A Fully Integrated Architecture for Fast and Accurate Programming of Floating Gates Over Six Decades of Current

Arindam Basu, Student Member, IEEE, and Paul E. Hasler, Senior Member, IEEE

Abstract—This paper presents an on-chip system with digital serial peripheral interface (SPI) interface that enables accurate programming of floating gate arrays at a high speed. The main component allowing this speedup is a floating point current measuring analog-to-digital converter (ADC). The ADC comprises a wide range logarithmic transimpedance amplifier (TIA) followed by a linear ramp ADC. The TIA operates over seven decades of current going down to sub-pA levels. It incorporates an adaptive biasing scheme to save power. The topology provides a relatively temperature independent measurement of the floating-gate voltage. The TIA-ADC combination operates over six decades at a thermal noise limited accuracy of 9.5 bits when average conversion time is around 500 μs. The system features level-shifters and selection circuitry at the periphery of the floating gate array, current-steering digital-to-analog converters (DACs) to set gate and drain voltages, and SPI for a microprocessor or field-programmable gate array (FPGA). Algorithms using either pulse-width modulation or drain voltage modulation can be implemented on this platform. We present data for this system from 0.5 μm AMI and 0.35 μm TSMC processes.

Index Terms—Floating-gate programming, floating-point analog-to-digital converter (ADC), hot-electron injection, logarithmic compression, low power, programmable analog.

I. FLOATING GATE PROGRAMMING

FLOATING-GATE transistors have been used in many VLSI systems as multilevel digital memories, neural network synapses or reconfigurable switches in field programmable arrays. We present a generic architecture for programming floating-gates over a wide range of currents at moderate accuracy and speeds. Moreover, the fully digital interface allows easy integration of the floating-gate chips in a larger embedded system.

Fig. 1(a) and (b) show two different scenarios for programming. In the first case, a bandpass filter is shown with corner frequencies set by floating gate current sources, which need to be programmed accurately. In the latter case though, the floating gates are used as programmable interconnects in a large scale programmable digital or analog array. Here, they need to be programmed to a large but arbitrary current.

Applications of Floating gates: (a) A bandpass filter with programmable corner frequencies that need accurately tuned currents. (b) Floating gates are used as programmable interconnect in the switch matrix of the FPAA IC [1] which need to be programmed to a large but arbitrary current.

References [1] to [5] are discussed in detail with measurement results being provided in [5] without much details about its features and performance. This paper provides complete measurement results of the programming system from 0.5 and 0.35 μm CMOS process presented in [4]. The integration of this subsystem in a FPAA was described in [5] without much details about its features and performance. This paper provides complete measurement results of the programming system from 0.5 and 0.35 μm CMOS and describes final accuracy and dynamic range achieved in programming floating-gates by this method.

Sections II and III discuss the terminology and architecture of the chip. In Section IV, the different sub-circuits in the system are discussed in detail with measurement results being provided in each subsection. Section V presents data from programming floating-gate elements using the infrastructure described earlier. In Section VI, we discuss data retention ability of floating-gates, temperature sensitivity of the programming and total programming time. Finally, we compare our approach with others and draw conclusions in Section VII.
II. OVERVIEW OF PROGRAMMING FLOATING GATES:
FIRST PRINCIPLES

Floating-gates can be programmed by both Fowler–Nordheim tunneling and hot-electron injection processes. In this system, tunneling is used as a global erase while hot-electron injection is employed for fast, accurate programming of these elements. Fig. 2(a) depicts a floating gate with its terminals marked. The process of accurately programming an FG-transistor consists of two distinct phases: “measure” when the current through the devices in the array are measured and “inject” when the devices are injected to reach their desired targets. To inject a device, all the terminal voltages, i.e., $V_{dd}$, $V_d$, $V_g$, and $V_{th}$ are raised to a value higher than the normal operational value but their relative values are kept same. This process is referred to as ramp-up. The high electric field necessary for injection is produced by pulsing the drain to a lower voltage for a certain time ($t_{pulse}$).

