Parallelism exploited within the system.

Injection has been shown to occur with pulses down to 10μs using the new FPGA controlled programming board. The change in the floating gate can be increased for an applied pulse by either increasing the injection pulse time or increasing the drain-to-source voltage. Smaller pulse times can be applied for programming by increasing the source-to-drain voltage. Figure 32 shows injection rates as the injection pulse width increases. Fig. 31 shows convergence in eight to ten iterations, which results in an individual programming time of 80 - 100 μs. For an array with 2K floating-gate elements, the total programming time could be as low as 80 - 100 ms.

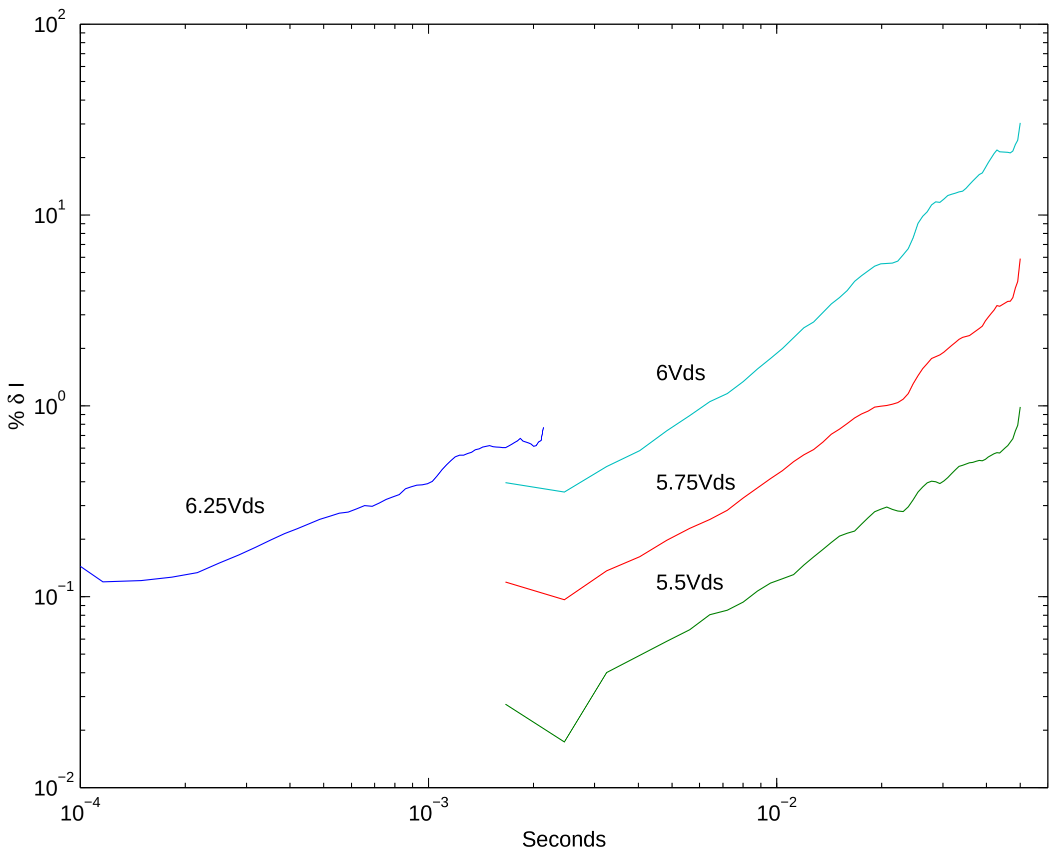


Figure 32. Plot of injection rate versus injection pulse width for different drain-to-source voltages. The injection pulse width was limited by programming hardware and is shown to occur at a pulse width of 100μs. The minimum injection pulse width can be decreased by moving the control circuitry on chip.

External current measurements are limited due to the huge line capacitance for wires running off chip and the equipment used to perform the current measurement. The current measurement circuitry can typically provide one current measurement between 10μs for large currents and 100ms for very small currents. Off chip also requires additional filtering of the data to ensure accurate results which can make the measurements even slower. It is appearant that for rapid programming the current measurement circuitry must be on-chip.

Programming board serial port communication is limited by port speed and also the operating system running the algorithm. Using windows machines has shown to add additional serial port timeout delays. The programming algorithm in a parallel effort by colleagues has been moved onto a field-programmable gate array (FPGA), thereby removing

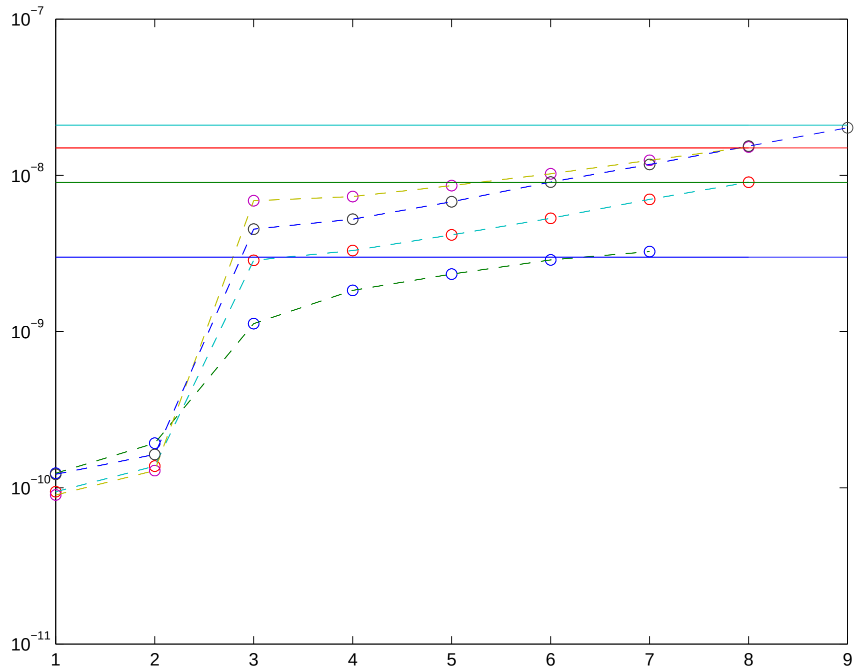


Figure 33. All four values converged within 9 steps.

the OS delays. Moving circuits on-chip to programming an entire row or column at a time will significantly reduce overall programming time.

Initial programming results were obtained from a 1.2μm MOSIS process. A 2x4 array of floating gates was used for this experiment. The operation voltage for the chip was 3V. For programming, 8V was used to allow for significant injection in this process to occur. While in program mode, the drain voltage was held at 5V during the current measurements for system operation with a 3V supply. The timing, T, used for injection was 2 seconds. This value was chosen only to ensure no timing issues in the test environment. Figure 33 shows an attempt at programming four devices in the array to different values. Figure 31 shows the attemps on a 0.5μm process targeting the same current.

4.7 Custom Programming Board

The floating gate computing array has a large potential in many systems. The arrays real benefit is gained when its parallelism is stretched to array sizes well beyond 100 or even

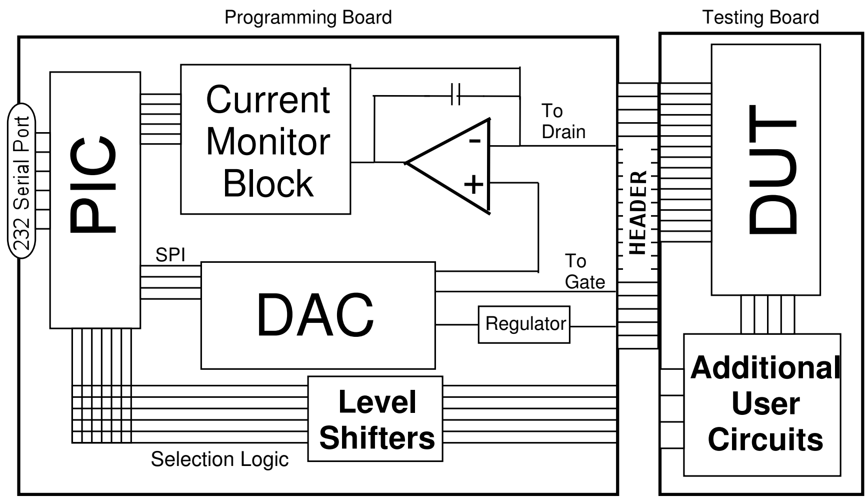


Figure 34. Block diagram of our custom programming board for automatic programming of large floating-gate arrays. This board, controlled by a PIC micro controller and interfaced with a computer through a serial port, is capable of programming floating-gate arrays fabricated in a wide range of processes. This board allows easy integration with a larger testing platform, where programming and computation are both required.

1000 floating gate elements. Manual programming via knob tweaking or even experimental lab setup, utilizing testing equipment, methods of programming these arrays quickly show their limitation as array size grows. Also, if you were presented with the task of programming many systems you would run into the same problem, programming time per element. The solution is a custom programming board containing circuits able to quickly and accurately program the large number of floating gates in these large arrays.

Further motivating the design of this programming board, is a low cost alternative to racks of testing equipment needed to program systems utilizing the ACAs. Not only does using self contained custom testing circuits delivers the speed needed to realize the programming of large arrays, but ultimately this programming board will remove the need for the low-level understanding of how to program the arrays much like a PIC programmer does for PICs. The system in Figure 34 allows for flexible floating-gate array programming over a wide range of IC processes, and allows for nearly transparent operation to the user. This board has three external connectors, a power connector, a serial port for communication with a computer, and a standardized header to connect to the testing board. The

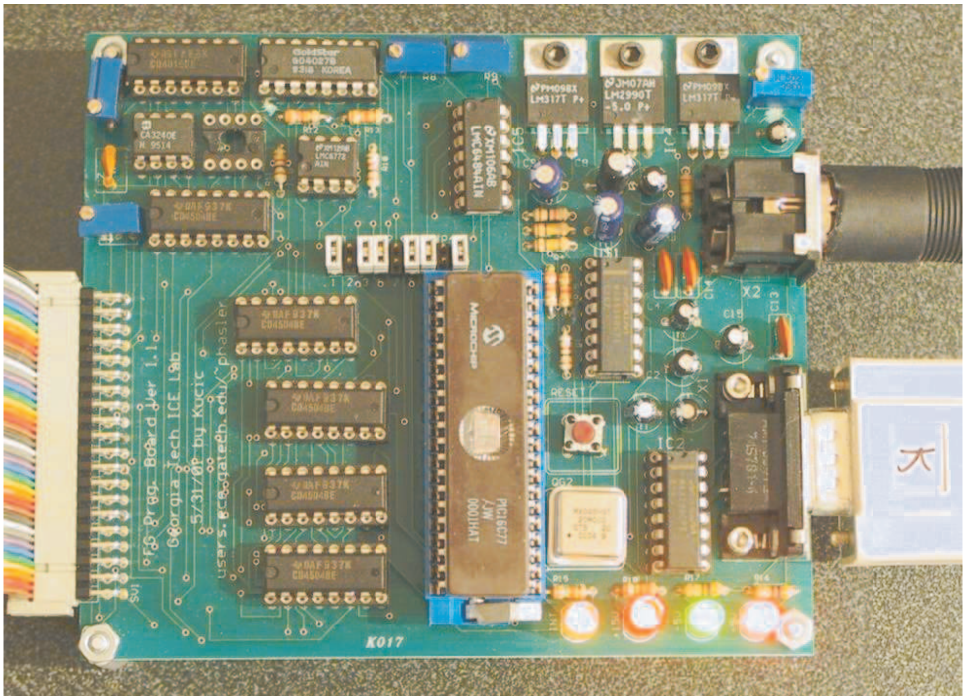


Figure 35. This is version 1.1 of the programming board. Over 20 of these were assembled and used for programming systems in the ICE lab and also for lab experiments in graduate classes. This board has been updated to a newer version that function with a separate and dedicated FPGA development board.

standardized header provides low level digital signals for decoders and run/program modes, power for the testing board, and analog signals necessary to program the array. The header connects to a testing board allowing the flexibility to use the programmer across different chips which may not contain all the on chip circuits necessary to program the array. At the heart of the programmer is a PIC16C77 micro-controller, providing support for the serial interface, the DACs on the board, the current measurement circuits, and the accurate timing necessary for programming.

The current monitor block circuit on the programming board is used to drive the drain lines of the computing array during programming. The requirement for this circuit is to set a voltage while being able to read a current. This circuit replaces the Source Measure Unit(SMU) used in initial bench testing. The circuit shown in Figure 36 was designed to meet these needs in a discrete form off-chip. Using an OPAMP, the drain voltage is set

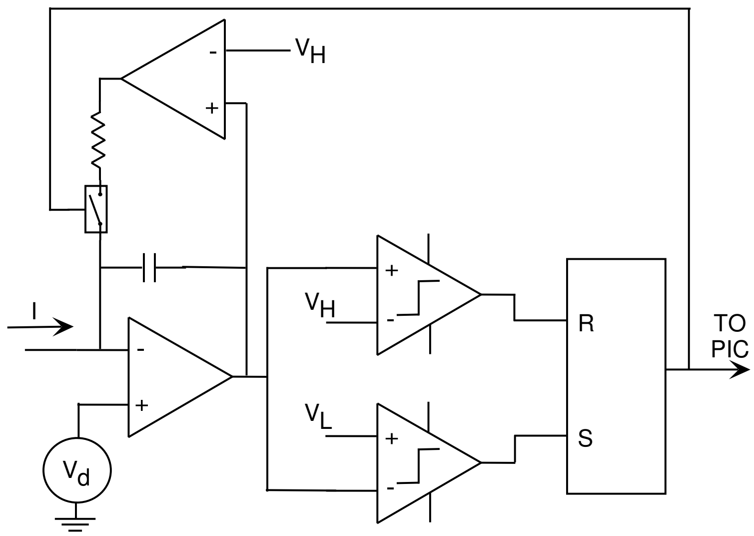


Figure 36. This circuit is used to set a drain voltage while being able to read the resulting drain current. The integrator integrates the incoming current until the output voltage of the OPAMP drops past a threshold triggering the comparator on the SET input of the S-R Latch. The S-R latch closes the switch placing the second OPAMP around the integrating amplifier. This discharges the integrating capacitor until the output of the integrating OPAMP goes above an upper threshold triggering the comparator on the RESET input of the S-R Latch, opening the switch allowing the integration to resume.

on the non-inverting input and appears on the inverting input through it's virtual short. By using a capacitor for the feedback, the OPAMP integrates the incoming current that can be seen as the output voltage dropping in proportion to the current. Unfortunately, if left alone the integrator output would reach the ground rail, after which the feedback would not hold. The two comparators on the output of this integrator are used to catch the output before it reaches either supply rails. The output of the comparators is connected to the S-R latch to enable the discharge circuit only long enough to reset the integrator to a known voltage point. The integrator capacitor then recharges until the catch point is reached and the cycle repeats as is shown in Figure 37.

The PIC monitors the output of the S-R latch, producing a time value indicating the time required to charge the capacitor from a known voltage to another known voltage. Knowing

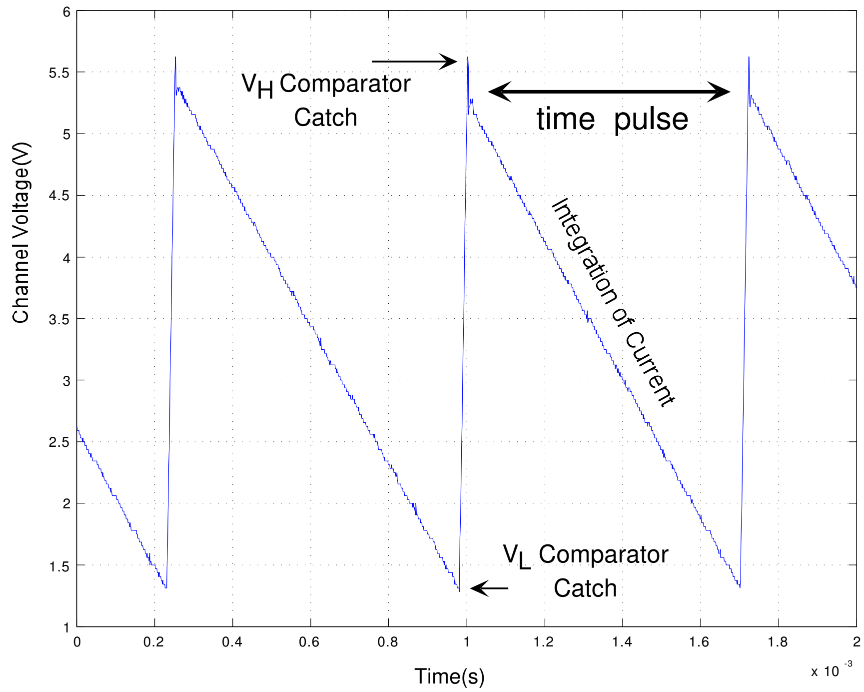


Figure 37. This is a scope trace of the integrators output. When the output drops below a set point the integrating capacitor is discharged until the output reaches the upper voltage catch stopping the discharge. The Δt is read by the PIC and converted into a current value.

the capacitor directly results in the current via

$$I = C\frac{\Delta V}{\Delta t}. \tag{37}$$

With ΔV and C known, we simple combine them into a constant, A, relating I to Δt

$$I = \frac{A}{\Delta t}. \tag{38}$$

The choice of C will determine the current range over which can be read as well as the time need to make the measurement. With a little additional logic and circuitry a future revised version will be able to auto-range by selecting from various capacitors on the programmer. To auto-range to a lower current the PIC would simply have a time limit which when exceeded switches to the smaller capacitor, meaning the previous capacitor took too

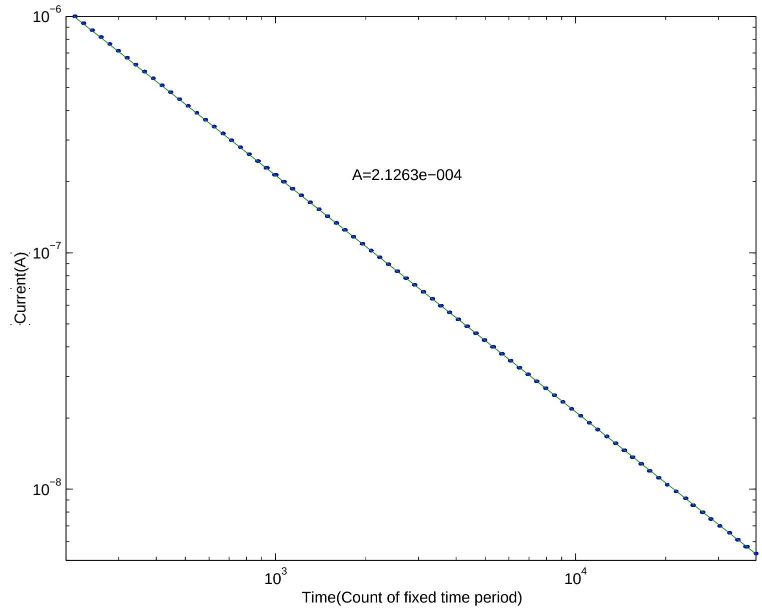


Figure 38. This plot was generated when the programming boards current measurement circuit was calibrated. Each dot is 100(all plotted, not averaged) simultaneous read values from the PIC at the current. To perform this calibration a Keithly Source Measure Unit(SMU) supplied the current controlled via MATLAB and the PIC was queried to obtain the time count. A was extracted to be 2.123E-4, currents were then placed into the circuit and using this extracted number the current was measured to within 1% accuracy. Only one current versus time measurement(point) is required to calibrate the setup, all other points obtained are optional but increase the accuracy of the calibration.

long to integrate this small current. To auto-range to a larger current can be done via an additional comparator, currents larger than the current capacitor can handle will keep the OPAMP output to be at or below the lower threshold tripping this additional comparator. The extraction of A from (38) is achieved by slope fitting a current sweep vs ΔT values, or in this case the PICs returned value of a count where each count is represented as a fixed amount of time obtained from the circuit. Figure 38 shows the fit to a particular setup as well as 100 current reads for the system plotted on the fit. It is clear that the system provides about 9-bit of accuracy across more that two orders of current magnitude. This system represent the current reading in a fashion similar to a floating-point representation providing the large range accuracy. This extraction takes into account the actual capacitance value including parasitics and eliminates the need to accurately know the known voltages required to compute ΔV. This extraction might be performed in future versions using a switch capacitor circuits to accurately generate the current needed during calibration on chip, instead of using the SMU. A counter on the PIC is used to monitor the pulse present on the output of the S-R latch and reports the value of time over the serial port when queried. The PIC counter has a bit limit, which is currently limiting the smallest current that can be read for a given capacitor. However, any measurement taking that long should be done with a different capacitor, increasing the measurement speed. Therefore, this will be one of the signals used to auto-range in a future revision.