Fig. 2(b) shows an array of these elements. To select a device, an enabling voltage is applied to its gate and drain terminals. All other elements have either of the gate or drain voltages turned to $V_{dd}$ thus prohibiting injection. This condition is a direct application of the fact that ample source current and large drain-channel potential are both necessary for hot-electrons to inject onto the gate. In the chip fabricated in 0.35 µm, the source current is cut off in the non-selected devices by an explicit switch. Fig. 2(c) shows the data flow in the automated programming system. The digital word corresponding to the present current of one possible programming algorithm, details of which are available in [3]. This algorithm modulates the drain-source potential while maintaining a fixed pulse width. It is shown in [3] that the change in FG current due to injection can be modelled as

$$\log(\Delta I) = m(I_{init})V_{th} + f(I_{init})$$ (1)

where $I_{init}$ is the current in the FG device prior to injection, $\Delta I$ is the change in the FG current after injection and $m$, $f$ are polynomials (typically quadratic). Initially, a random set of FG devices in the chip are subjected to hot-electron injection for different $V_{th}$ values to obtain a mean characteristic. This function can now be inverted to obtain a desired $V_{th}$ for a certain value of $I_{target}$ and $I_{init}$. To ensure that the FG current does not exceed $I_{target}$, at every step either $V_{th}$ or $t_{pulse}$ is reduced from the computed one by a factor determined by mismatch between devices. This leads to a tradeoff between programming time and accuracy/mismatch. A similar algorithm may be formulated for pulse width modulation.

III. ON-CHIP PROGRAMMING: ARCHITECTURE AND TIMING

In this section, we discuss the architecture of the on-chip programming system. The system has been tested as a separate chip and also as part of a larger FPAA IC. The architecture is general and can be employed to program floating-gates in other systems as well.

Fig. 3(a) shows the generic architecture of a system to program an array of floating-gates. The floating-gate(s) to be currently programmed are selected by applying a digital word to the selection circuitry on the periphery of the array. The selection may be done one at a time or a row at a time or in any other parallel fashion as desired. The selection circuit passes the desired gate and drain voltages to the selected FG device, while the gate and drain terminals of all the rest are set to $V_{dd}$. The source of all floating-gate transistors is tied to $V_{dd}$ and do not have any selection mechanism. The tunneling voltage connection that goes to all FG devices is not shown in this figure. The gate and the drain DACs supply the desired voltages to the gate and drain of the FG elements and are controlled digitally through an SPI interface. While the drain DAC is used only during injection, the gate DAC is used in both programming and operational mode.

The measurement of the charge on the gate is accomplished by measuring the current through the device using a logarithmic transimpedance amplifier (TIA). The logarithmic compression allows the TIA to measure currents varying over several decades in magnitude. The amplifier maintains stability without dissipating excessive power by employing an adaptive biasing scheme that will be described later. The output of the amplifier is low-pass filtered and then digitized using a ramp ADC. The ramp topology is chosen because of its linearity and ease of implementation. The combination of the logarithmic
TIA and the linear ADC form a floating-point ADC as will be explained later.

The four major control signals determining the operation of the architecture are described as follows.

1) **PROG**: This signal being high indicates that the FG elements in the chip are being programmed to the desired value. When it is low, the chip is in operational or RUN mode.

2) **MEASURE**: This signal defines a sub-mode for PROG mode. MEASURE being high indicates that the system is in “measure” mode, i.e., currents of the programmed gates are being measured using the floating point ADC. On the other hand, MEASURE being low signifies that the system is in “inject” mode, i.e., gates are being injected to reach the desired target values.

3) **PULSE**: This signal is high when the floating gates are being injected. The pulse-width of this signal determines the time, \( t_{\text{pulse}} \), for which the selected gate is injected in the current cycle.

4) **SWC**: This signal is asserted high when the chosen floating gate needs to be programmed as an ON switch, i.e., it needs to be programmed to an arbitrary low floating gate voltage. Fig. 3(b) shows the system timing diagram for programming “N” FG elements accurately. After PROG is asserted high (signaling the beginning of programming mode), a tunneling pulse is used to globally erase all the array elements. SWC is then asserted low indicating accurate injection mode followed by selection of the desired element or row of elements. The “measure” phase begins first where the charge on all the floating gates are measured. The MEASURE signal needs to toggle for every element as it marks the beginning of the ADC conversion cycle. After the “measure” phase, the chip is ramped up followed by “N” short pulses on the signal PULSE for injecting the FG device. This is followed by ramping down the chip followed by another “measure” cycle and so on.

For programming the switches, the control sequence is simpler as measurements are not needed. In that case, SWC is asserted high to indicate switch programming mode and MEASURE is kept low throughout the process. The rest of the signaling is as described earlier.

**IV. ON-CHIP PROGRAMMING: COMPONENTS**

In the last section, the architecture and global signaling scheme was detailed. In this section, the major components of the system, i.e., the drain selection block, the drain and gate DACs, the logarithmic TIA and the ADC are discussed along with measured results from 0.5 and 0.35 \( \mu \text{m} \) chips.

**A. Drain Selection**

The drain selection circuitry as shown in Fig. 4 acts as a second selection level after the desired floating-gate drain terminal has been selected by multiplexors. This block switches the selected drain to injection or measurement sub-circuits depending on the programming mode. If the system is in measure mode, the drain is connected to the transimpedance amplifier. In inject mode, if PULSE is low, the selected drain is tied to \( V_{\text{dd}} \) thus prohibiting injection. When PULSE is high, the selected drain is switched to the drain DAC or to ground depending on the polarity of the signal SWC. This is because for switch programming, it is always desirable to have the maximum \( V_{\text{ds}} \) across a selected device, while in accurate programming, the \( V_{\text{ds}} \) is modulated depending on difference from target current.

**B. Gate and Drain DAC**

The gate and the drain DACs share a binary current scaled architecture as shown in Fig. 5(a). The reason for this choice was the low required resolution of 7 bits for either DAC. We do not need very high resolution for the drain DAC since we can tradeoff the time needed for injection with the number of possible drain voltage levels. The gate DAC’s resolution can also be low, since, it is used to set the operating regime of the...
FG transistor being injected within a range of subthreshold currents that correspond to high injection efficiency. The current sources are cascoded PFETs biased by a PTAT bootstrap current source while the cascode transistors are biased by the structure described in [6]. The sizing of the current source array was done following [7] and the references therein. To guarantee operation, the devices were chosen large enough to satisfy 8 bit matching. Dummy devices were employed to eliminate systematic mismatch. The resulting area of the DACs is around 50% of the entire area of the programming infrastructure. A differential pair is used to switch the currents to increase switching speed. A latch is used to convert the 3.3 V digital signals to smaller voltage swings with a low crossing point [7] so that the ON transistor of the differential pair is in saturation.

The drain DAC needs to provide voltages close to ground and hence the currents are directly passed into a resistor. The gate DAC, on the other hand, needs to provide voltages close to the programming while it is powered from a separate 3.3 V supply that is common to the programming circuit. So the current was mirrored using a cascoded NFET mirror and passed into a resistor referenced to the programming.

Digital words for the DACs are loaded into a shift register from the digital controller through a SPI interface.

Fig. 5(b) shows the measured DNL and INL from the DAC structures fabricated in a 0.35 µm chip. The matching of the transistors was better than 7 bits as expected.

C. Adaptive Logarithmic Transimpedance Amplifier

Fig. 6(a) shows the transimpedance structure that has a variable resistance in the feedback path across an amplifier. The amplifier used in this case is a simple five transistor OTA. The detailed analysis of this structure can be found in [8], but for the sake of completeness the salient features of the design are mentioned here. The amplifier tries to hold the input node constant, forcing the current to flow through the feedback transistors M1 and M2 and reducing the current through the capacitance at the input node.