The programming board contains a SPI controlled DAC which supplies the analog voltages needed during programming. This DAC also controls a voltage regulator to supply the V_{dd} to the Device Under Test(DUT) allowing the chip to safely be brought up to injection voltages only after drain and gate lines have; the current setup ramps all voltages up and down together to avoid un-intended injection when switching to programming mode. The SPI input to the DAC is supplied by the PIC which can be controlled via commands over the serial interface or also can execute precisely timed injection pulses. For the injection pulses used to program the floating gates, the drain voltage must be lowered for a specific amount of time and to a specific voltage. Injection information such as pulse width and pulse voltage can be loaded into the PIC before the pulse occurs achieving the short precise pulse widths, which were unattainable in the previous bench setups. Also, digital outputs on the PIC are sent through level shifts to the connector to the testing board. These digital outputs voltage follow the V_{dd} ramping into and out of programming mode. This allows selection of gate and drain lines through the on chip Multiplexor/decoder circuits as well as run/program modes of each cell. These digital outputs are controlled at the software level by the computer to allow flexibility in their use.

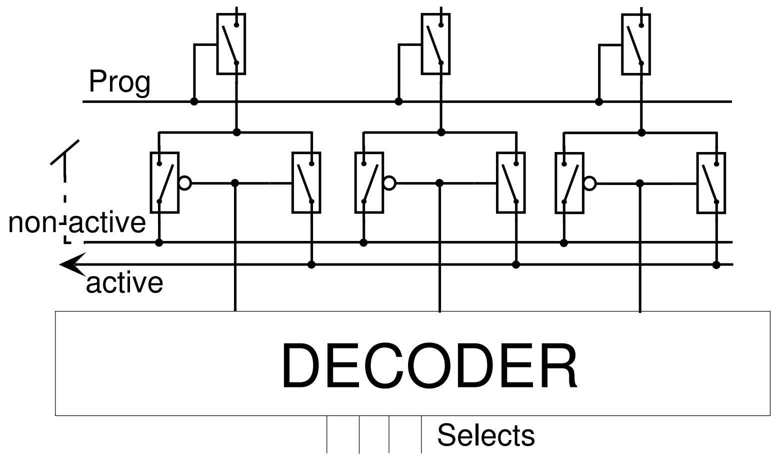


Figure 39. The use of these transmition gates and decoder allows the selection of a row or column in the programming array. The array line selected is passed out onto the active line while all other array lines are connected to the non-active line. The non-active line is used to disable the rows or columns by being brought to V_{dd} or an off voltage. The active line can be modulated to control the injection(programming) process.

4.8 Architecture Issues for Array and Non-Array Layout

There are a few architectural issues that must be examined when placing computing arrays into systems. When it comes to implementing the overly simplified scheme depicted in Figure 26 there are a few things that must be considered. First, the decoder/MUX structure shown in the bottom and side of this figure requires further explanation.

This structure resembles a modified multiplexor allowing one of the multiple columns to be selected; this is shown in Figure 39. When the array is placed in program mode the decoder selects which gate column is individually sent out on the active line while all other columns are connected to the non-active line. The non-active line is generally placed at V_{dd} which turns the connected columns gates off by providing no current through the floating gates, inhibiting injection when the drain lines in the row direction are modulated. In the row multiplexor/decoder the same occurs on the drain lines. It switches out the desired drain row onto a single line and places all the other drain lines onto an inactive line that is also generally placed at V_{dd} removing the possibility of a high electric field from V_{ds} inhibiting injection along these rows.

The multiplier structures that is used as the core of the ACAs needs relatively little

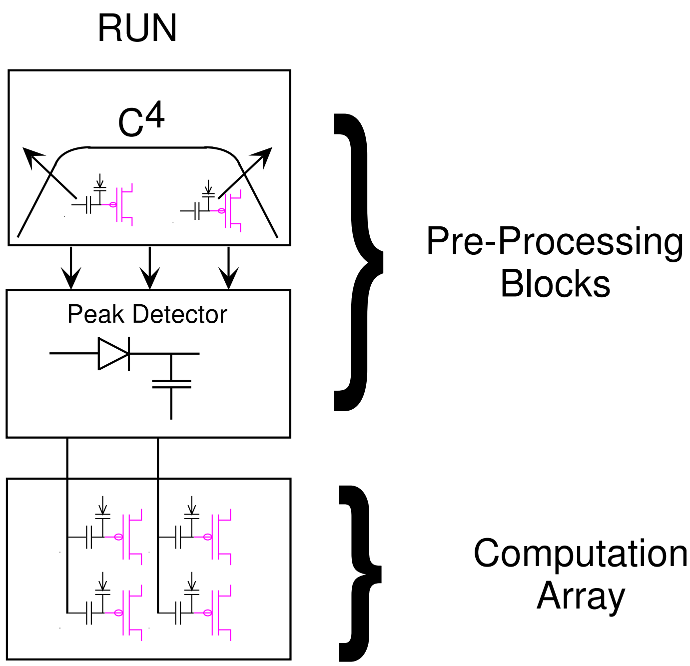


Figure 40. In run mode all block process the incoming information and broadcast the result downward in this model. The output of the previous block connects to the input of the next block. The output of the last block, the Computation Array, is sent out horizontally on the row's drain lines.

advance consideration when being placed into this programming array scheme. This is because the arrangement in the running system is identical to the arrangement needed during programming, with all the column's gates tied together and all the row's drains connected together. Placing floating gates into circuits like C^4's or peak detectors is a bit more challanging as it is preferable to program and address them like the multipliers in the array. Following are design considerations allowing even the non-multiplier cells to be programmed and allow individual testing of each column in the array. In addition, there are other forethoughts that should be considered if it is desirable to have some cells programmed while others are adapting.

Shown in Figure 40 is the block level of two pre-processing circuits which contain bias elements that need programmed. These circuits ultimately feed into the multiplier computing array. Therefore, the output of these circuits will connect to the gate columns in the computing array. In programming mode the columns should connect to the gates of

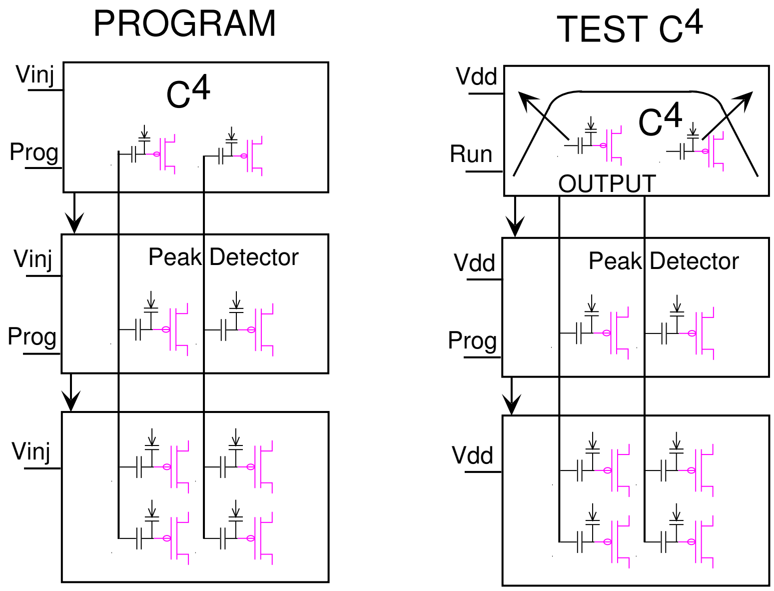


Figure 41. In this figure, Program and Test modes are shown. In Program mode all the gate line are accessible from the Multiplexor/Decoder which is placed below the computation array. This is accomplished by transmission gates in the preprocessing blocks. In Test mode certain blocks are left in Program mode, allowing the signal from the above block to pass through. With the addition of the logic in Figure 42 the signal from each column can be viewed.

every floating gate in the circuit including those in the pre-processing circuits. Through the use of transmission gates which have 2 states, programming and non-programming, we can access the floating gates in the circuits. The outputs of the pre-processing blocks then become inputs for column lines. Additionally the input and outputs of the pre-processing blocks must short together in order to allow propagation of the column line signals into the circuits above them. The top most circuit obviously does not need to send the signal above itself, which in the programmable filter was the C^4 cell.

Each pre-processing row should have it own program/run and V_{dd} pin. The need for each preprocessing block to have its own V_{dd} is so that blocks that are to adapt can be run at voltages allowing injection while not affecting the blocks already programmed that must be left at non-injection voltages. If the chip contains no adapting blocks all the pre-processing rows V_{dd}'s can be time together. The advantage of the individual program/run pins for each pre-processing blocks along with a modified gate-column decoder structure shown in Figure 42 allows each preprocessing block to be tested. One can then view the

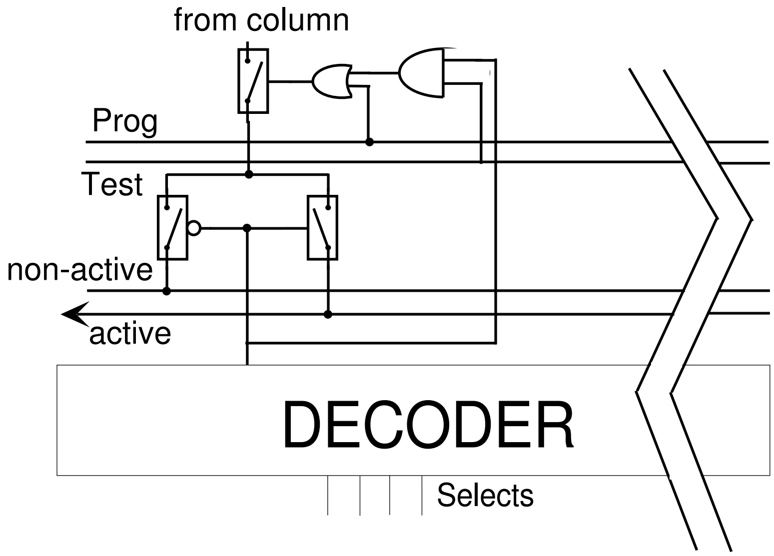


Figure 42. Using this modifier multiplexer/decoder circuit, which has some additional logic, allows each channel to be tested while running. When the Prog pin is asserted all channels from the column are connected to the decoder circuit. When the Prog pin is not asserted and the test pin is asserted only the selected column will be connected to the decoder circuit. All other columns are therefore unaffected.

output of only the C^4 pre-processing block by placing that block in run mode and every block below in program mode which allows the output to be fed all the way down and out the multiplexer. To see the effect of the next pre-processing block only requires moving that block from program to run mode. In large systems this dramatically improves the testability of the overall system by permitting the testing of each block.

Using these methods allows for fully programmable systems. Again the Programmable filter only had to account for attaching the C^4 cells floating-gate biases into the array. Systems such as the analog cepstrum processor, that is shown in Figure 43, include additional pre-processing blocks such as the peak detector that must allow the gate signal to pass through. The more pre-processing blocks involved the more time that will need to be spent in assuring the system is programmable and testable once the chip returns from fabrication. Some system may benefit by breaking up the system in half and duplicating the access blocks to the top or even potential several times throughout the system. Global decoder lines can be run up and down in a high level of metal to access the separated blocks which additional bits to select which device access block is currently being used.

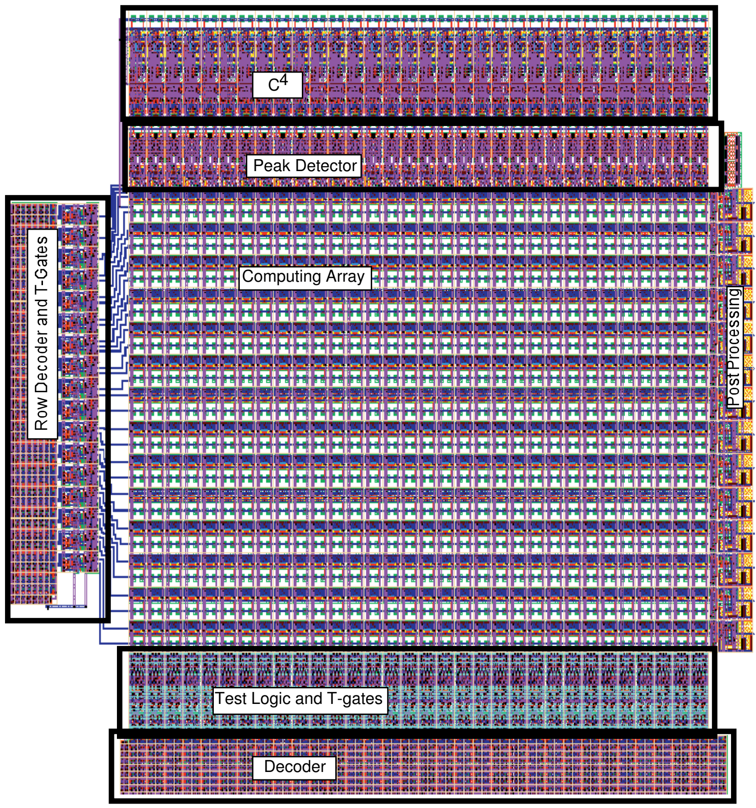


Figure 43. Analog cepstrum processor utilizing the Computing Array which was based on the Programming Filter core. The Programmable Filter is essentially the same except it does not have the added peak detector making it slightly easier to implement.

CHAPTER 5

ROW-PARALLEL PROGRAMMING OF FLOATING-GATE ELEMENTS

5.1 Motivation

The use of floating-gates in analog systems to perform computations on signals has expanded over the last several years [29]. Furthermore, these system are growing in complexity, requiring the use of an ever increasing number of Analog Floating-Gate Elements (AFGE). Currently, AFGEs are generally programmed using custom off-chip programming circuits and algorithms [53]. However, the time required to program many elements to a desired value still impedes the wide-spread use of AFGEs in analog circuits. Also, off chip programming consumes many pins on a chip, reduces programming accuracy, and requires specially designed boards be present whenever a AFGEs value needs to be changed. This has motivated research into on-chip programming to eventually provide digital only signaling for programming AFGEs. Presented in this section are the circuits used to program elements on-chip and in a parallel fashion, further decreasing the time needed to program AFGE elements.

5.2 Row-parallel Scheme

Altering the stored charge on AFGEs can be carried out using UV photo injection, electron tunneling, or hot-electron injection. For AFGE arrays, we have chosen hot electron injection, also referred to as CHannel Initiated Secondary ELectron (CHISEL) Injection in some literature, to program the deivce. This assumes we will initially start from a floating-gate voltage below the desired value to program to. To erase the AFGE array, the floating-gate voltages are reduced using electron tunneling to flash erase (erase all devices at once) the array. The rationale for program and erase choices is methodically explained in previous chapters on off-chip programming algorithms [36, 53] but mainly is due to large device

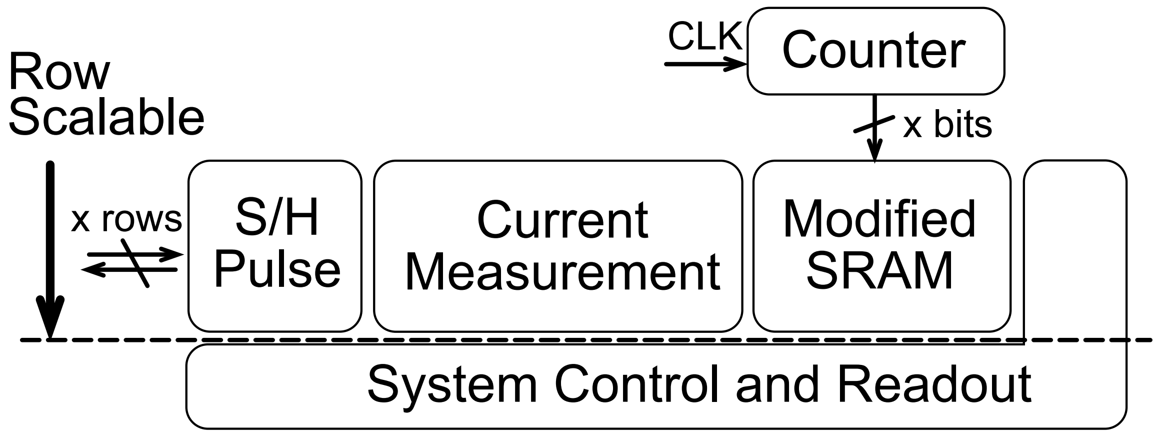


Figure 44. This is a block level diagram of the on-chip scheme and not drawn to scale. In order to provide row-parallel readings and keep the pins count minimized modified SRAM cells are used to store the current measurements. They can then be read off-chip using serial methods such as SPI again keeping the pins count requirement to a minimum. All the circuits needed to program the array are located on-chip providing quick and accurate programming for the ACAs.

mismatch in the tunneling junctions used in AFGEs. The on-chip row-parallel structure where designed to work with a constant-time varying-drain voltage-pulses needed to program AFGE based on the programming algorithm described previously [53].

The current measurement structure was built to source a voltage and read the current. This is important for programming when devices are programmed to a given current for a V_{ds} other than the supply voltage. Also, this permits the reading of currents in the array after the array has been ramped up to injection voltages. This is useful for some of the algorithms that are use to bring scattered devices into range. The design also maintains the correct V_{ds}, whether the current measurement circuit is in integrate(read), pause or reset mode. The design allows the current of all rows' AFGEs in a given column to be read simultaneously with the same applied gate voltage. This is not considered a drawback as the V_g diffrence from the desired user V_g can be calculated out in the desired current by knowing the effective device kappa. Further, in practice only one gate voltage is used when calculating what currents are needed. In many circuits like the multiplier [37] the importances is to program a difference in floating-gate voltages for the computation and can be done at any gate voltage. The additional logic needed to permit varying the gate

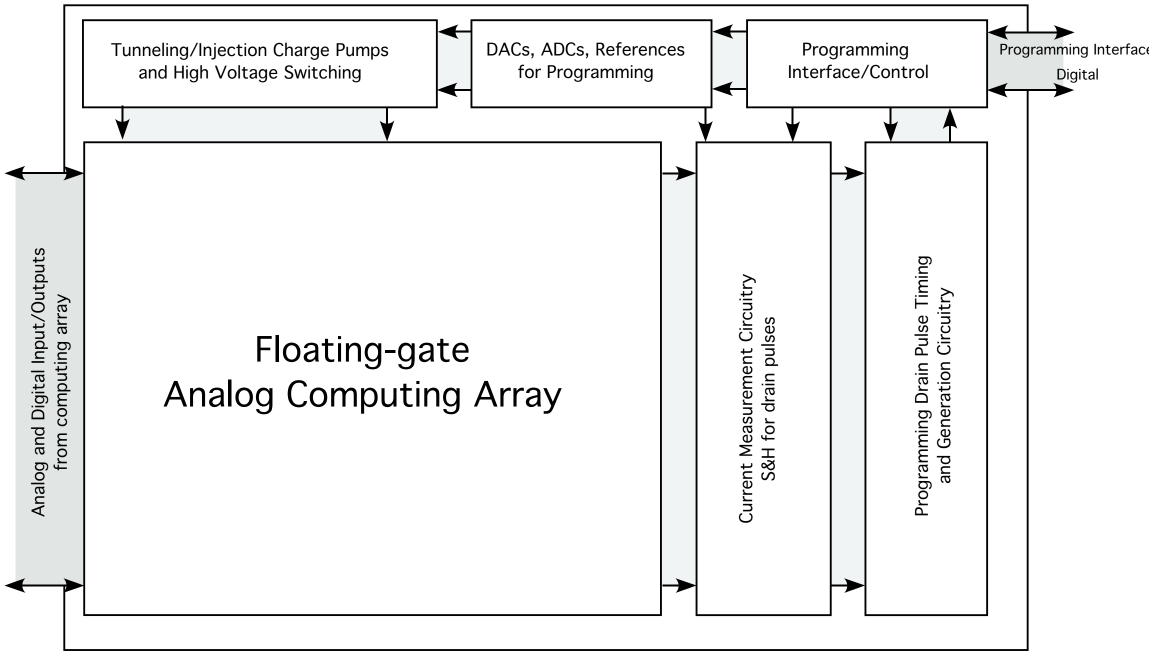


Figure 45. This shows the high level diagram of the the peripheral circuits needed for programming the Analog Computing Arrays. The goal is to provide a system that is unnoticeable to the user. To accomplish this: The supply voltage needs to be no greater that the operating voltage. The programming of the array in performed by loading target values over a digital interface. The programming of these values occurs on-chip without a detailed understanding by the user.

voltages along rows during a simultaneous read is more costly than the benefits gained for most of the current AFGE systems.

The current measurement circuit, shown in Fig. 46 operates in one of three configurations throughout the read cycle. The first is a reset mode; in this mode the capacitor used to integrate the current is reset by placing the terminals between V_{start} and V_{drain}. This places the capacitor charge in a state where the integrator output will be V_{start} when integration begins. All rows in the current measurement block are reset at the same time once the *reset* signal is asserted. Once the reset mode is complete, the circuit is placed in a pause mode waiting for the *read* signal to go high. After the *read* signal is asserted all rows begin to integrate the current coming from each row in the array on the feedback capacitor. As the current is integrated on the capacitor the output voltage of the op-amp that has the capacitor in the feedback begins to drop. Once the output voltage drops below V_{stop} that rows comparator trips and takes the row out of integrate mode and places it back in pause mode. The completion of the integration is passed to the SRAM block and the current counter value is locked into the SRAM for that row. This counter value provides us with the time (δT) that the circuit integrated between the two voltages (δV) using the feedback capacitor (C).

This circuit provides a clean way to read the currents in each row at the same time. Also, the circuit stops the integration once the integration output reaches V_{stop} and waits for a reset. The voltage on the output of the integrator will remain fixed for some short amount of time. Because of this, the comparators voltage can be swept over time and the counter used to read out the δV discretized that was integrated over the fixed δT. This is handy for measuring devices that would otherwise be out of range for the capacitor and counter clock used. The last bit of the counter can be used to mark the transition from a time-measured to voltage-measured mode and on the transition of this bit the integration in placed in pause mode. Then during each clock transition into the counter the voltage V_{stop} can be increased by some amount and the counter will record the voltage where the comparator tripped for the row. The δT then used for the δV sweep is just $\frac{1}{T} * 2^{(bits-1)}$ and provides a way to measure the δV from the inital integration. Any counter value with the MSB as a zero will indicate a row where the time for the initial δV was integrated over and a one will indicate a row where the voltage integrated over was measured.

After current measurement have been made an algorithm is used to calculate a V_{ds} to apply to each row. This is currently being computed off-chip from the measurements obtained on-chip. This algorithm is being converted to a format that will allow it to be implemented completely on-chip in the next revision. Moving the algorithm on-chip will decrease the time needed to program and array and further minimize the data needed to be moved on/off chip. Once the V_{ds} is calculated, the voltages are loaded into sample and holds in each row, allowing the pulse voltage for each row to be loaded independently. All the voltages are then pulsed simultaneously for the same amount of time providing the correct V_{ds} pulse for each row. After the pulse is completed the drain voltage is restored back to the global drain voltage supplied.

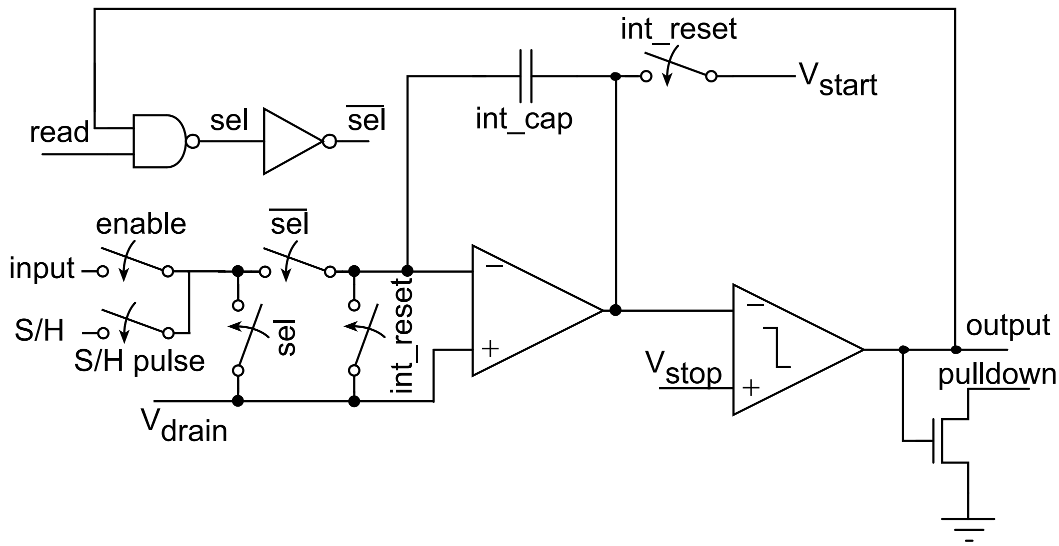


Figure 46. The row-parallel current measurement circuit allows a voltage to be sourced while the current is read. This is accomplished using a switch-capacitor like integrating op-amp. The circuit will integrate using I=$C\frac{\delta V}{\delta T}$, where δV is $V_{start} - V_{stop}$, C is the integration capacitor, and δT is measured using the output of the comparator and an on-chip counter. The results of each row are stored in a modified SRAM cell and later read out from off-chip.

5.2.1 SRAM Block

The SRAM used to store the counter's value in for each row is a modified SRAM cell. This permits separate control for loading and readout and separate inputs for loading and reading. This cell can be read simultaneously while it is loaded. Figure 47 shows the schematic representation of the SRAM cell. When the SRAM is in load mode one of the inverters in lifted from ground and not able to pull the input of the SRAM to ground. This is desired as high voltages applied at the input do not pass completely through the single nFET input providing only a mid-rail input voltage. This voltage reduction using a single pass device is evident in the cells readout as seen in Fig. **??** where the high voltages are attenuated.

The SRAM cell has been measured to hold the output of the counters running at 10Mhz. This was tested by placing the integrator block in reset mode and applying V_{start} below V_{stop} to lock in the counter value. When V_{start} is brought above V_{stop} the SRAM load in the current counter values. Figure 47 shows the ability of the SRAM to hold a one or zero

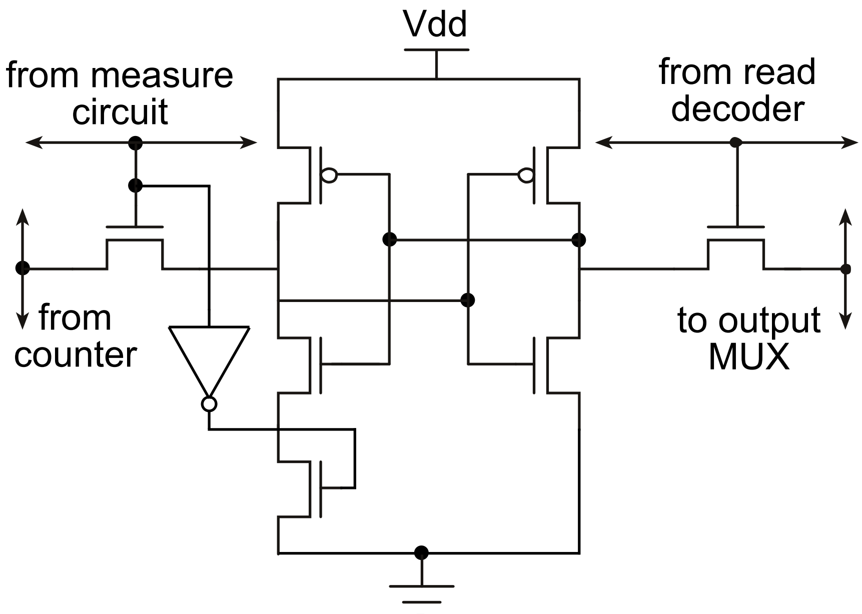


Figure 47. To store the time value supplied by a standard counter a modified SRAM cell is used. This cell allows reading an writing from separate inputs and does not require a differential signal. The counter bit is fed into the Write Bit Line until the output from the comparator drops shutting off the Write Word Line. This locks in the current counter value in the SRAM row once the integration is complete. To read the row, row decoder selects the Read Word Line and allow another set of decoder-muxes to send out each bit along the row.

while the clock input is still applied after being loaded.

5.2.2 Sample and Hold

A sample and hold is needed to provide drain pulses to the array simultaneously. After the rows current is measured using the on-chip row convertors it is compared against the target values. The sample and holds can then loaded from an on-chip or off-chip Digital to Analog Convertor (DAC) according to a lookup table. The sample and hold needs to be relatively small in it's footprint as it is replicated for each row measured. For this reason a simple 5 transistor OTA with a 200fF sampling cap was used in the design. This section shows the measured performance of this simple sample and hold.

Figure **??** shows the input vs. output characteristic of the sample and hold. The sample and hold tracks well for voltages in the range of 0.8V to about 3.1V for a supply voltage of 3.3V. This sample and hold has a gain error of about 48.5mV/V of the input range.

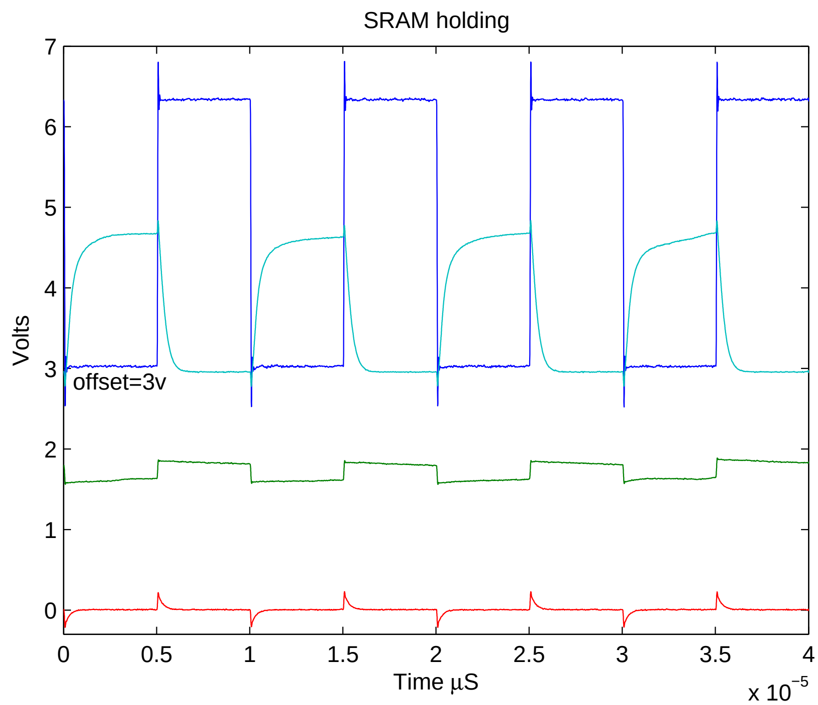


Figure 48. A single SRAM has been shown to hold it value. A 1MHz input signal was applied to a single SRAM cell. Shown in the figure is the ability to hold a zero and and one. The cell uses a single nFET for readout and therefore the high values stored in the SRAM as V_{dd} appear at the output around 1.5v

There appears to be a input offset of about 0.07V as measure from the chips sample and hold input to the chips sample and hold output. This is a result of the small sized input transistors mismatch or voltage drop in the sample and holds input switch. This however may be acceptable in this system, but would require a slightly longer time on an industrial tester. The injection table on-chip could take into account the off-sets in the sample and hold when initially calibrated / tested when back from fabrication. To provide this testability a switch needs to be added to the output of the sample and hold of each row and indexed off the row decoder.

5.2.3 On-chip Measurement Counter

The ripple counter on chip was tested experimentally using up to a 30MHz input signal. This was supplied using a external function generator that could supply signals only up to 30MHz. The input clock, output of the first division and the output of the 6th division

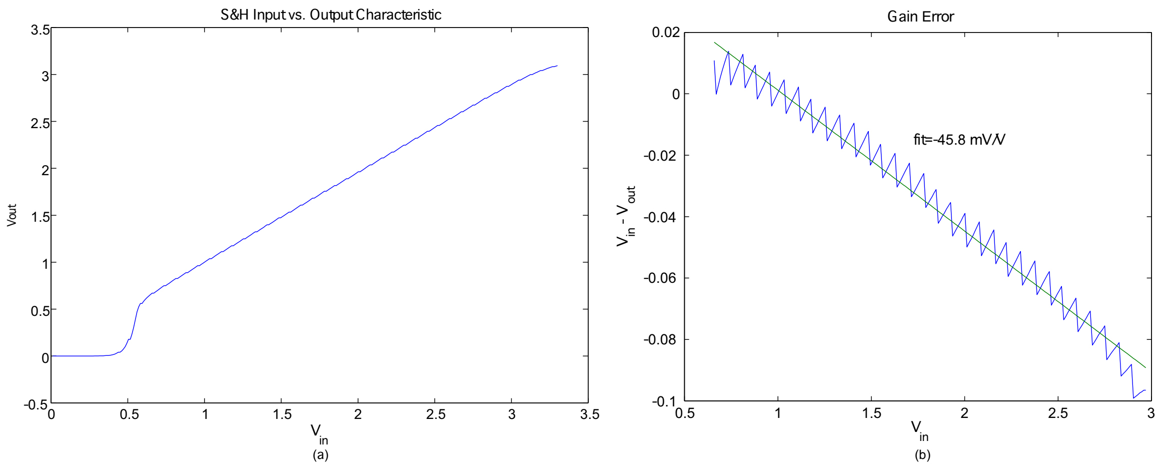


Figure 49. (a)The sample and hold tracks voltages well between the ranges of 0.6V to 3.2V for a 3.3V supply. This is the expected operation for a 5 transistor OTA connected in unity gain feedback. (b) The gain error of this sample and hold was extracted to be -45.8mV/V. A slight non-linearity can be observed

are shown in Fig. 53. The output of each counter bit is buffered internally using a buffer-inverter. This inverter allowed the counter to function up to 30MHz even when the output of the first division was severally attenuated as measured off-chip. The load of the protection pad, proto-board, and leads to the measurement equipment was the source of the load observed at the output of the first division. The sixth division frequency was measured to be 469.2kHz, close to the expected value of 468.7kHz.

The resetable counter used in the design was a ripple counter. Therefore, the carry must ripple through the system and propagate to the MSB before it is updated. This delay was measured by obtaining the output transitions from division of the counter. The delay from the first transition to each subsequent division transition was measured and plot in Fig. 54. A fit of the points resulted in an experimentally measured propagation delay for the ripple in this counter design to be 4.275ns per bit.

5.3 Resolution and Mismatch Issues

Programming AFGEs to a precise analog current will be subject to some maximum accuracy. This accuracy can arise from the resolution of the measurement block or from

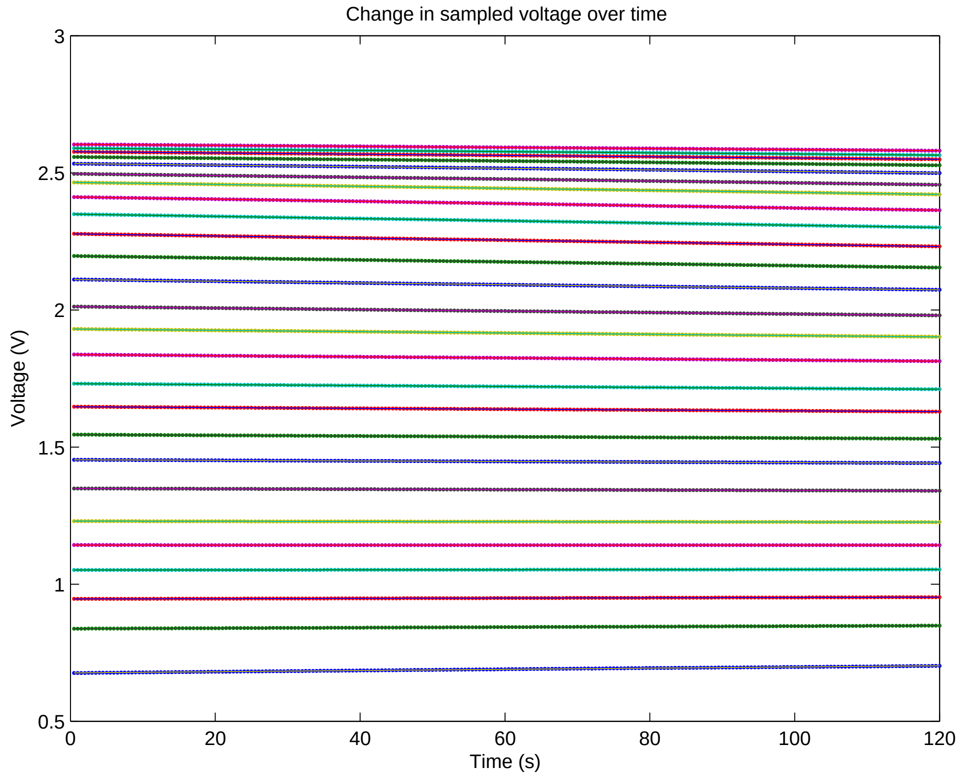


Figure 50. Decay over time for the starting voltages. This data used to get the decay voltage plot

mismatch in any of the programming circuitry. The resolution of the current measurement circuit in the on-chip AFGE programmer is not fixed once fabricated. The measurement circuit works by integrating the current to be measured on a capacitor. The voltage on the output of the capacitor changes from one fixed voltage to another fixed voltage over the integration (δV). This is governed by

$$I = \frac{C * \delta V}{\delta T} \tag{39}$$

where the current I is being measured by a δT.

The resolution available to measure I is set by several parameters. The first is the minimum δT that can be measured. This is directly related to the clock frequency supplied to the counter as the LSB will switch at this clock frequency. The clocking frequency is however scaled by two other terms; δV and C. The integrating capacitor C is fixed during

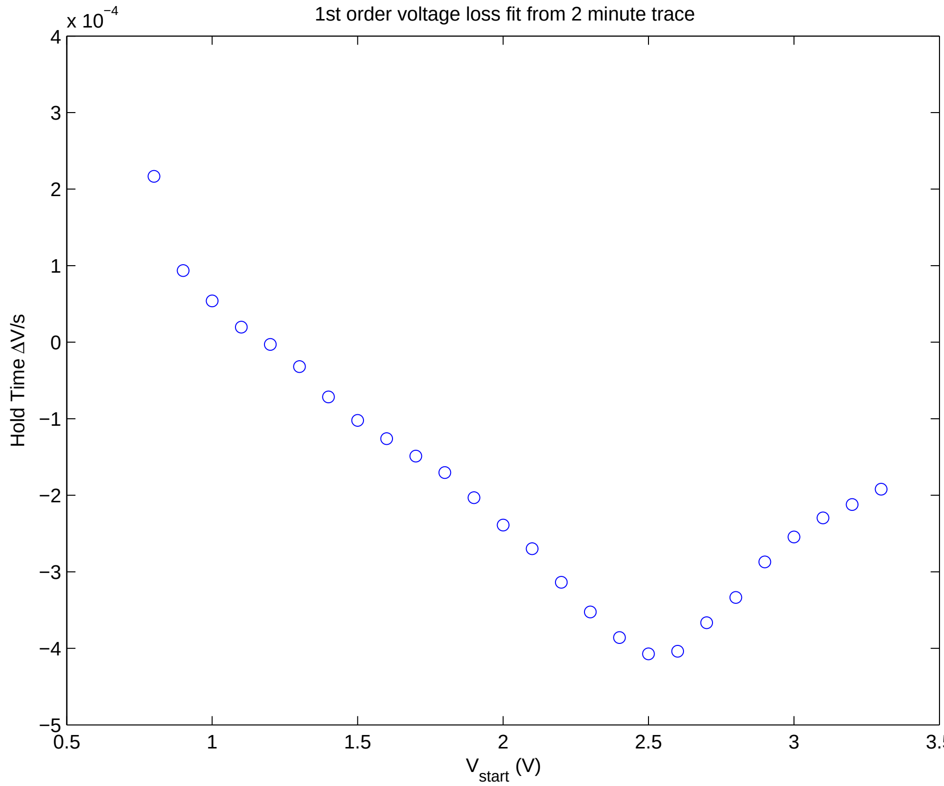


Figure 51. The decay rate of the sample voltages vs. the voltage stored on the Sample and Hold is shown. The decay is at worst case 0.4mV/S or 12-bits of resolution with hold times of one second.

fabrication and multiple capacitors can be fabricated for each row, then switched in to change the currents that the circuit is able to integrate over a set time window. Increasing the capacitor slows down the integration for a fixed current, allowing the resolution for that current to be increased for a fixed clock frequency. However, by increasing the capacitor with all other parameters fixed the time to integrate the current is now longer. It is apparent that by changing the capacitor we can trade off time for resolution. The δV which is the voltage over which the current is integrated can be changed after fabrication as well. This can be changed and affects the resolution vs. time tradeoff in the same fashion as the capacitor.