To measure a wide dynamic range of currents spanning several orders of magnitude, a fixed feedback resistance should have a very small value, which poses an SNR issue at low currents. On the other hand, a MOS transistor changes its resistance depending on the current flowing through it. To increase the sensitivity of the conversion, M2 is used as a source degeneration...
for M1. This results in replacing $\kappa$ of M1, where $\kappa$ is the inverse of the subthreshold slope [9], by an effective $\kappa_{\text{eff}}$ of the combination which is lesser and is equal to $\kappa^{2}/(\kappa + 1)$. The current to voltage relation is given by

$$V_{\text{out}} = \frac{V_{\text{ref}}}{\kappa^{2}} - \frac{U_{T}}{\kappa_{\text{eff}}} \ln \left( \frac{I_{\text{in}}}{I_{F}} \right)$$

(2)

where $U_{T}$ is the thermal voltage and $I_{F}$ is the pre-exponential factor in the I-V relation of a sub-threshold pMOS. Small-signal analysis yields the dominant poles in the typical case where $C_{||} \gg C_{F}$ to be

$$p_{1} \approx -\frac{A \cdot G_{\text{m}}}{C_{||}} \quad p_{2} \approx -\frac{G_{\text{ob}}}{C_{F} + C_{L}}$$

(3)

where $A$ is the amplifier’s gain, $G_{\text{m}}$ is the transconductance of the feedback transistor (we can consider the combination of M1 and M2 acting effectively as one transistor) and $G_{\text{ob}}$ is the output conductance of the amplifier.

The major issue in this implementation is to maintain stability, the amplifier’s output pole should be at a frequency that is sufficiently higher than the pole at the input due to $C_{||}$ and any parasitic/explicit feedback capacitor $C_{F}$. To achieve this, most approaches burn excessive power because the amplifier is biased with currents that are much higher than the highest input current to guarantee stability in the entire operating range of currents. In our implementation, M3 and M4 replicate the input current, while M5 mirrors the current into the amplifier’s bias with a gain. Thus we burn less power in the amplifier when the input current is low.

The two possible issues with the added feedback loop for adaptation is instability and extra noise. However, note that the adaptation current is a common-mode input to the differential amplifier. Hence, any possibility of oscillation is nullified by the high CMRR of the differential gain stage. Similarly, noise contribution of the adaptation loop is also reduced by the common-mode rejection properties of the differential amplifier. Measurements confirming these facts are presented in [8]. The additional current source $I_{\text{ref}}$ ensures that the adaptation loop is always biased at a base current level allowing a minimum speed of operation. It can be shown [8] that the average power dissipation of this adaptive structure is smaller than that of the non-adaptive case by a factor of $\sqrt{DR}$, where DR is dynamic range or the ratio of the largest and smallest input currents.

An advantage of this topology is that when the TIA is measuring currents from a floating gate, its output voltage is effectively measuring the floating-gate potential. This leads to the following expression for the output voltage of the TIA:

$$V_{\text{out}} = \frac{V_{\text{ref}}}{\kappa^{2}} - \frac{V_{d}}{\kappa_{\text{eff}}} - \frac{U_{T}}{\kappa_{\text{eff}}} \ln(M)$$

(4)

where $V_{d}$ is the floating-gate potential and the FGMOS has an aspect ratio that is $M$-times the aspect ratio of the feedback transistors. Thus by appropriately sizing the feedback transistor, the output of the TIA can be made temperature insensitive to a first order since the charge on the floating-gate does not change appreciably with temperature [10].

To evaluate the noise performance of the circuit, we denote the four noise-contributing transistors (excluding the tail current source) in the five-transistor amplifier by $M_{\text{amp,k}}$ with transconductances equal to $G_{\text{amp}}$. Also, the transconductances of M1 and M2 are denoted by $G_{fb}$. M3–M5 do not contribute much noise as explained earlier. Then the input referred noise current can be shown to be

$$\gamma_{\text{in}}^{2} = \left( \frac{\kappa}{\kappa + 1} \right)^{2} \left( \gamma_{1}^{2} + \gamma_{2}^{2} \right) + \left( \frac{\kappa}{\kappa + 1} \right)^{2} \left( \frac{G_{fb}}{G_{\text{amp}}} \right)^{2} \left( \sum_{k=1}^{4} \gamma_{\text{amp,k}}^{2} \right)$$

(5)

From (5) it can be seen that since the adaptation makes $G_{\text{amp}} \gg G_{fb}$, the noise contribution of the amplifier is negligible (measured results supporting this are shown in [8]). The SNR of this structure can now be computed by integrating this expression over frequency. Ignoring 1 over $f$ noise, we get

$$\text{SNR}_{\text{power}} = \left( \frac{\kappa}{\kappa + 1} \right)^{2} \frac{G_{\text{eff}} U_{T}}{q}$$