Mismatch in any of the rows circuits can obviously alter the counter value (current) read from each row using the same conditions. Mismatch in capacitances used for integration and transistors in the comparator are such places for this mismatch. However, the use of

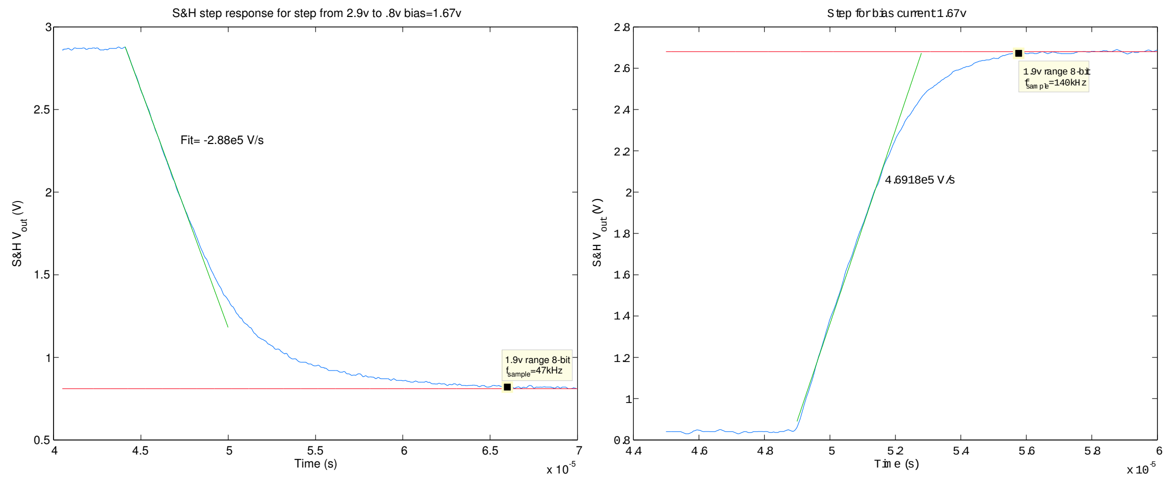


Figure 52. The minimal conversion time for the sample and hold used is 45kHz for a range of 2V. This conversion time will determine the amount of time needed to load all the rows sample and hold. For example to load 100 rows would take 2ms. The first sample and hold loaded would change by less than 1 μV after being loaded.

AFGE in arrays can tolerate to some degree or in some systems are even not affect by row mismatch. For the multiplier, which makes up about 80% of all operations in many of the ACA systems, only the difference in the neighbor affects the calculation, where absolute values alter bias currents only that do affect the computation. Each row can therefore measure the same current slightly off and the computation accuracy will be preserved. AFGE currents that require good matching to each other should be placed along the same row due to these row to row mismatches. Therefore, while the mismatch exists, correct choices in the design of computation systems can minimize the effects of the programming mismatch. In future revisions of the programming circuitry correction registers on-chip could be used to correct such offsets. During initial test, known currents can be applied and the offsets stored on-chip for each row providing a way to later remove these offset errors.

5.4 Simulation

The row-parallel structure, consisting of the sample and hold; integrator; counter; and modified SRAM, has been simulated in Cadence and demonstrated to work. Simulation shows that the dynamic counter used can operate at frequencies up to 10MHz in the AMI 0.5μm

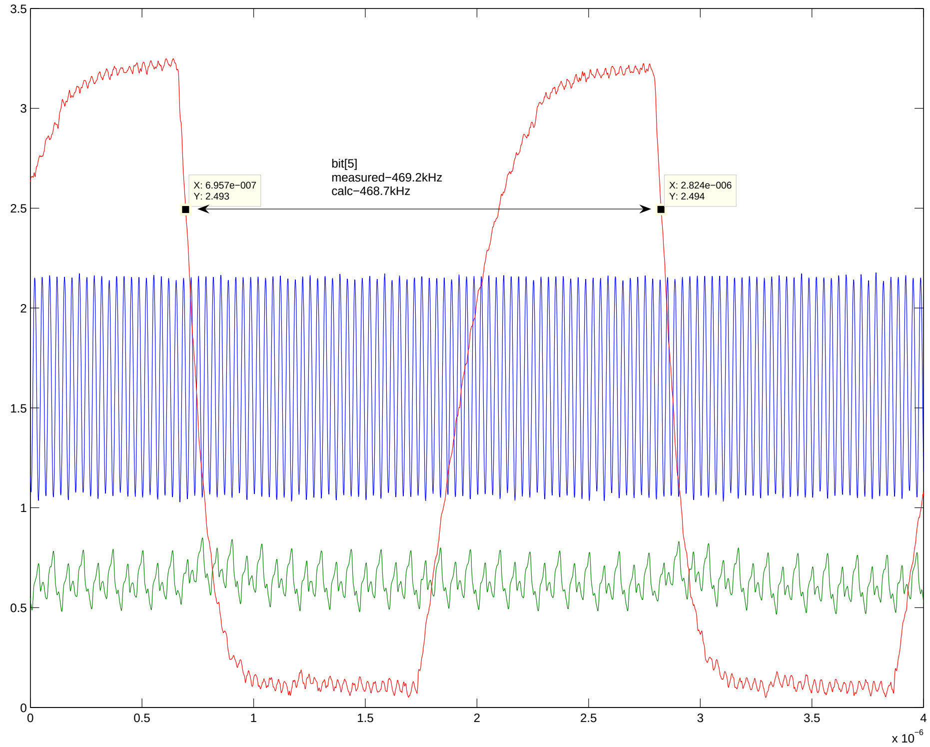


Figure 53. The ripple counter on chip was tested using up to a 30MHz input signal. The input clock, output of the first division and the 6th division are shown. The sixth division frequency was measured to be 469.2kHz, close to the expected value of 468.7kHz.

process. If a 10fF capacitor and a δV of 1v was used to integrate the signal this would give an upper limit of 100nA. That is if only the first bit of the memory is latched the 10fF capacitor integrated over 1V in 100ns and according to (39) that would be a current of 100nA. It can be seen in this simulation as discussed earlier that two things will limit the maximum current that can be read from the structure. The size of the capacitor; where a larger capacitor will permit the reading of larger voltages and the speed at which the counters can operate; where a fast counter permits larger currents for a given sized capacitor. Because this cell is arrayed it will be advantageous to limit the cell size as much as possible. With a capacitor of size 100fF we can measure up to 1uA with the LSB change.

If we use a 16-bit counter to measure the current in the smallest readable current can

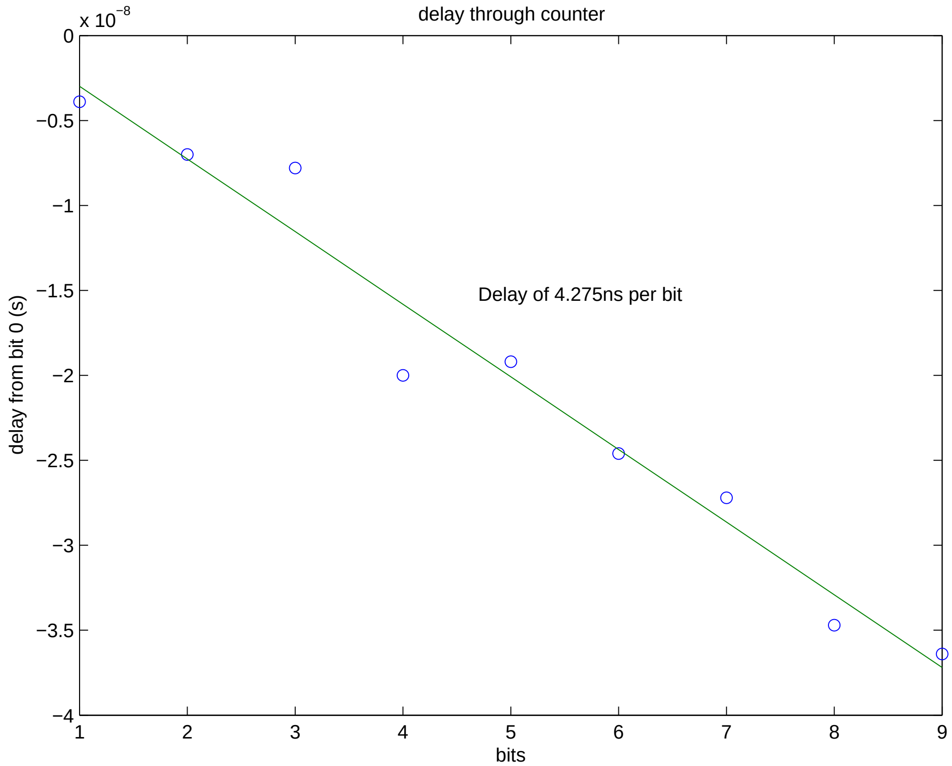


Figure 54. The resetable counter used in the system was a simple ripply counter. The propagation delay for the ripple in this counter design was experimentally measured to be 4.275ns per bit.

take up to 6.5ms to be read. This would be the case if all bits in the counter are high and the current read would be 1.52pA for the 10fF capacitor. Obviously the faster the counter the more bits can be used to provide full range in a set amount of time; where more bits increases the read time along with the range for smaller currents. Shown in Fig. 55 is the integration and SRAM values for a 1fF capacitor plus parasitics integrating 20nA over 0.6V. The integration completes in only 200ns as seen by the SRAM values. This chip is back from fabrication and is currently undergoing testing. The chip will be tested with a FGPA simulating the on-chip controller to rapidly program an array of float-gate elements.

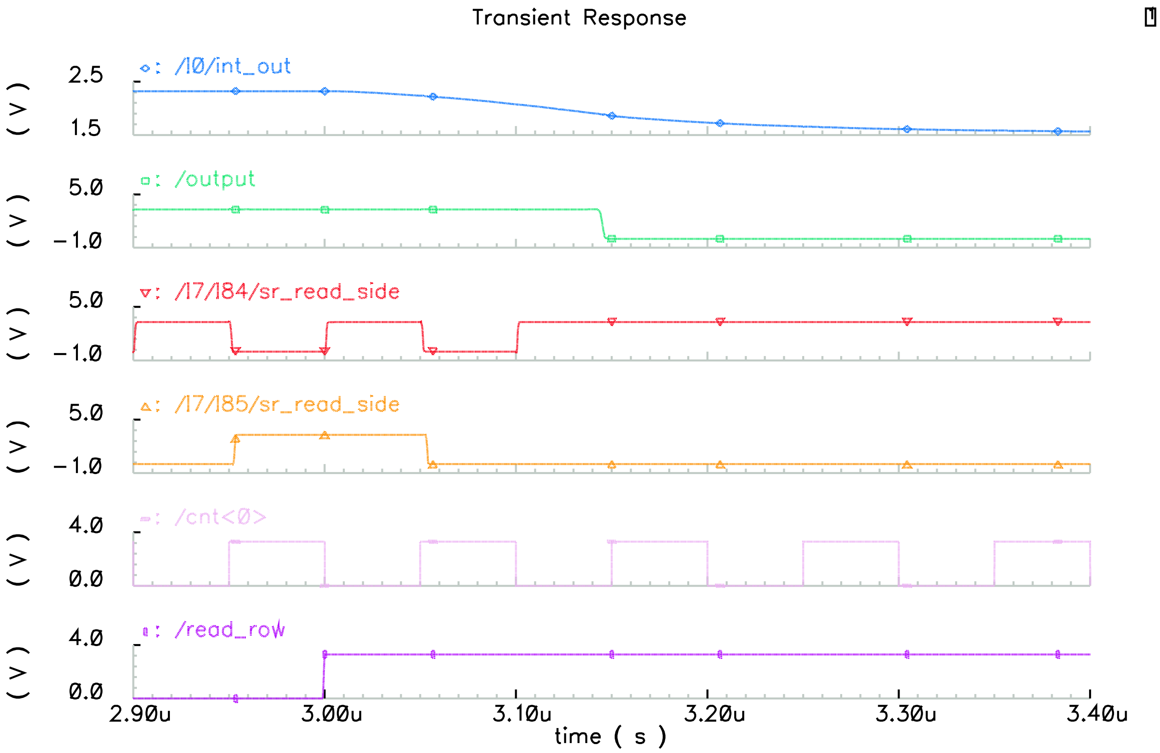


Figure 55. This is the integration and SRAM values for a 1fF capacitor plus parasitics integrating 20nA over 0.6V. The integration completes in only 200ns as seen by the SRAM values.

5.5 Charge Pumps

If the ACA systems are to be self-contained and completely transparent to the user, the high-voltages needed for programming should be generated from the standard supply voltage. In order to accomplish this switch-regulators or charge-pump/voltage-doublers will be needed a part of the on-chip solution. Both circuits provide DC-DC conversion using a high switching frequency clock. The main difference between the two circuits is the element used to store energy during the switching cycles. Switching regulators use an inductor to store the energy whereas charge pumps use capacitors. In the case of the switching regulator, energy is stored by the change in current in the inductor whereas in the charge pump, energy is stored in the capacitors by the change in voltage. Capacitive energy storage circuits have the advantage of being able to be completely integrated on chip when the power requirements are relatively small thereby reducing cost, eliminating an external supply, reducing pin count and hence using less board space. The work presented in this section is a collaborative venture with another colleague in the ICE lab.

5.5.1 Charge-pump Direction

After the need for a chargepump was realized, both switching carge-pumps(voltage doublers) and Dickson charge pumps were examined as possible solutions. First, the switching charge pumps were evaluated. This charge pump operates by charging-up capacitors in parallel, the reconnect them in series to produce a greater voltage than the one supplied. It can be though of as connecting batteries in series to obtain a higher supply voltage when each, individually, have a lower voltage. These charge pump structures [50] are based on switched-capacitor techniques [1] This structure is highly efficient and lacks the leakage currents and threshold-drops that plague the Dickson charge pumps. However, the switching charge pumps appeared to have a major draw-back; in that the switches used to provide the series-to-parallel conversion required the same high-voltage that you are attempting to generate. There are solutions to this drawback [4] and require additional driver circuits to control the charge pumps. The information on these driver blocks is limited mostly likely due to intellectual property concerns. Also the need to non-overlapping clocks is a requirement for many of these types of pumping structures.

The second structure that was entertained as a solution to supplied injection and tunneling voltages was the Dickson charge pump [8]. The Dickson charge pump uses two clock signal in anti-phase of each other, alternately applied to each stage, to pump charge through diodes that operate as a self-timed switch. The schematic diagram of a CMOS Dickson charge pump is shown in Fig. 56. The operation of the clock is responsible for the boosting operation. For example at point A, the voltage is boosted to 6V when the clock is pulsed high. This occurs because the voltage is set at 3V by the voltage source input and it is immediately elevated to 6V when the clock toggles high since the voltage across a capacitor cannot change instantaneously. Therefore the output capacitor is charged to 6V

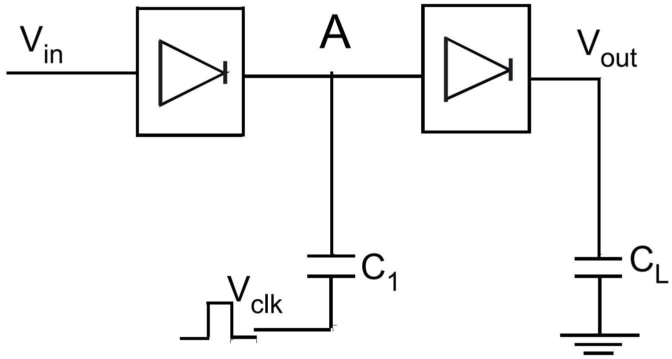


Figure 56. This is a schematic representation of a charge pump. In our work we will be designing a Dickson charge pump. Our objective is to find the optimum rectifier structure which will yield the highest output voltage with the best efficiency. The value of the capacitor used in the designs presented in this paper equals 1pF.

since this stage is pulled to ground by the clock which operates in antiphase. Thus each successive stage is increased by the magnitude of the clock, i.e., 3V. The output voltage is given as:

$$V_{out} = V_{in} - V_d + n * [V_{\phi}^{i} - V_d - V_l] \tag{40}$$

where

V_{in} is the input voltage to the circuit;

V_{ϕ}^{i} is the voltage swing at each node i due to capacitive coupling of the clock;

V_d is the voltage drop across each rectifier;

V_l is the charging and discharging of the capacitors when the charge pump is sourcing current.

5.5.2 Dickson Chargepump Rectifying Element

One of the elements used in the Dickson charge pump is the rectifying element. This element while drawn as a simple diode must be implemented on-chip using structures available to the designer in a standard CMOS process. After thinking through the MOSIS CMOS process available, six rectifying structures were identified. These six, shown in Fig. 57, are: diode connected NMOS (DCN), high voltage NMOS (HV), body controlled diode

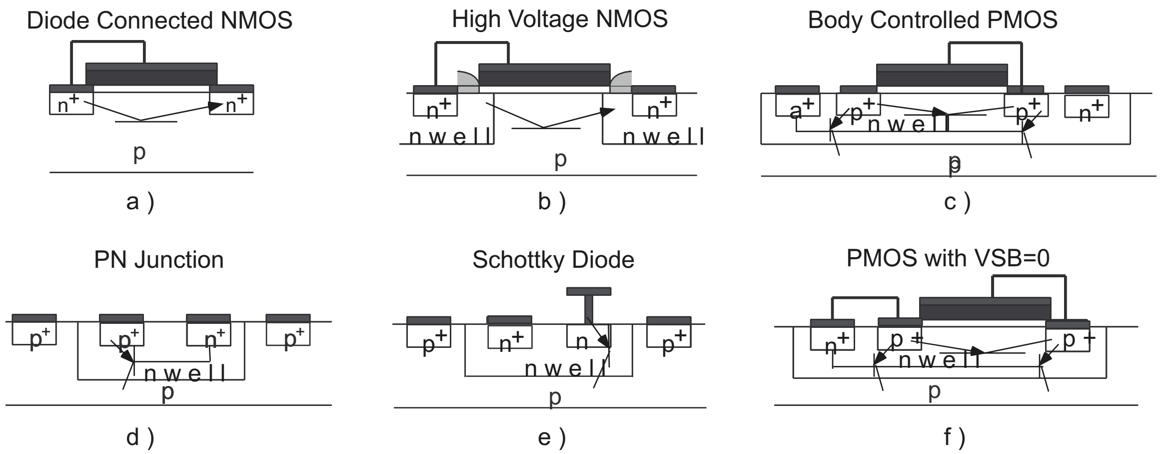


Figure 57. Shown in this figure are the 6 possible rectifier structures designed in a standard .6um CMOS process.
a) Diode connected NMOS (DCN) transistor W=6um and L=.6um. b) High voltage NMOS transistor (HV), W=6.75um. & L=.6um. c) Body controlled PMOS transistor (BCP),W=6um and L=.6um. d) PN Junction (PN), diffusion dimensions 5.4um x 2.4um. e) The Schottky diode (SCH),diffusion dimensions 2.4um x 3.9um. f) PMOS diode connected transistor with VSB=0 (VSB=0), W=6um & L=.6um.

connected PMOS transistor (BCP), PN junction (PN), Schottky diode (SCH) and PMOS diode connected transistor with source to body voltage equal to zero (VSB=0). All of these devices were fabricated in a standard .6um CMOS double poly process. In order to layout some of these rectifiers in standard CMOS some of the DRC violations were ignored.

5.5.3 IV Curves

Shown in Fig. 58 are the IV curves for both forward and reverse bias regions for five of the devices. The high voltage device was omitted because it was found to be inoperative, probably due to an inadequately long channel length which most likely resulted in punch through. The channel length was later doubled and as such was successfully used in subsequent multi-stage charge pump designs [31]. In the reverse bias regions it is seen that all the rectifying structures except for the DCN rectifier break down with a reverse bias of 1V or less. The IV curves for the BCP device were measured for body voltages of 1V (Bdy=1), 3V (Bdy=3) and 5V (Bdy=5) and its breakdown voltage also occurred when the reverse bias was approximately 1V or less. In the forward bias case, the Schottky diode

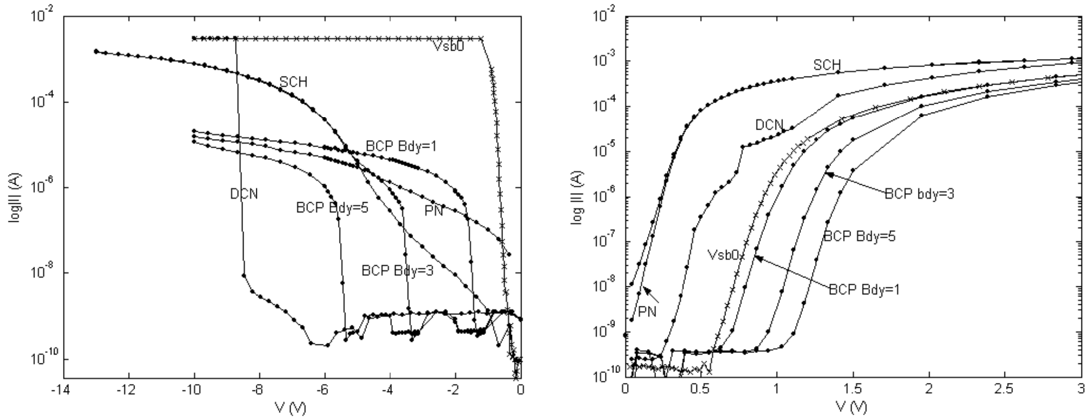


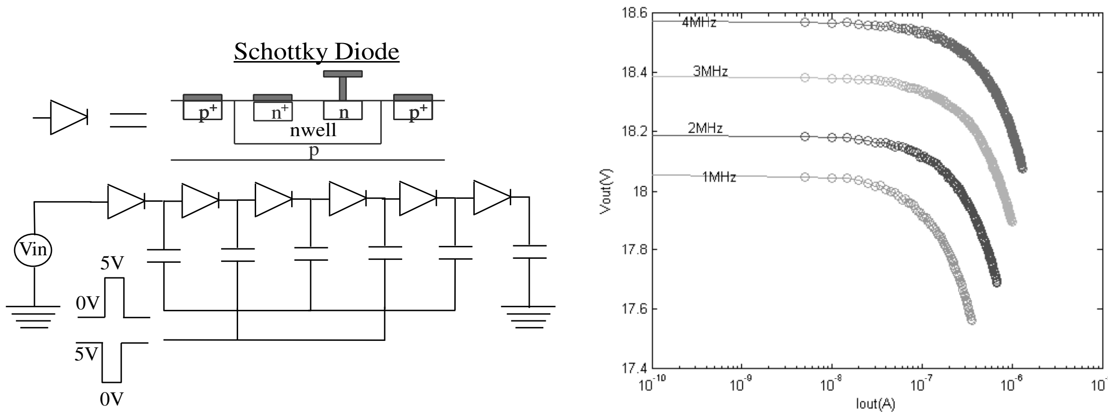
Figure 58. Fabrication results from .6u standard CMOS process indicating the reverse and forward bias characteristics of the 6 different rectifiers.
Legend: **DC ⇒ Diode Connected NMOS transistor; BCP Bdy=5 ⇒ Body controlled PMOS transistor with Vsb = 5V;**
BCP Bdy=3 ⇒ Body controlled PMOS transistor with Vsb = 3V; BCP Bdy=1 ⇒ Body controlled PMOS transistor with Vsb = 1V;
PN ⇒ PN junction; SCH ⇒ Schottky diode; Vsb0 ⇒ PMOS diode connected transistor with VSB=0.

structure is the most current prolific device and has the highest initial value of current.

5.5.4 Pump Design

For Analog Computing Arrays, two high-voltages will need to be generated from the operating supply. The first is the tunneling voltage. To remove charge the floating-gate, a special tunneling junction is used. For the $0.5\mu m$ process a voltage of approximately 15V is needed to tunnel, with tunneling rapidly occurring around 17v. Therefore, this charge pumps need to be able to supply a high voltage. The current requirements for the output of the charge pump for tunneling are minimal per device. Tunneling junctions are extreamly efficient in that all the charge supplied to the tunneling junction is passed through and onto the gate.

It is seen in Fig. 59(b), that in general for higher frequencies, the output voltage is increased. This corresponds well with the output voltage v.s. frequency characteristics shown in Fig. 61(b). The clock signals were generated off chip to simplify testing of initial structures. Currently in fabrication is a design with the clock and inverted clock on-chip



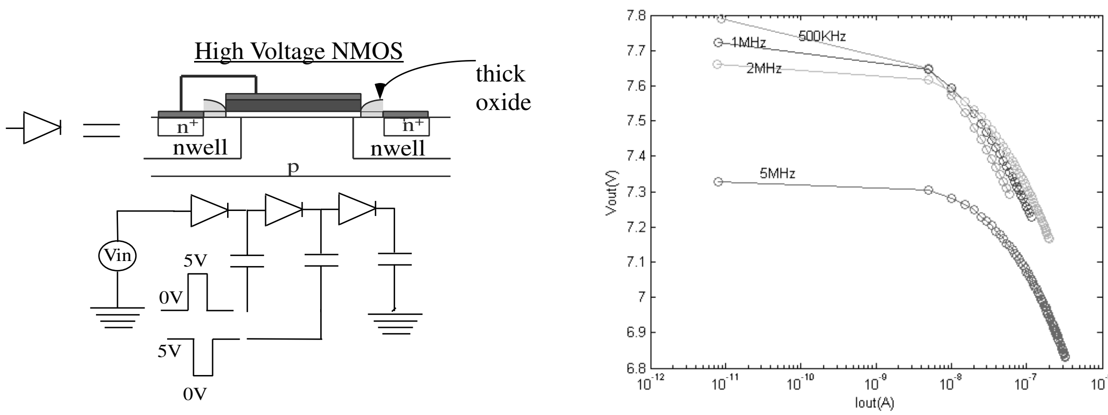
a) b)

Figure 59. Illustration of the Schottky charge pump used for tunneling and its characterization from fabrication results in a standard 0.5μm CMOS process. (a) The Schottky charge pump configuration to effect tunneling. (b) Current loading characteristics for various frequencies. The output voltage of approximately 18V will be sufficient for tunneling in a 0.5μm CMOS process. This configuration shows ample sourcing capability to drive an array of floating gate circuits sinking a total of 1nA where each floating gate transistor sinks 1fA.

to prevent signal degradation. This charge pump will be used to control a floating gate array where each transistor draws approximately 1fA for a total of 1nA. Shown in Fig. 61(a) is the transient output of the Schottky charge pump boosting the output voltage from approximately 4V to 18V. The output exhibits a charging with no dramatic overshoot spikes that could potential damage the tunneling oxide.

The second is the injection voltage. The requirement for this charge pump will be moderate pumping voltage(6.5V from a 3.3-5V supply) but a need for an adequate supply current. Injection requires a channel current to be present and therefore is not 100% efficient as in the tunneling case. While some seem to indicate that large drain currents are needed [49], injection actually occurs more efficiently with sub-threshold drain currents. The channel current generally needed for injection programming of ACA is no more than 10nA.

Shown in Fig. 60(b) are the current loading characteristics for a 3 stage Dickson charge pump using high voltage transistors. Conforming with the results shown in Fig. 61(b) there is a tendency for the output voltage to decrease with frequency. However once again at



a) b)

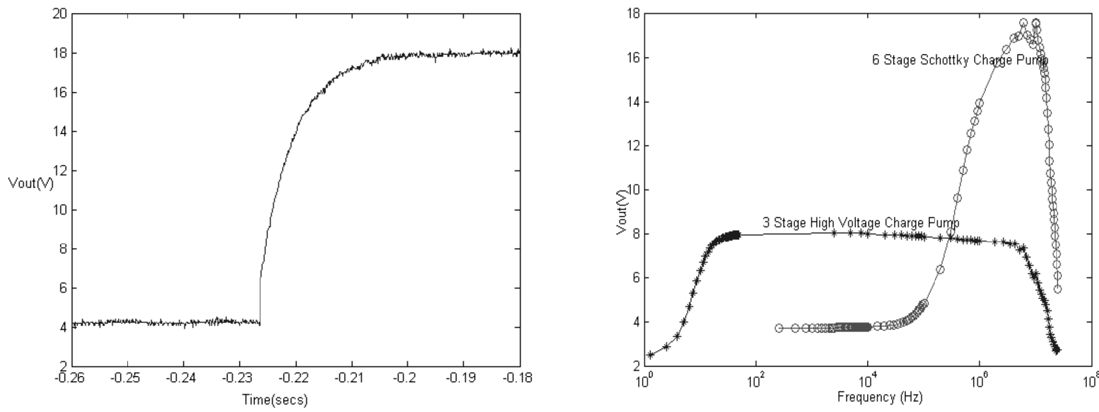
Figure 60. Illustration of the high-voltage charge pump used for injection and its characterization from fabrication results in a standard .5um CMOS process. (a) A 3 stage high-voltage charge pump used to control injection in floating gate circuits. (b) The current loading characteristics for controlling injection on the drain of the floating gate device. The voltage interval of approximately 7V-8V will definitely accommodate the required interval 6-6.5V needed for injection.

higher frequencies, at approximately 5MHz, there is a sharp decrease in magnitude that is a result of attenuation of the inverting clock signal magnitude that was supplied using an off-chip source. It was observed that the inverted clock signal, as measured at the protoboard, started to experience attenuation and phase shifting above 1MHz. The charge pump used for injection would be used to source approximately a total of 1μA for an array of 100 elements, each sinking 10nA. The high voltage structure was an NMOS diode connected transistor except that the source and drain regions were surrounded by N-wells.

5.5.5 Incorporating these into the ACA programming structure

These charge pumps have been shown to actually tunnel and inject a single floating-gate element. However, charge pumps will present a bit of change in the way the ACAs have been constructed in the past. As pointed out, charge pumps are limited in current drive before their output voltage start to drop. Because of this the arrays must be zoned in to smaller chunks. This zoning has been considered and actually changes the row column selection of the devices.

Shown in Fig. 62 is the low-power scheme for switching the charge-pumped injection



a) b)

Figure 61. Illustration of transient and frequency output voltage characteristics from fabrication results in a standard 0.5um CMOS process. (a)Transient output of Schottky 6 stage charge pump boosting output voltage from approximately 4.3V to 18V. (b) Output voltage as a function of frequency for the high voltage and Schottky charge pumps. These tests were effected with a 5V input and a 5V clock operating at 5MHz for a no load case.

voltage onto columns. Because of the current limitations the wells of a minimal number of floating-gate elements can be charged at one time. There is no static power dissipation in this implementation and transient switching power was calculated in simulation to be orders under the maximum output from the charge pump. Even if the voltage at the level-shifter drop during switching, which would occur if it could not supply enough current while switching, it will still switch but at a slower rate. Once the transition is complete the charge pump voltage will recover back to injection voltage. This was verified in simulation by a current limited voltage source

This current limitation, and subsequent need to separated the columns wells has a negative effect on the density that can be obtained. However, this changes the access method that has been used in the past to isolate a floating-gate element to program. Traditionally we have used the gate lines to provide column selection of devices. However, with each columns well separated we can use this to provide the column selection. There is an advantage to this method, current when devices are highly programmed and in a column not selected, they can still injection. This is because the V_{ds} need to inject exists in all devices

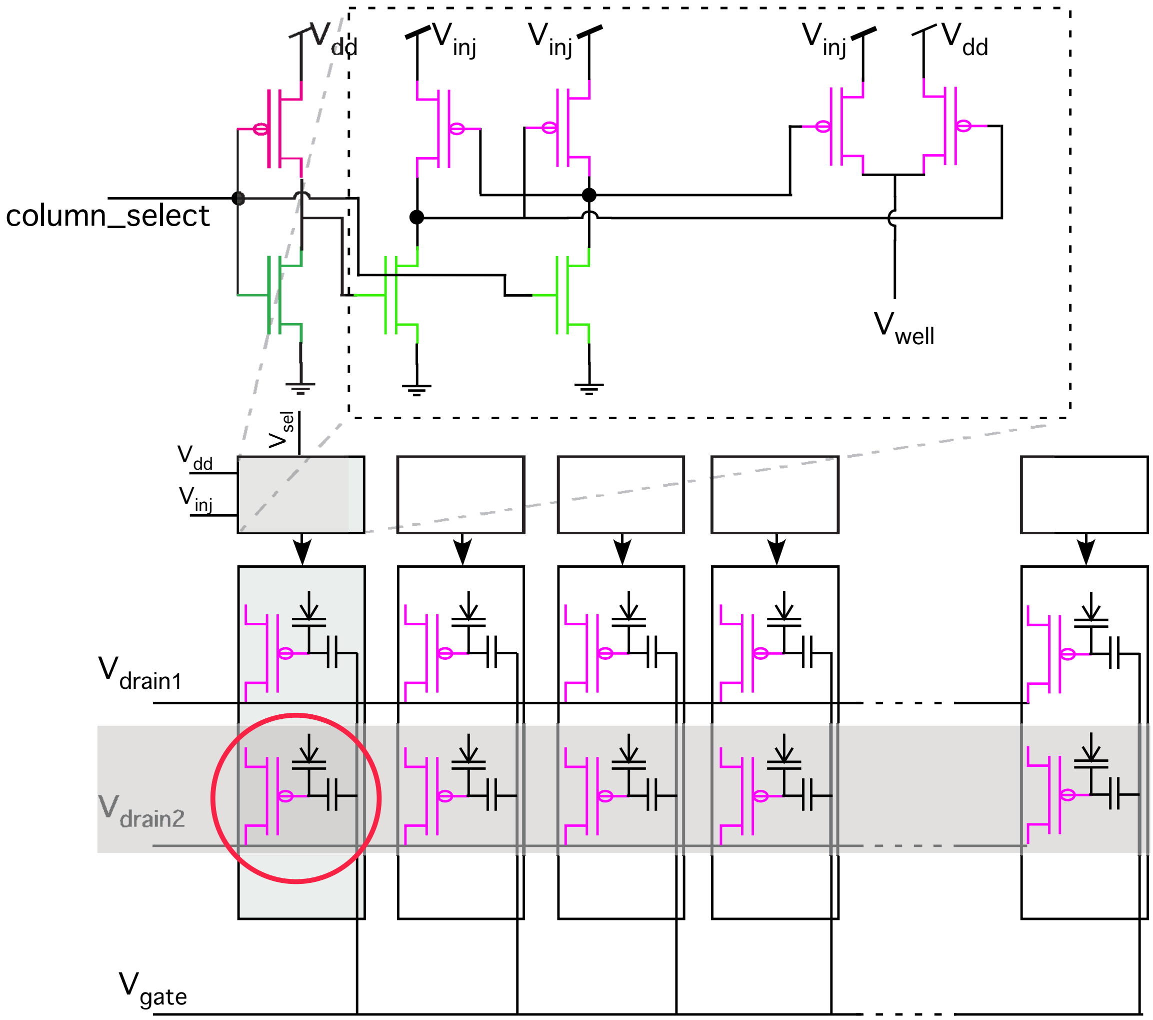


Figure 62. Shown is the low-power scheme for switching the charge pumped injection voltage onto columns. To use the charge pumps in the arrays the columns well are separated to allow only the devices in single column to the powered up and injected.

in the current programming scheme. If the device is highly programmed there will still be channel current even when the gate if pulled to Vdd or off. In the new scheme even if there is channel current there will not be an appropriate V_{ds} for injection. Actually, all gates in the array can be tied together during programming pulse. However, they will still need to be separated to measure the current programmed into the devices. This new method is still compatible with row-parallel programming. The number of rows that can be programmed simultaneously will be dependant on the current supplying ability for the injection charge pump.

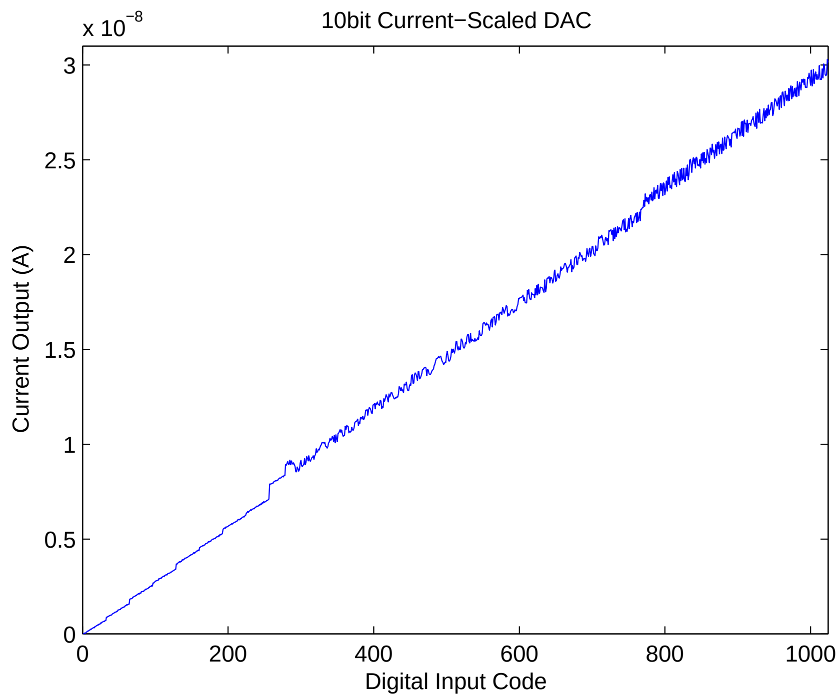


Figure 63. Shown is the measured data of the 10-bit current-scaled DAC convertor

5.6 DAC Block

The row-programming system requires a few digitally controlled voltage source for providing the programming pulses to gate and drain lines. When these voltages were needed that have been supplied in the off-chip programming systems by commercially obtained DACs. It is desired to move these DAC on-chip to reduce pin-count and move towards the goal of a digital-only interface for programming ACA chips. A current-scaling DAC [1] was designed, fabricated and measured. This design uses a unit transistor that is connected in series and parallel to alter the effective W/L of each bit. By using the unit transistor and replicating it to create each bit the hope is the matching can be improved to provide increased accuracy on each binary weighted current source.

The DACs measured INL and DNL are shown in Fig. **??**. It is observed that the INL and DNL error increases as the code word increases. This is because the noise from the bias supplied to the reference unit becomes amplified. The voltage sources that are used

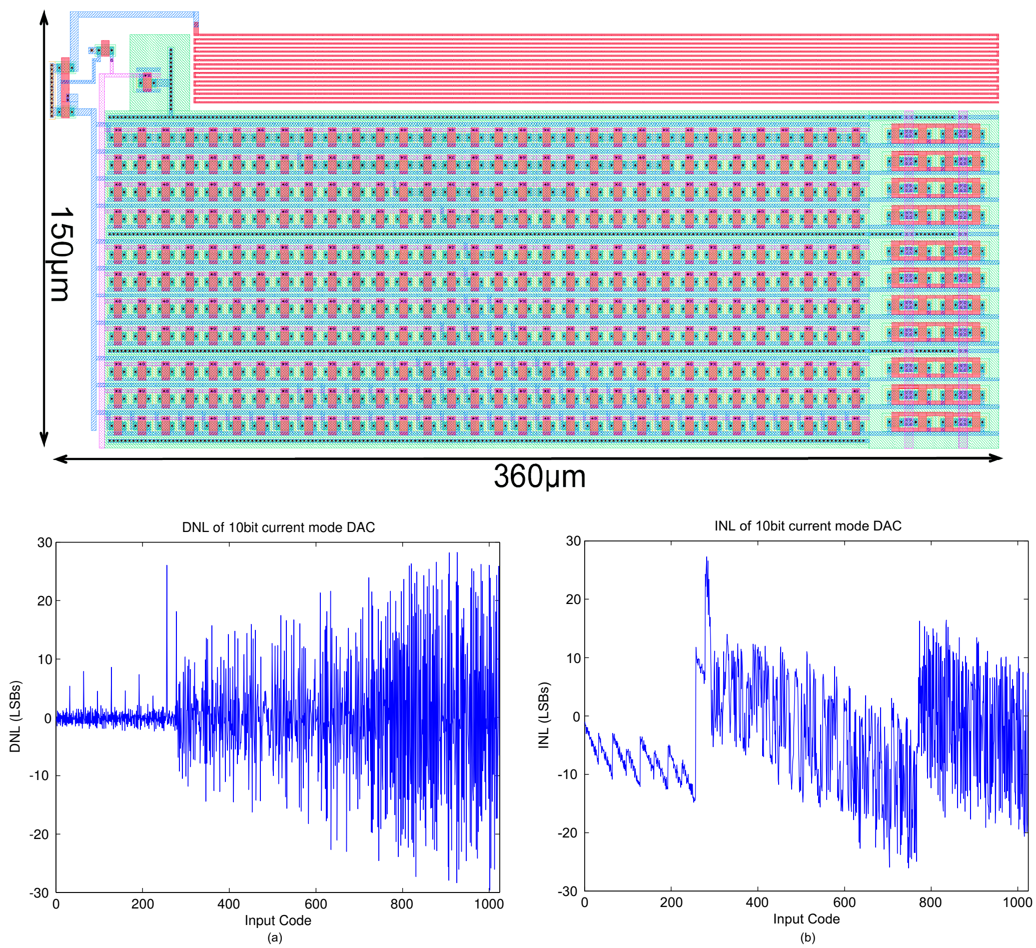


Figure 64. Layout of the 10-bit DAC convertor. This layout show the DAC design from a unit cell replicated and connected to provide the DAC bits. The output of the current mode DAC is converted back to a voltage using a poly-resistor. The Intergral Non-Linearity (INL) and Differential Non-Linearity(DNL) for the 10-bit DAC is presented. It is observed that the INL and DNL error increases as the code word increases. This is because the noise from the bias supplied to the refrence unit becomes amplified.

to set the bias current on the reference cell was done using a potentiometer. This reference current will need to be generated on-chip using current bias-generators to provide a clean DAC.