(6)

where $G_{\text{eff}}$ is $C_{||} / A + C_{F}$. This evaluates to 45 dB using value of $C_{\text{eff}}$ extracted from the measured frequency response. Here the integrated noise is independent of current because as $I_{\text{in}}$ increases, the noise spectral density reduces while the bandwidth increases. However, since we are interested in a fixed bandwidth, using a filter with cut off at $f_{BW}$, the SNR equation becomes

$$\text{SNR}_{\text{power}} \approx \left( \frac{\kappa}{\kappa + 1} \right)^{2} \frac{I_{\text{in}}}{q f_{BW}}$$

(7)

The actual SNR does not keep increasing as predicted by (7) since it gets limited by 1 over $f$ noise. In this implementation the TIA is followed by a $G_{m} = C$ low-pass filter to limit the bandwidth.

The difference in the structure here as compared to the standalone one described in [8] is the added multiplexors and diode connected PFETs M6 and M7. These were added based on system considerations since the feedback circuit described earlier loses stability at very high currents. So, M6 and M7 is kept as a coarse I-V converter for high currents. The output of the two converters are multiplexed based on the signal SWC as only transistors programmed as switches might produce such high currents. The current source $I_{\text{amp}}$, implemented by a pMOS, is kept to bias the circuit when it is not measuring currents. The source or gate of the PFET is controlled by a DAC on the board and is also used for characterizing the performance of the TIA.

Fig. 6(b) shows measured characterization data from both 0.5 and 0.35 $\mu$m IC designs. Fig. 6(b-I) shows the I-V relation for a logarithmic amplifier fabricated in a 0.5 $\mu$m process. The input current was created by sweeping the gate voltage of the pMOS used to create $I_{\text{amp}}$ in Fig. 6(a). The deviation from logarithmic relation at high currents is due to transistors entering above threshold region of operation. At low currents, the off-chip pico-ammeter (Keithley 6485) loses accuracy. Fig. 6(b-II) shows the same data but plotted against the gate voltage of the PFET used to create $I_{\text{amp}}$ on the X-axis. The logarithmic relation is now maintained for very low currents too (high gate voltage). This conclusively shows that the logarithmic TIA is more accurate than the off-chip pico-ammeter for lower currents. Fig. 6(b-III) shows similar data from a 0.35 $\mu$m FPA.
chip. Here, the measured data between currents of 1 to 10 nA is used to fit a polynomial to the logamp characteristic. Using this curve-fit, the output voltage of the logamp is used to infer measured currents and compared with the pico-ammeter measurements. While the off-chip measurements saturate at around 100 pA due to noise and leakage from ESD diodes, the on-chip measurement can go down to sub-pA levels proving its utility. Conformance to logarithmic behavior can be measured by fitting a line to the characteristic. The average error is 3.1% (considering currents less than 1 μA) because of the feedback transistors entering the above threshold regime at high currents. However, using a second order fit, the error reduces below 1% for the same range. Using a high order fit for the I-V conversion is not a problem since this can be done on a PC before sending the target voltages/codes to the μP.

D. Ramp ADC

The ADC in this system acts as the interface between the TIA and the digital controller. As the settling time of the TIA for sub-pA currents is of the order of a few msec, the conversion time requirement for the ADC is also relaxed. This led to the choice of a ramp ADC architecture as shown in Fig. 7(a) because of its simple structure. In Fig. 7(a), \( M \) denotes the control signal MEASURE starting the conversion and \( V_{\text{start}} \) is the starting voltage for the ramp. The comparator trips after the ramp generated by the current source, \( I_{\text{ramp}} \) crosses the input voltage freezing the counter. When the conversion starts, there is a shift in the start voltage of the ramp due to charge injection from the switch. But this is signal independent and hence can be treated as an offset. It can be taken care of by either offsetting \( V_{\text{start}} \) (analog trim) or by subtracting the digital word corresponding to the offset (digital trim). The accuracy of the ramp is limited by the early effect of the cascode current source used. The biasing of the cascode is done following [6] while the current source is biased using a bootstrap current source chosen because of ease of implementation and relatively low temperature dependence (PTAT in subthreshold operation). It can be replaced by a lower TC current source in the future. The comparator in Fig. 7(b) is a simple high gain amplifier comprising a differential pair followed by a push-pull output stage. The gain of the comparator will be increased in future versions by employing cascode transistors in the output stage.