CHAPTER 6

HANDLING AND RETENTION ISSUES

Presented are handling and use issues to consider when using floating-gates in analog circuits, such as where to set bias voltages to reduce long-term effects and what to watch for when powering up and down the system. Designs to compensate for long-term changes resulting from global disturbances are presented, both for computational blocks as well as bias currents using floating-gates. Also presented is long-term data from an array of floating-gates demonstrating their ability to hold charge over time.

6.1 Motivation

We seek to address the reliability of floating-gates in analog systems as they have recently been perceived and published[2] as unreliable. Currently, it is accepted that digital Flash (EEPROM) memory, which is based on the same floating-gate structures, is reliable - so much so that we entrust our irreplaceable pictures to this technology. Flash circuits are generally rated to retain their information without any failure for ten years. Also in the flash systems the charge on the floating-gate is written and erased many more times than one could envision many of these analog systems to be. This constant writing and erasing tends to degrade the oxides and yet they are still rated and trusted to ten years. The ultimate goal is to produce the same level of confidence for floating-gates in analog systems as is demonstrated in digital systems.

6.2 Floating-gate Device

The schematic diagram of a CMOS floating-gate is shown in Fig. 65. A simple way to examine this circuit is as a MOS transistor with a capacitive divider attached to the input. Besides the input V_{in} and tunneling capacitor V_{tun} there are capacitors, inherent to the transistor, to the drain, source, substrate, and channel which have been omitted for

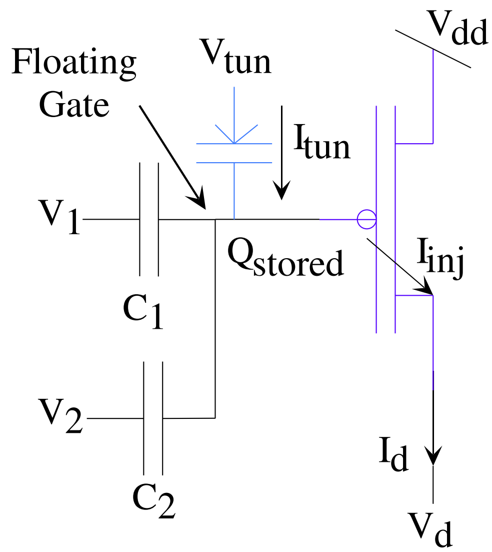


Figure 65. Schematic representation of a floating-gate device. This figure demonstrates the ability to have multiple input signals into the device. Also noted is the actual floating-gate which has no DC path to a supply rail resulting in trapping of charge noted as Q_{stored}. Modifying this charge results in a change of the floating-gate voltage V_{fg} and therefore a change in current through the transistor(I_d) at a give input state.

simplicity. Therefore, many input signals are coupled onto the input of this one transistor

$$V_{fg} = V_1 * \frac{C_1}{C_T} + V_2 * \frac{C_2}{C_T} ... + Q_{stored} \tag{41}$$

where C_T is the total sum of all the capacitances onto the gate and V_{fg} is the voltage on the floating-gate. Note that the floating-gate voltage which affects the output current has a term Q which is the change stored on the floating node. This charge, that can be modified through several processes, will stay constant unless modified as it is isolated with no DC path to ground. Any modification of this charge quantity will result in a change in the current output of the transistor at a give input state. Therefore, unlike volatile circuits that analog designers are accustom to working with, anything that can modify this charge will result in a permanent change in the circuit operating state.

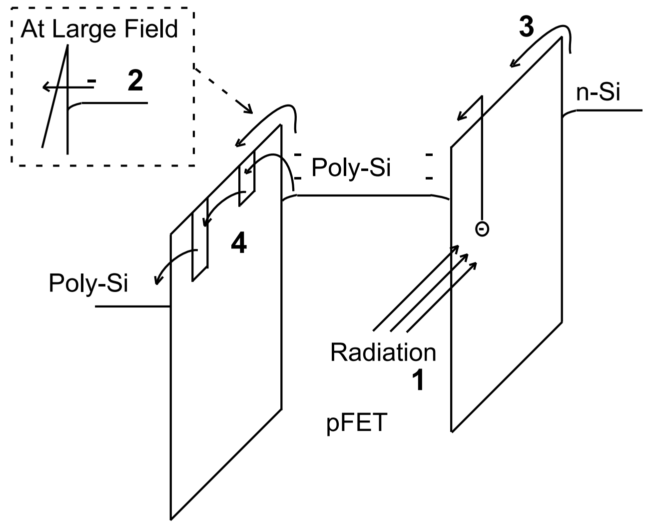


Figure 66. This figure shows the band diagram representation of the floating-gate with respect to an input gate and the channel. Shown also are the mechanisms for changing the charge on the floating-gate which are (1) Electron generation from radiation(UV, cosmic rays) (2) Tunneling through oxides with high fields (3) Electrons with enough energy to overcome a barrier, generally hot electron injection, (4) hopping of charge through traps in the oxides. Also trapping and subsequent detrapping of charge effect total gate charge.

6.3 How to Modify the Charge

When using these devices in circuits it must be understood how this charge can be modified. Not only can these methods be used to intentionally modify the charges but care must be taken when handling, testing, or using the device to avoid an unintentional change from these same processes. Also, understanding how the charge is modified will give insight into where voltages should ideally be left on nodes used to program the device afterwards when change is no longer desired to the device. There are four methods for modifying this stored charge. They are radiation which is generally UV performed through a glass cover in the package, tunneling through a thin dielectric layer, channel hot-electron injection, and hopping through or trapping/ detrapping from the oxides surrounding the floating node.

First, radiation can alter the charge on the floating-gate. This charge alteration takes place when applied radiation causes electron-hole pairs to be generated throughout the chip. By biasing fields in the desired direction charge can be added or removed from the floating-gate. In most cases this is used solely to remove charge from the floating-gate as in

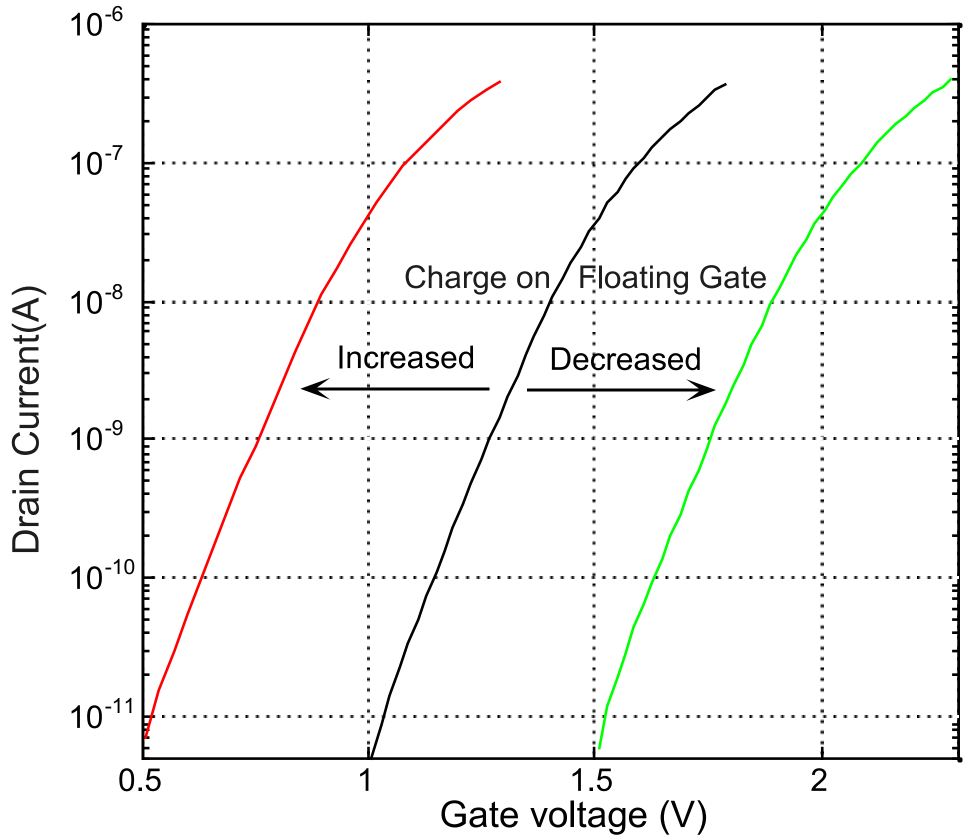


Figure 67. This shows the effect of altering the floating-gate charge on the device. Increasing the floating-gate charge results in a larger current at a given gate(input) voltage. Decreasing the floating-gate charge results in a smaller current at a given gate voltage. The charge is trapped and therefore the device is said to be programmed.

UV erasing of EPROM but it has been demonstrated that it can be used to program floating-gates[4]. Other forms of radiation are unintentional from the environment and generally are critical to retention times. Floating-gate devices used in space will be exposed to a large amount of cosmic rays which will reduce the retention times from normal operating environments.

Next, Fowler-Nordheim tunneling through thin oxides can modify the charge on the floating-gate. This field assisted tunneling is achieved when a large enough voltage appears across the interface of the oxide. The height and thickness of this barrier depends upon the material and process used. Due to the high electric field the electrons in the conduction band see a triangular barrier. Therefore the barriers effective width is dependent upon the magnitude of the applied field or voltage. When the barrier is small enough electrons can tunnel through this oxide. The current resulting from tunneling is increased exponentially with a linear increase in applied field. Therefore care must be taken once the tunneling current starts so as not to over-tunnel a device. Also large transient voltage spikes can damage the oxide creating oxide holes allowing a path off of the floating-gate. Lowpass filtering of this input to the chip is recommended to prevent oxide damage. For this process to occur, a large enough voltage is placed across the oxide and the direction of this voltage will determine the field direction and where the electrons will be moved from-to(on or off the gate).

Another process to modify the gate charge is to excite the electron with enough energy to surmount oxide barriers. Unlike tunneling where we attempt to reduce the effective barrier width to move through it, giving an electron enough energy will allow it to go over the barrier. This is generally done by a process called hot electron injection. In the pFET devices we accomplish this by giving a minority hole carrier enough energy that when it collides in the drain region it results in an electron-hole pair with substantial energy. The generated electron will follow the nearest field. In the pFET case it will be swept onto the floating-gate near the drain, and the resulting hole will be swept into the well. For hot electron injection to occur a sufficient field is placed across the channel from source-drain to accelerate the minority carriers in this region. Also, a gate voltage must be applied to the device so that a channel can form allowing the minority carriers to flow. Therefore, the gate and drain voltages can be modulated to adjust the amount of gate current or carrier injection onto the floating-gate. Finally this charge modification method occurs exponentially with varying source to drain voltage.

Lastly, and the most determinate to the long term changes experienced on the floating-gate is hopping and trapping/detrapping. Defects in the oxide create states which can be occupied by electrons. These states can be visualized by a decreasing of the barrier height and width throughout the oxide as seen in Fig. 66. Charge which has enough energy to move into the next state can tunnel through the barrier, which is smaller than the complete oxide width, to the next trap. The field across the oxide will still determine the probability of tunneling(hopping) and the direction in which tunneling will occur. This field will therefore determine if the charge will increase or decrease on the floating-gate due to oxide defects. Also during tunneling, when fields are present to move charge completely through the oxide the electron can become trapped in these defects in the oxide. This charge, whether trapped in the oxide or on the floating-gate, will affect the overall channel of the transistor. One could visualize the single oxide as having many floating-gates all coupling into the real floating-gate at different ratios. If this charge detraps onto the floating-gate the electron will not have a stronger influence on the channel resulting in increased current. If this charge detraps onto the input node it will be collected and the transistor's current will decrease according to the coupling of the trap to the channel.

6.4 General Handling Issues

The last section describes how the charge on the floating-gate can be changed, this can be done intentionally or even unintentionally. Most analog designers and testers are used to the volatile nature of their circuits and tend to overlook issues that have unintended effects on non-volatile circuits. We are accustomed to when our chips are moved into an undesired state, such as latch-up, simply powering down and perhaps placing them in conductive foam for sometime. One might not even take care in moving them from the setup to the foam due to chip-saving protection circuits on the input pins. After sometime these chips return back to their normal state due to the volatile design. However with non-volatile designs, any change in the charge stored on the floating-gate will effect operation long term as the state is preserved when power is removed.

Handling of these chips is even more critical than in the aforementioned systems. In volatile systems an induced voltage onto the pins, if not large enough to puncture oxides, will not alter the system's state. Because floating-gate systems store information when

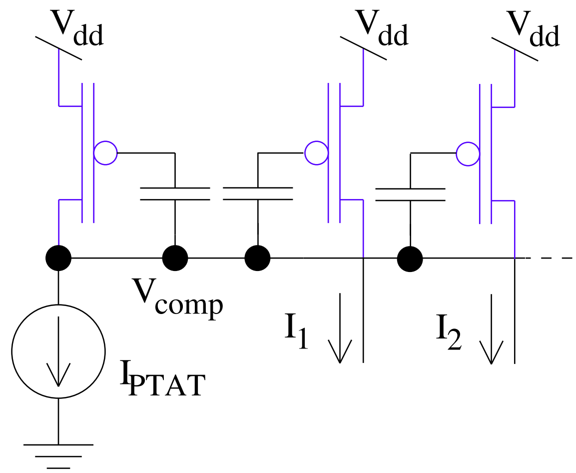


Figure 68. This structure can be used to create programmable bias currents that are resilient to global changes in floating-gate voltages. The diode connected floating-gate provides feedback from changes in the floating-gate voltage over time to the bias current floating-gates in a master-slave fashion. This device provides feedback by adjusting V_{comp} as its floating-gate charge changes.

the power is removed a voltage to a pin large enough to tunnel charge but small enough not to puncture the oxide will alter the stored information unintentionally. Also, because floating-gates deal with high voltages for tunneling, not only is it signifiicant to address possible unexpected conditions from the standpoint of causing change to the stored value, but voltages slightly higher than tunneling can unintentionally be applied large enough to puncture the oxides destroying or degrading the floating-gate.

Shown in Fig. 65 is the schematic for a floating-gate. In a lab setup the voltages to each node are generally applied with separate electronically controlled voltage sources such as the common Keithley SMU, allowing the current into each node to be monitored. The most common unintentional change in the floating-gate occurs when multiple voltages are changed simultaneously. Most setups control these units over GPIB via lengthy commands introducing delay between the change in each device. While it might be assumed all voltages change together even slight delays can result in one of the conditions from section 6.3 to occur, generally injection. When the device is brought up to a voltage to allow for injection, the system vdd would be moved up by some amount and then the drain voltage with a slight delay occurring between the two. If the source to drain voltage becomes large enough because the system voltage was moved first, even if for only a few milli-seconds, injection onto the floating-gate can occur. Therefore stepping of voltages in increments and not instantly can be employed to prevent this. However, as noted brief if even milli-sec pulses have been shown to favor injection. Therefore if your test equipment when changing ranges pulses briefly in a direction favoring injection the charge on the floating-gate will unintentionally change. In a volatile system these glitches which might occur daily do not affect the system state after the correct range is ultimately established.

From past experience and the untimely retirement of several floating-gates, it has been noted another common state change occurs when the system is powered down in a certain order. Again test setups generally use multiple power supplies to set voltages. If all these voltages are referenced to ground and the system supply voltage is turned off before the tunneling voltage, voltages then across the tunneling oxide can become great enough to tunnel or destroy the oxide. Therefore it has become custom for our setups to reference the tunneling voltage to the system vdd. That way when the system voltage is reduced or removed the tunneling voltage will follow by the same amount keeping the fields in the oxide below those required to tunnel. Because this occurs when the system is powered down, it may appear as if the floating-gate was unreliable and lost charge while power was not supplied to the chip. However, what really occurred was the charge was modified as the system was powering down.

6.5 Design to Compensate for Long-Term Effects

Most longterm drift of floating-gates will not be due to tunneling or hot-electron injection. Generally, the voltages applied to the chip after programming has been performed prevent these two charge modification methods from occurring. The charge modification methods that occur long term are radiation from environmental sources and detrapping from or hopping thorough defects in the oxides surrounding the gate. Both of these methods tend to

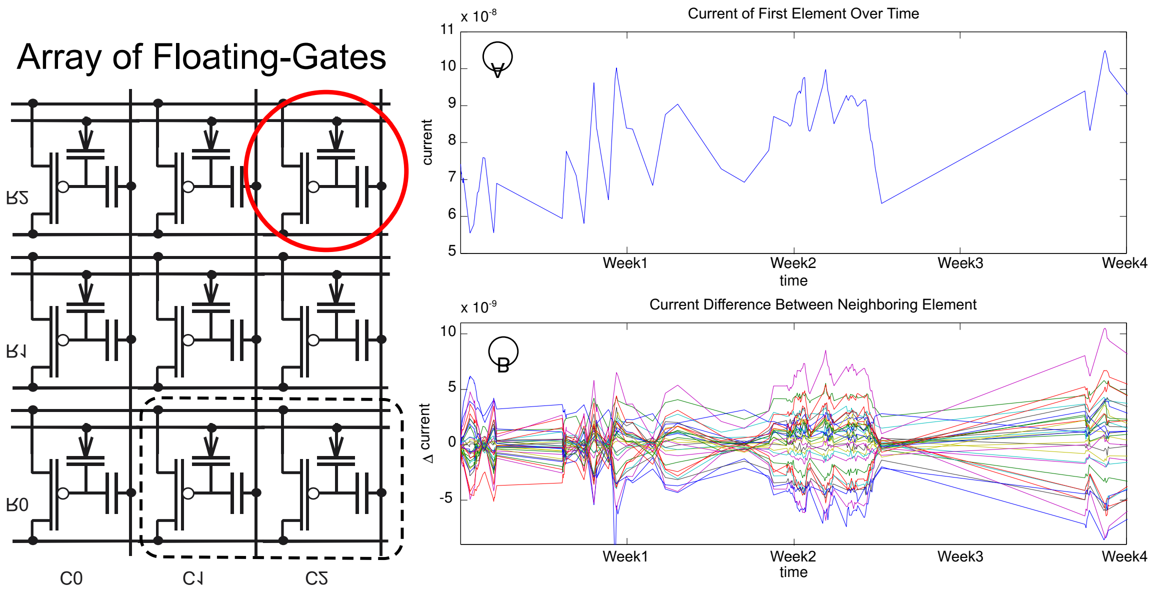


Figure 69. The array of floating-gates is shown above. Access to each floating-gate element is possible and a single element is shown with a circle around it. A single differential structure is shown by the dotted box. Plot A was recorded from a single element sampled periodically over a 1 month period. Plot B shows the difference in charge change between neighbors. This is the difference that would appear in a differential setup using floating-gates. Noise as well as differences in both plots is most likely due to temperature fluctuations or other random effects influencing the system.

occur fairly uniformly across the chip over time as the whole chip is exposed to the same radiation source or the defects are generally uniformly occurring. Carefully setting biases to unused inputs onto the floating-gate so that fields across the oxides are minimized will help prevent the magnitude of change in charge due to these effects. For example the tunneling junction should be left at a voltage slightly above that of the floating-gate voltage. This will reduce the field across the oxide and the tendency to pull electrons through the traps from charge hopping. Also electrons generated from radiation to the oxide and detrapping will tend to be swept off towards the tunneling pin and not onto the floating-gate.

The use of differential configurations can be used to overcome locally occurring disturbances in floating-gate systems as well as provide compensation for long term drift effects. Many of the systems we have built based on floating-gate concepts rely on the processing that comes from the simple multiplier structure as shown in Fig. 69 enclosed by the dotted box. This core circuit performs its computation on the incoming signal by the difference in two floating-gate charges or weights.

$$I_{out} = I_{so}(W^{+} + W^{-}) + I_{so}(W^{+} - W^{-})\frac{\Delta V_{in}}{V_{y}} \tag{42}$$

If the charge of both gates changes in the same direction by the same amount the computation weight will be maintained. The effect will be a slight change in the bias current present at the output along with the signal which could be compensated if not desired by another floating gate.

Floating-gates that are used as biases obviously cannot enjoy this differential benefit. For bias circuits a system such as that in Fig. 68 can be used. The diode connected transistor provides feedback to globally occurring phenomena that change floating-gate charge by adjusting V_{comp} as it's floating-gate charge changes. This change in V_{comp} is the necessary adjustment to correct for the change in the bias current transistors. The actual current out of each floating-gate can be adjusted by intentionally programming the charge on each device.

However noting this difference lends to efficient system design. floating-gates that are to be used in the multiplication structure can be made smaller to pack more of these core computation A small change in the charge of the floating-gate might have a noticeable effect on the threshold shifting but as we mentioned When designing the floating-gate cells that might be used for biasing currents they should be built larger so larger changes in The number of biasing floating-gates in systems might only be a small faction, about 5% of the total number of floating-gates

6.6 Long Term Testing

As was addressed in section 6.3, charge on the floating-gate is permanent unless some process occurs to move this charge off the gate. Processes that exist at all times even when not desired are radiation and charge hopping through the oxide. In order to determine the rates at which these process occur we left a setup running for 1 month after tunneling then injecting the device. This setup consisted of a custom programming/testing board that provided the bias voltages and read the device's current connected to an array of floating-gates fabricated in a 0.5μm MOSIS process. A computer was connected up to this custom board via a serial connector at periods throughout the month and the current from each device in the array read. The following biases were applied thorough the month; 4V V_{ds}, 3.65V V_{gs}, and the tunneling pin voltage placed and left at 4V after tunneling was performed. All current measurements shown in Fig. 69 were taken at these biases throughout the month. The setup was placed in a metal enclosure in an attempt to reduce stray interference because the currents were read off chip and small. This setup however was not isolated from temperature variations throughout the month.

The results obtained from this month long test in Fig. 69 are encouraging. Overlooking the random fluctuations, the current lacks any dramatic change with possibly only a slight linear increase in current. If electrons were to be removed due to leakage off the gate we would actually expect a decrease as noted in Fig. 67. Therefore, it is possible that even at a V_{ds} of 4 volts injection might have been slowly occurring, although one would expect more of an exponential instead of linear trend. However, there is enough noise in the data from random fluctuations that a week more of data could result in no movement of the current from day 1. What can be taken away from this data is that the change is not as dramatic as some have claimed and more in line with what has been observed from digital EEPROM publications.

Shown in Fig. 69B is the current difference of neighboring elements in the array. In the design section above, this method was proposed to reduce error from global changes experienced in the chip. It is clear from this plot that while the single element varied by 30%, the change versus the neighbor was always less than 2% and generally less than 1%. This demonstrates the beneficial nature of the differential configuration for analog computing systems which use floating-gates.

CHAPTER 7

VECTOR QUANTIZER - ACA SYSTEM

Another system that has been constructed using the ACA architecture is the Vector Quantizer (VQ) [30]. This system provides a classification of incoming signals against a set of stored vectors and therefore requires programmability, a perfect fit for utilizing the ACA system approach. The VQ is generally used for data compression by classifying signals into symbols. A VQ is used to reduce the set of detectable spectrum vectors to a manageable set for later classification. An Analog-to-Digital Conversion (ADC) classifies data in a single dimension; a vector quantizer classifies data in an arbitrary number of dimensions. VQ is typically used in data compression, and like an ADC, VQ results in lossy compression. The goal of VQ is to provide the simplest possible accurate description of a signal so as to minimize the subsequent complexity of signal processing algorithms such as classification.

The VQ system computes how far an input vector is from desired or programmed vectors and then provides a coded output of which stored vector is closest to the input vector. The ACA version of the VQ system is built using an array of floating-gate bump circuits [7] to implement the stored vectors. The output vectors are measured using a winner-take-all circuit [41], continually computing which vector is closest to the incoming signal. This lossy compression provides a description of the incoming signal in terms of symbols instead of raw data, providing some degree of computation. This system is a fundamental component for use in a low-powered speech recognition system [51]. A revision of this system utilizing the programming core and the under-development on-chip system has additionally been sent for fabrication.

7.1 Mathematical Basis of VQ

A VQ system will compute how far away a particular input vector is from the desired target vectors, and pick the code vector that is closest to the input vector. For example, in an ADC,

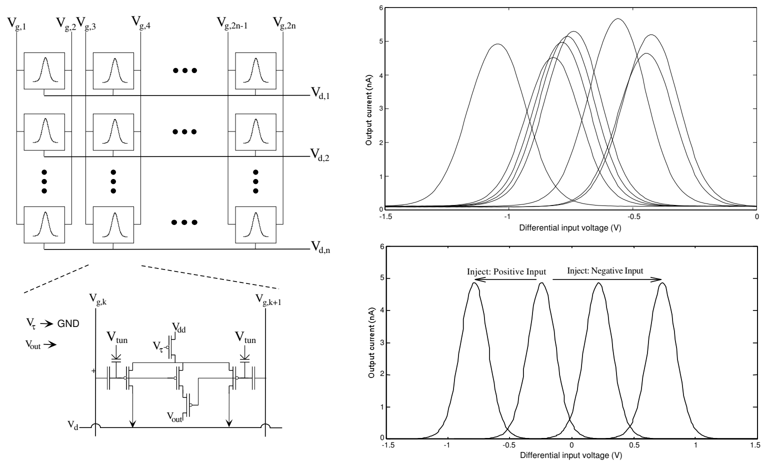


Figure 70. Programmable VQ using floating-gate circuits. We reconfigure the VQ circuit so that it fits within the standard floating-gate programming architecture and algorithms. Capacitors with an additional arrow are our symbol for a tunneling junction, which is a MOS capacitor that can also remove charge from the floating-gate node. We show experimental results after programming eight cells to different offset voltages. The difference in the bump peaks is due to mismatch in the MOSFET transistors. We reset the floating-gate charge using electron tunneling and program positive or negative offsets using hot-electron injection.

we assign a particular digital code to a given input based upon which analog representation of a given digital code is closest to that input. For both VQ and ADC some information is lost in the representation, but the goal is that it should be a sufficient representation for the problem at hand.

We compute the closest input vector by choosing an appropriate distance metric. The question is how to choose the distance metric, $d(\mathbf{x}, \mathbf{m})$, between the incoming vector signal ($\mathbf{x}$) and the desired or target mean value ($\mathbf{m}$) for these signals. There are many different distance measures. Two particular measures we discuss are

$$\textbf{Norm} : d(\mathbf{x}, \mathbf{m}) = ||\mathbf{x} - \mathbf{m}||_n$$

$$\textbf{Gaussian} : d(\mathbf{x}, \mathbf{m}) = e^{-||\mathbf{x}-\mathbf{m}||_2^2/(2\sigma^2)} \tag{43}$$

where n and σ is depend upon the problem and input statistics. The first approach is preferred for real-time implementation, where the second is preferred for algorithmic reasons. Previous IC implementations have used simple $||\mathbf{x}-\mathbf{m}||$ metrics for their difference functions [12, 5, 44].

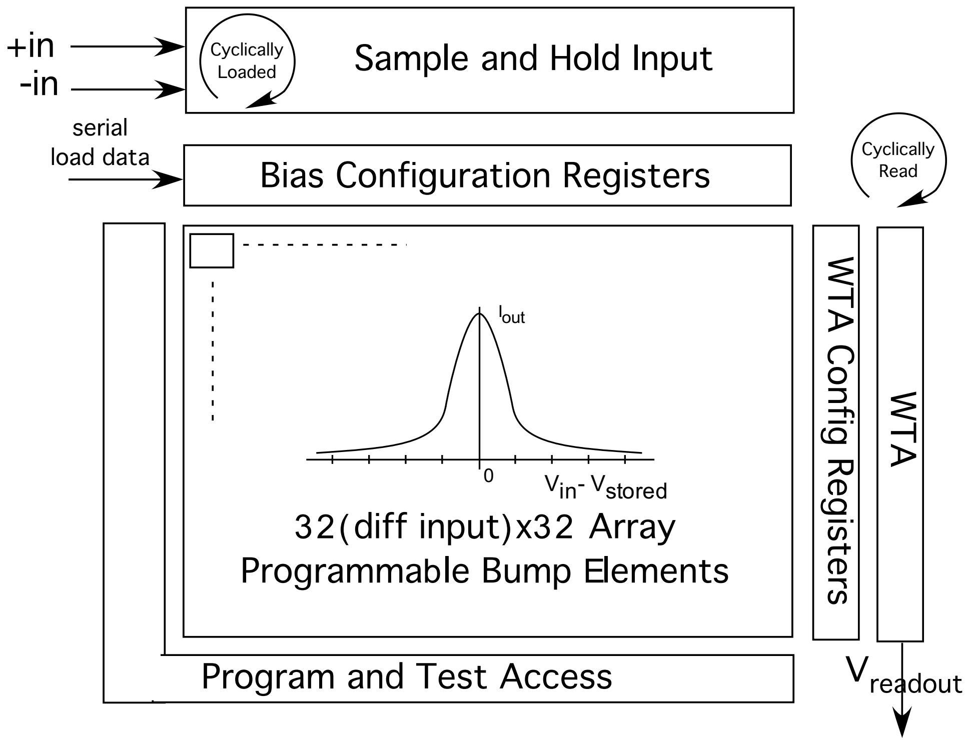


Figure 71. This diagram presents the VQ's system architecture. The main core of the VQ consists of programmable floating-gate bump elements. The implemented system has 32 differential input vectors. Each differential input uses two floating-gate elements to implement the programmability of the bump. The system also has up to 32 vectors that can be stored in the chip. The output of these 32 vectors is sent to a Winner-Take-All to provide either a A/D conversion representing the closest matching vector. This chip uses configuration registers to partition the chip from a 1x1 vector distance measurement up to a 32x32 vector quantizer. Additional the chip can be configured in many alternate configuration even permitting multiple vector qantizers out of the one chip.

We will use a metric non-volatile

$$d(\mathbf{x}, \mathbf{m}) = \sum_{k}^{n} sech^2\left(\frac{x_k - m_k}{\sigma}\right) \tag{44}$$

where σ is dependant on circuit parameters. We will also present a metric based upon a rough exponential function of this metric, which effectively turns the summation into a product, resulting in a more Gaussian-like formulation. This metric is close to the Guassian approach and easily implementable in CMOS.

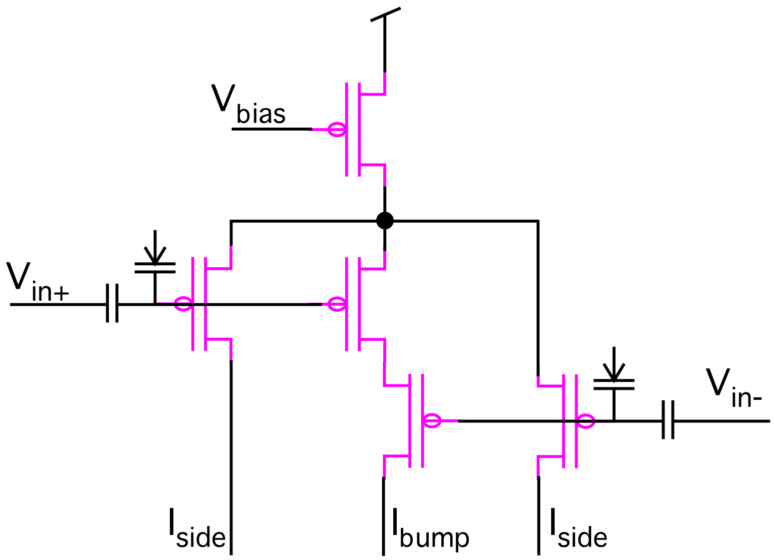


Figure 72. This is the bump circuit used in this implementation of the programmable Vector-Quantizer. It permits the distance programming in the bump, but does not allow the bumps width to be adjusted after fabrication.

7.2 Floating-Gate VQ Circuit and Architecture

Figure 70 shows the circuit and measured data from the VQ classifier array. Each cell in the array compares the value of that column's input to the value it has memorized; the output current flows out of the V_{out} node. This circuit is a variation on the bump circuit [7], which compares the two inputs to this circuit; this cell returns a high value if the two values match (minimal difference). The circuit performs a continuous distance computation along a particular input coordinate:

$$I_{out_k} = \frac{I_b}{2} sech^2\left(\frac{\kappa(V_k - V_{mk})}{2U_T}\right), \tag{45}$$

where V_k is the differential input voltage, V_{mk} is the resulting stored voltage representing the ideal mean value for this particular element, κ is the coupling from gate to surface potential, and $U_T = kT/q$ is the thermal voltage. This system outputs a measure of the similarity; therefore, the outputs of all the elements can be added (by KCL) and the largest output is the vector with the maximum similarity. The sum of these current outputs are sent through a Winner-Take-All circuit that outputs the N largest results, where N can be 1 or more [41].

We utilize floating gates elements at the inputs to provide the ability to store and subtract

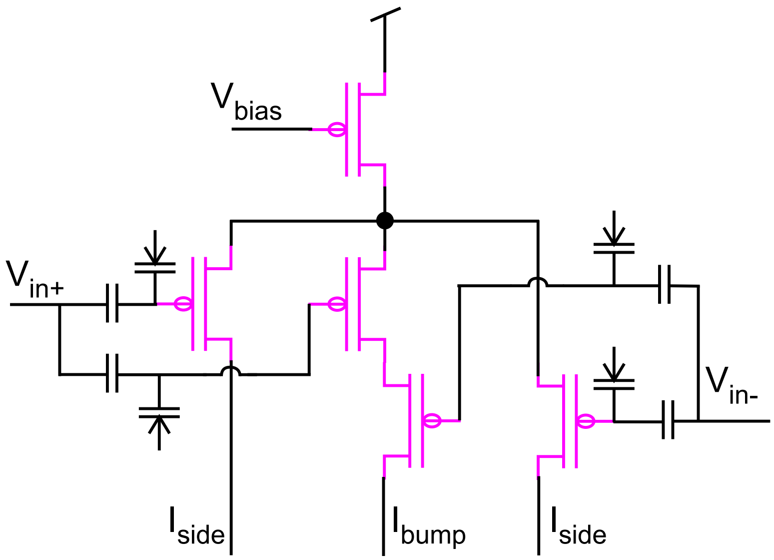


Figure 73. This bump permits the adjust of the bump width after fabrication. However this cell requires two more separate floating-nodes than the simple design even though the transistor count does not change. Additional, to program the floating-gate transistors used in the center leg more transistors and therefore signal speed reduction are the tradeoff for this additional degree of post-fabrication adjustment. The application will dictate whether this additional adjust is needed.

off the each cell's mean value. Setting the floating-gate charge establishes the mean value as well as eliminating the mismatch between the two-transistor pairs. Setting the size of the input capacitor as well as other capacitor elements around the floating-gate sets the linear range of the circuit, and therefore sets the width of the *bump* element. We increase the floating-gate charge (remove electrons) by electron tunneling, and decrease the floating-gate charge (add electrons) by hot-electron injection. Over the next two sections, we will address how to modify this floating-gate charge in an array of bump circuits. We present an architecture and resulting circuits which will enable direct programming / storage of weight vectors, as well as various methods for VQ weight adaptation.

7.3 Implementation

The VQ is constructed as an ACA where the core computation cell uses arrays of floating-gate bump circuits. The floating-gate bump circuit is shown in Fig. 70 uses two floating-gates to program the location of the bump by adjusting the bump's mean value. The number

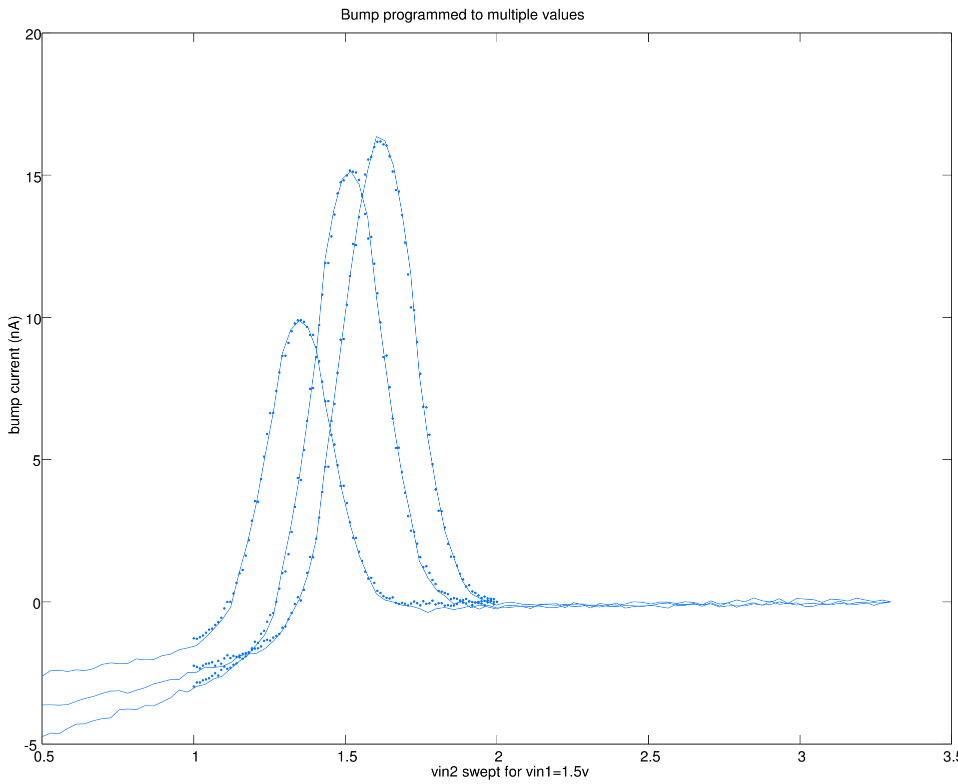


Figure 74. Shown is the bump programmed to different values. For each sweep one input was held at 1.5v while the other was swept. The bumps value was measured using the side leg current. The point should the same sweeps but using the Sample and Hold instead of directly through the bump.

of manageable floating-gate elements will determine the number of matching vectors that can be stored on-chip and the size of each vector. For the VQ to be useful in speech recognition systems we will need about 14-16 channels (vector size) and store 160-200 vectors. This translates to a requirement that 4000-6000 floating-gate elements be used on-chip.

The system architecture for the VQ is shown in Fig. 71 The core block of the VQ consists of programmable floating-gate bump elements. Therefore, because the bump block is the most repetitive block it should be optimized for size because the size of the bump block will determine the density of this system. In the system that was implemented and fabricated the bump block used is shown in Fig. 72. This bump core was designed using the same pitch height and width as the multiplier block used in many of the ACA system. This choice was made to permit the re-use of program control side-blocks without the need

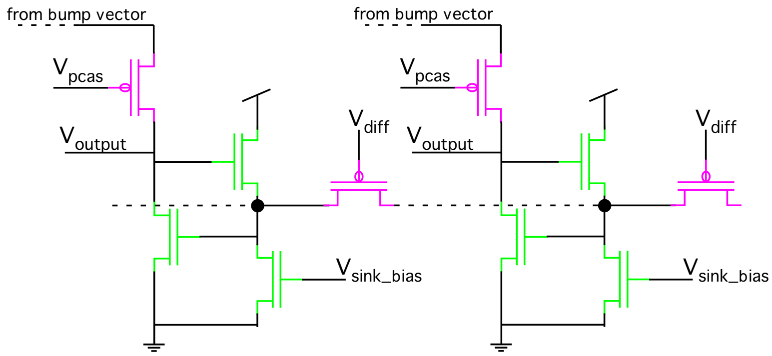


Figure 75. This is the WTA block used to convert the vector signal. It can be used to provide just the winner operating as a psudo A/D convertor. Scanning the outputs using the muliplexer with some post-processing would provide ranking information and the closeness to the stored vectors.

to redesign or re-layout these blocks and attempt to conform to the group standard. This choice did come with a tradeoff. The bump core allows the bump distance for each vector channel to be adjusted, but the bump width can not be programmed / adjusted after fabrication. The bump width of this design is set by the W/L of the center bump leg to the W/L of the side legs transistors [7]. The circuit shown in Fig. 73 additional allow the width of the bump to be programmed as a V_t offset can be created from the side legs to the middle node. This circuit however requires two additional floating-gate elements to be implemented in each bump core even though the transistor count does not increase. Moreover, to permit the programming of the middle leg's floating-gates, additional transistors must be used. reconfigure the block during programming mode to allow device selection for programming.

The implemented system has 32 differential input vectors. Each differential input uses two floating-gate elements to implement the programmability of the bump. The system also has up to 32 vectors that can be stored in the chip. The output of these 32 vectors is sent to a Winner-Take-All to provide either a A/D conversion representing the closest matching vector, or the ability to read out and smooth the closeness of each stored vector to the incoming vector. While the later provides more information it also requires more post-chip processing to extract the information. The application will obviously determine the

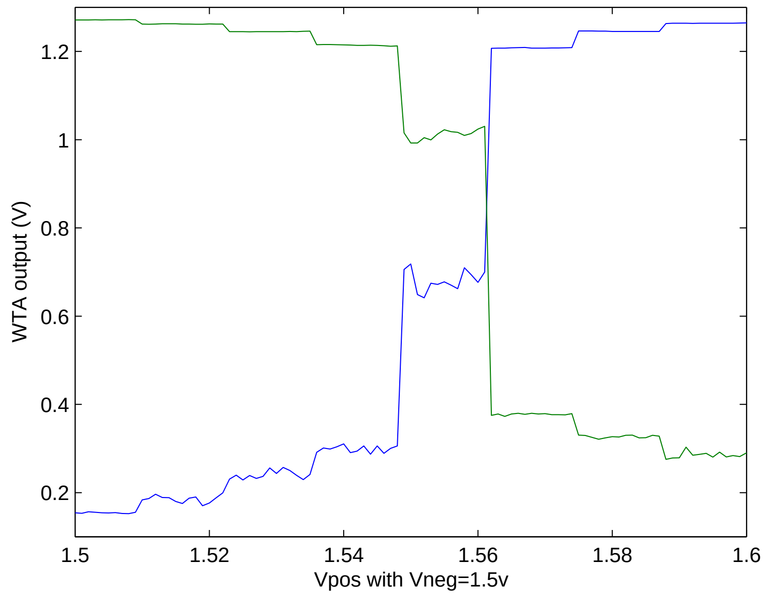


Figure 76. Show winning.

configuration that must be used. This chip uses configuration registers to partition the chip from a 1x1 vector distance measurement up to a 32x32 vector quantizer. Additionally the chip can be configured in many alternate configurations, even permitting multiple vector qantizers to be realized out of a single chip.

Post processing in this system is performed using a winner-take-all circuit, which provides data compression as just the winning vector can be transmitted to the next stage. There are several issues to address when implementing the winner-take-all circuit. First, the level of compression must be considered. If we would like only the closet match, the original Lazzaro WTA [41] circuit would provide this. However, in many system implementations it is desired to know the closest x number of vectors and the rank of each. Circuits to perform ranking currently exist [55, 33] but dramatically increase the transistor count over the basic winner-take-all.

The construction of a WTA that is compatible with the ACA is more complicated than just using the Lazzro WTA. The isolation between the drain line of the ACA arrays and the

WTA must be considered. In the Lazzro WTA, the current input node is also the evaluation node. This presents a problem for the ACA. The voltage at this node will vary as the inputs move from losing to winning. For the ACA to correctly function the drain voltages in the computing array must remain constant to preserve accurate computations. Therefore, methods such as cascoding the input or mirroring the current must be used to solve this problem. In this systems, the use of a cascode was choosen. Simulations show the the ability to fix the drain currents out of the bumps.

The Vector Quantizer as implemented as 64 inputs that represent the 32 vector differential signals that will be classified by the chip.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Accomplishments

In 1999 when my graduate research began there was no infrastructure in place to program floating-gate devices. Early on many potential systems were discussed that could only be constructed with the use on many (100+) floating-gate devices. The early work on the Analog Fourier Transform system that used only 64 floating-gate devices revealed the need to develop a methodology to program large arrays for floating-gate devices. My major contribution to this field of work is in the development of this programming infrastructure. This includes the device access for programming, layout considerations for the array; especially when charge-pumps are introduced, the physics based programming algorithm used, the PCB system used to externally program the array, and finally the movement of all these system into an self contained on-chip programming system.

A scheme for systematically programming the floating-gate elements in the computing array and other preprocessing blocks has been presented. The existing methods for changing floating-gate charge were considered, and their properties determined the programming/selection scheme. Hot-electron injection was used in to program the devices, utilizing the high selectivity which can be achieved in the array configuration allowing for greater accuracy. Tunneling was used to erase the array globally, or individually by element with the use of on-chip high-voltage switches, by bringing the floating-gate charge below the desired level to program. The ability to program these arrays was demonstrated from experimental results in an initial 2x4 array and in the programmable filter that contained a larger array of 3x20 elements.

The design and construction of a custom PCB programming board used to program chips using the analog computing arrays has been designed and tested. The single board contains all the measurements and control circuits need to quickly and accurately program large arrays. The programming board circuits are locally controlled by a PIC that is controlled over a serial port from a computer. Matlab can then be used to implement the programming method as well as graphical user interfaces further removing the knowledge of the underlying physics need to program these arrays. This programming board design has provided the fundamental starting point in several chip custom PCB designs in the ICElab, including design errors that seem to survive the re-designs. There is a revised programming board updated by others that is FPGA control and simply a rework of the circuits used on the original programming board.

The current measurement circuitry has been implement on chip. This was done to permit the rapid measurement of the small subthreshold currents. Also, placing the measurement circuits on-chip allowed the transition into a row parallel architecture using a scheme similar to that employed in the Address Event protocol to simultaneously read all rows currents. The design and initial test of additional on-chip circuits, such as the digital to analog convertor and the sample and hold to load in multiple drain voltages into the entire array, the array could provide injection pulses to every row simultaneously. This simultaneously programming of an entire row/column allows for a dramatic speed increase from programming the arrays using the current single device scheme. The evaluation, design, and test of many sub-circuit blocks needed for the full on-chip programming system have been tested. Data from the full system is still forthcoming. Research, design, and testing in this area is ongoing and will continue by myself after graduation.

The design of charge pumps for providing the tunneling and injection voltages have begun. Several designs have been fabricated and shown to tunnel and inject the floating-gate devices. The limitations of the charge-pumps prompted the consideration of redesigning of the floating-gate arrays from initial designs. This was done to accommodate the low current supply from the charge-pump available to hot-electron injection during programming. These new array structure with attached charge-pumps have been submitted to fabrication for silicon testing. This work will continue under a colleague with whom I have been working collaboratively since just after I sent out the first charge-pump test structures.

Additionally many systems using the Analog Computing Array approach where designed and presented. The Programmable Analog Filter is one such design. The combination of the analog computing arrays with the C^4 used as a pre-processing block, resulted in a fully programmable analog filter. The programmability of the analog computing array allows for the realization of many filters simultaneously through the parallel analog processing. This filter's operation was demonstrated via experimental results from initial fabricated chips in MOSIS 2.0μm process. Current versions of this system include programmable corners in the bandpass filters that allow arbitrarily spaced filters. This system has been carried on by others who are characterizing this filter for parameters such as noise, distortion and speed. The programmable filter is fundamental for system designs such as Cepstrum phoneme recognition, MEMS preprocessing, and others.

The C^4 filter that was used in the programmable analog filter was fabricated, tested and analyzed. The transistor-only version of the Autozeroing Floating-Gate Amplifier(AFGA), named C^4, that was used as the basis function in the programmable analog filter has been developed, tested, and characterized. This circuit is a bandpass filter and behaves similarly to the AFGA with different operating parameters. Both, the low-frequency and high-frequency cutoffs are controlled electronically, as is done in continuous-time filters. This circuit has a low-frequency cutoff at frequencies above 1Hz and high frequency cutoff dependant on the g_m of the output transistor.

Initial reliability studies of floating gates utilized in this array scheme has been investigated. Studies of the reliability of these devices long term needs to continue. A test structure has been fabricated and initial data obtained from the chip. More data is need in this area and will be undertaken by myself in the future as access to a regulated temperature oven will become available. Also some of the long term affects are being investigated by others in the ICE lab as well. The effects of drift and how to compensate for it from a design aspect, like a coupled slave circuit globally compensating for drift and designs like the multiplier that are immune to common drift due to a differential configuration, have been initially examined. However, with a rapid on-chip programming system the analog values could be refreshed in a weekly, monthly, or on an as needed basis if retention issues in a process were not favorable.

Initial design and simulation shows promising results for the chips use in weight-perturbation adaptive filters. The goal is to create filters that program themselves to the correct place. The initial system design that was collaborativly undertaken was a Adaptive Channel Equalizer (ACEQ). This system will use a set of training signals to adaptive create a reverse filter for the transmitting medium. Initial blocks and system design have already been set to fabricated and will be tested when they return.

Floating-gates for large computation array were in an infant stage back in 1999. Since then many of these advancements and those of others in the ICE lab have resulted in a mature yet still growing field of new low-power analog computation systems. This work has such great potential that 2 other colleges, my advisor, and myself have formed a company based on these systems called GTronix. This company has licensed the research and patents from Georgia Tech that have resulted from the ICE labs pioneering working in this field. GTronix has such a compelling value proposition that a top venture capital firm has committed to a seed investment. This company will take this technology and industrialize many of the systems already designed. Therefore, my research will continue on to some degree even after I transition out of graduate studies at Georgia Tech.

8.2 Papers and Publications

The following is a list of papers and publications resulting from my graduate studies.

8.2.1 Journals

1. A programmable continuous-time floating-gate Fourier processor Kucic, M.; Low, A.; Hasler, P.; Neff, J.; Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, Volume: 48 Issue: 1 , Jan 2001 Page(s): 90 -99

8.2.2 Co-Author Utility Patents

1. ”Floating Gate Analog Circuit”- Georgia Tech No. 2637; TKHR No. 062020-1340 Patent Filed April 2003

2. Programmable Floating-gate ADC,DAC, and offset adjustment. - Patent Filed May 2003

8.2.3 Conferences

1. Investigating programmable floating-gate digital-to-analog converter as single element or element arrays Serrano, G.; Kucic, M.; Hasler, P.; Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on , Volume: 1 , 4-7 Aug. 2002 Page(s): I -75-7 vol.1

2. Design and use based on long-term measurements of analog floating-gate array circuits Kucic, M.; Hasler, P.; Smith, P.; Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on , Volume: 1 , 4-7 Aug. 2002 Page(s): I -295-8 vol.1

3. Accurate programming of analog floating-gate arrays Smith, P.D.; Kucic, M.; Hasler, P.; Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on , Volume: 5 , 26-29 May 2002 Page(s): V-489 -V-492 vol.5

4. Mel-frequency cepstrum encoding in analog floating-gate circuitry Smith, P.D.; Kucic, M.; Ellis, R.; Hasler, P.; Anderson, D.V.; Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on , Volume: 4 , 26-29 May 2002 Page(s): IV-671 -IV-674 vol.4

5. Programmable and adaptive analog filters using arrays of floating-gate circuits Kucic, M.; Hasler, P.; Dugger, J.; Anderson, D.; Advanced Research in VLSI, 2001. ARVLSI 2001. Proceedings. 2001 Conference on , 14-16 March 2001 Page(s): 148 -162

6. A programmable continuous-time floating-gate Fourier processor Kucic, M.; Low, A.; Hasler, P.; Neff, J.; Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on , Volume: 48 Issue: 1 , Jan 2001 Page(s): 90 -99

7. An exponential model of channel-length modulation applied towards floating-gate circuits Duffy, C.; Kucic, M.; AiChen Low; Hasler, P.; Circuits and Systems, 2000. Proceedings of the 43rd IEEE Midwest Symposium on , Volume: 3 , 8-11 Aug. 2000 Page(s): 1044 -1047 vol.3

8. A programmable continuous-time analog Fourier processor based on floating-gate devices Kucic, M.; Low, A.; Hasler, P.; Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on , Volume: 3 , 28-31 May 2000 Page(s): 351 -354 vol.3

9. A transistor-only circuit model of the autozeroing floating-gate amplifier Hasler, P.; Kucic, M.; Minch, B.A.; Circuits and Systems, 1999. 42nd Midwest Symposium on , Volume: 1 , 8-11 Aug. 1999 Page(s): 157 -160 vol. 1

10. Characterization of Charge-Pump Rectifiers for Standard Submicron CMOS Processes, Mark Hooper, Matt Kucic, Paul Hasler

11. 5V-Only, Standard .5um CMOS Programmable and Adaptive Floating-Gate Circuits and Arrays Using CMOS Charge Pumps, Mark Hooper, Matt Kucic, Paul Hasler

8.2.4 Papers to be submitted shortly

1. Row-Parallel Programming of Analog Floating-gate Arrays, Matt Kucic, Paul Hasler

2. Programmable Analog Vector Quantizer, Matt Kucic, Paul Smith, Paul Hasler

3. Predictive Programming to sub pico-A ranges, Paul Smith, Matt Kucic, Paul Hasler

REFERENCES

[1] A , P. E. and H , D. R., *CMOS Analog Circuit Design*. Oxford University Press, Inc., 1996.

[2] B , J. A., H , R., H , P., and D , S., “A floating-gate pfet based cmos programmable analog memory cell array,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, (Geneva, Switzerland), 2000.

[3] B , W. D. and B , J. E., *Nonvolatile Semiconductor Memory Technology, A Comprehensive Guide to Understanding and Using NVSM Devices*. 345 East 47th Street, New York, NY 10017-2394: IEEE Press, 1998.

[4] C , R., M -C , P., P -V , M., and S -M , J., “Design of a step-up 400mw, 40v charge-pump for microrobotics applications in a 100v-0.7um intelligent interface technology,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2004.

[5] C , G. and P , V., “A low-power cmos analog vector quantizer,” in *IEEE Journal of Solid State Circuits*, vol. 32, pp. 1278–1283, 1997.

[6] C , R., B , A., S , V., and H , P., “A 531nw/mhz, 128 x 32 current–mode programmable analog vector–matrix multiplier with over two decades of linearity,” in *Custom Integrated Circuits Conference*, October 2004.

[7] D ¨ , T., “Bump circuits for computing similarity and dissimilarity of analog voltages,” in *International Joint Conference on Neural Network*, vol. 1, pp. 475–479, July 8-12 1991.

[8] D , J., “On-chip high-voltage generation in nmos integrated circuits using and improved voltage multiplier technique,” in *Journal of Solid-State Circuits*, vol. 11, pp. 374–378, June 1976.

[9] D , J., *Adaptive Analog VLSI Signal Processing and Neural Networks*. PhD thesis, Georgia Institute of Technology, 2003.

[10] D , J. and H , P., “Improved correlation rules in continuously adapting floating-gate arrays using predistortion,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, (Phoeniz, AZ.), pp. 536–539, May 2002.

[11] F , G., “Digital signal processor trends,” *IEEE Micro*, vol. 20, pp. 52–59, Nov–Dec 2000.

[12] F , B., W , J., and A , A., “CMOS analog IC implementing the backpropagation algorithm,” in *Abstracts of the First Annual INNS Meeting*, vol. 1, p. 381, 1988.

[13] H , R., B , J., H , P., M , B., , and D , S., “A cmos programmable analog memory-cell array using floating-gate circuit,” in *IEEE Transactions on Circuits and Systems Special Issue*, vol. 48, p. 4, January 2001.

[14] H , R., B , J., H , P., M , B., and D , S., “A cmos programmable analog memory cell array using floating-gate circuits,” *IEEE Transactions on Circuits and Systems II*, vol. 48, Jan 2001.

[15] H , P., *Foundations of Learning in Analog VLSI*. PhD thesis, California Institute of Techonology, February 1997.

[16] H , P., M , B. A., and D , C., “Adaptive circuits using pFET floating-gate devices,” in *Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI*, (Atlanta, GA), pp. 215–229, March 1999.

[17] H , P., M , B. A., and D , C., “Adaptive circuits using pfet floating-gate devices,” in *Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI*, (Atlanta, GA), pp. 215–229, March 1999.

[18] H , P., M , B. A., D , C., and M , C. A., “An autozeroing floating-gate amplifier,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 2000. in Press.

[19] H , P., “Continuous-time feedback in floating-gate mos circuits,” in *IEEE Journal of Circuits and Systems II*, pp. 56–64, January 2001.

[20] H , P., “Continuous-time feedback in floating-gate mos circuits,” *IEEE Transactions on Circuits and Systems II*, in Press.

[21] H , P. and B , A., “A matrix transform imager allowing high-fill factor,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 337–340, 2002.

[22] H , P. and B , A., “A matrix transform imager and architecture,” in *Proceedings of IEEE Sensors*, pp. 177–182, 2002.

[23] H , P., D , C., and M , B. A., “A four-quadrant floating-gate synapse,” in *IEEE International Symposium on Circuits and Systems*, (Monterey), pp. 29–32, 1998.

[24] H , P., D , C., and M , B. A., “A four-quadrant floating-gate synapse,” in *IEEE International Symposium on Circuits and Systems*, (Monterey, CA), 1998.

[25] H , P., D , C., M , B. A., and M , C. A., “Single transistor learning synapses,” in *Advances in Neural Information Processing Systems 7* (T , G., T , D. S., and L , T. K., eds.), pp. 817–824, Cambridge, MA: MIT Press, 1995.

[26] H , P., K , M., and M , B., “A transistor-only model of the autozeroing floating-gate amplifier,” in *Midwest Symposium on Circuits and Systems*, vol. 1, pp. 157–160, 1999.

[27] H , P., M , B., and D , C., “An autozeroing floating-gate amplifier,” *IEEE Transactions on Circuits and Systems II*, vol. 48, pp. 74–82, Jan 2001.

[28] H , P., M , B., D , C., and M , C., “An autozeroing amplifier using pfet hot-electron injection,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 325–328, May 1996.

[29] H , P., M , B. A., and D , C., “Floating-gate devices: They are not just for digital memories anymore,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. II, (Orlando, Florida), pp. 399–391, 1999.

[30] H , P., S , P., D , C., G , C., D , J., and A , D., “A floating-gate vector-quantizer,” in *Midwest Symposium on Circuits and Systems*, (Tulsa, OK.), pp. 196–199, 2002.

[31] H , M., K , M., and H , P., “5v-only standard 0.5um cmos programmable and adaptive floating-gate circuits and arrays using cmos charge pumps,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2004.

[32] K , D. and S , S. M., “A floating gate and its application to memory devices,” *Bell System Technical Journal*, vol. 46, p. 1288, 1967.

[33] K , R. and W , D., “Semi-parallel rank-order filtering in analog vlsi,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 232–235, Jul 1999.

[34] K , A., “Array-based analog computation: Principles, advantages and limitations,” in *Proceedings of the International Conference on Microelectronics for Neural Networks*, (Lausanne, Switzerland), pp. 68–79, Feb. 1996.

[35] K , M., “Analog programmable filters using floating-gate arrays,” Master’s thesis, Georgia Institute of Technology, 2000.

[36] K , M., H , P., D , J., and A , D., “Programmable and adaptive analog filters using arrays of floating-gate circuits,” in *Proceedings of the Conference on Advanced Research in VLSI*, pp. 148–162, 2001.

[37] K , M., L , A., and H , P., “A programmable continuous-time analog fourier processor based on floating-gate devices,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. III–351, 2000.

[38] K , M., L , A., H , P., and N , J., "Programmable continuous-time floating-gate fourier processor," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, pp. 90–99, January 2001.

[39] K , M., L , A., H , P., and N , J., "A programmable continuous-time floating-gate fourier processor," *IEEE Transactions on Circuits and Systems II*, vol. 48, pp. 90–99, Jan 2001.

[40] K , S., *VLSI array processors*. Englewood Cliffs, N.J.: Prentice Hall, 1988.

[41] L , J., R , S., M , M., and M , C., "Winner-take-all networks of O(N) complexity," in *Advances in Neural Information Processing Systems* (T , D., ed.), vol. 1, pp. 703–711, San Mateo, CA: Morgan Kaufman Publishers, 1988.

[42] L , M. and S , E. H., "Fowler-nordheim tunneling in thermally grown sio_2," *Journal of Applied Physics*, vol. 40, p. 278, 1969.

[43] L , A. and H , P., "Cadence-based simulation of floating-gate circuits using the ekv model," in *Midwest Symposium on Circuits and Systems*, vol. 1, (Las Cruces, NM.), pp. 141–144, August 1999.

[44] L , J. and C , G., "A micropower learning vector quantizer for parallel analog-to-digital data compression," in *International Conference on Circuits and Systems*, 1998.

[45] M , C., *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[46] M , B. and H , P., "A floating-gate technology for digital cmos processes," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 400–403, 1999.

[47] N , J., M , B. K., B , E. A., D W , S. P., and H , P., "A cmos coupled nonlinear oscillator array," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 301–304, 2002.

[48] N , J., *Controlled stochastic resonance and nonlinear electronic circuits*. PhD thesis, Georgia Institute of Technology, 2000.

[49] P , L., "Charge pumps: An overview," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2003.

[50] S -M , J., "A switched capacitor double voltage generator," in *Midwest Symposium on Circuits and Systems*, vol. 1, pp. 177–180, 1994.

[51] S , P. and H , P., "Low-power speech recognition using analog signal processing techniques," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, (Orlando, FL), pp. 3988–3991, May 2002.

[52] S , P., K , M., and H , P., “Mel-frequency cepstrum encoding in analog floating-gate circuitry,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 671–674, 2002.

[53] S , P. D., K , M., and H , P., “Accurate programming of analog floating-gate arrays,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2002.

[54] V -H , J., H , P., W , D., D , L., G , G., and M , H. E., “Himos - a high efficiency flash eeprom cell for embedded memory applications,” in *IEEE Transactions on Electron Devices*, vol. 40, p. 2255, 1993.

[55] W , D. and D , S., “Rank-order filtering in analog vlsi,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 105–108, May 1996.