Fig. 7(c) shows the details of the timing of the digital interface for the ADC. Once the output of the comparator, FREEZE goes high, the counter has the valid digital word at its output. The chip-select signal, \( \text{ADC.CS.N} \) is then asserted low and the serialized data is read out from \( \text{ADC.SDO} \). Once the SPI data transfer is completed, MEASURE is asserted low and the ADC is ready for the next input. In this implementation, a 14 bit counter has been used. The FREEZE output is also buffered out allowing it to be used to control off-chip counters on the μP which can be 32 bits long.

Fig. 7. ADC Schematic: (a) Simplified schematic of the ramp ADC. The signal M represents MEASURE. The output of the counter is passed to the controller through a SPI interface. (b) The comparator is a nine transistor OTA. (c) The timing diagram of the ADC. (d) Measured input-output relation of the ADC in 0.35 μm CMOS. The input of the ADC was set by passing a certain current through the TIA. (e) Measured resolution of the TIA-ADC combination is plotted against varying ramp current, \( I_{\text{ramp}} \).
The effective resolution for the TIA-ADC combination $N$ can be found by relating it to the dynamic range. Let the logarithmic amplifier’s characteristic be given by $V = \text{Offset} + K \times \log(I)$ and $\delta V$ be the voltage noise level at the output of the logamp. Then denoting the input referred noise current to be $\delta I$ when the dc input current is $I$, $\delta V$ can be related to the SNR of the input as

$$\delta V = K \frac{\delta I}{I} = \frac{K}{\text{SNR}},$$

Then, using (8), dynamic range $(DR_{\text{ADC}})$ of the ADC is given by

$$DR_{\text{ADC}} = 2^N = \frac{V_{\text{max}} - V_{\text{min}}}{\delta V}$$

$$= \frac{K \log \left( \frac{I_{\text{max}}}{I_{\text{min}}} \right)}{\delta V}$$

$$= \text{SNR} \times \log(DR)$$

where $DR$ is the dynamic range of input currents. This equation explicitly shows the floating point nature of the system as the SNR sets the mantissa bits while $\log(DR)$ sets the exponent. The number of exponent bits are around 3–4 for 5–7 decades of current. The clock frequency can be decided considering the worst case conversion time to be $2^N \times T_{\text{clk}}$, where $N$ is the number of bits and $T_{\text{clk}}$ is the clock period. The maximum frequency of the clock available from the $\mu P$ was around 20 MHz, leading to $T_{\text{clk}} = 50$ ns. For $N$ equal to 14, the worst case conversion time is around 1 ms and average conversion time is around 512 $\mu$s. The value of the current source can then be chosen using the following equation:

$$V_{\text{LSB}} = \frac{I \times T_{\text{clk}}}{C}$$

where $V_{\text{LSB}}$ is decided by the noise at the output of the I-to-V and C is 5 pF.

Fig. 7(d) shows measured transfer characteristic of the ADC with a 32 bit counter implemented on the $\mu P$. This was done to examine the effect of finite register length on the ADC output shown in 7(e) and explained later. The input voltage of the ADC is swept by varying the current through the TIA. In this chip, the digital trim option was preferred and $V_{\text{start}}$ was set to ground to avoid using an extra pin. Thus, a part of the counter’s range was sacrificed in the process as the output of the TIA does not start from ground. This is evident from the count not starting from zero in the figure. It has been found that even though the TIA can measure currents down to sub-pA levels (found by monitoring the voltage output), the ADC cannot convert inputs lower than 6 pA reliably. This is traced back to the fact that the digital signal starting the conversion also starts the measurement phase for the TIA. For very low currents, the TIA output does not settle in time. This has been rectified in future version by having separate digital controls and employing a bidirectional logamp for faster settling.

The effect of reducing the quantization noise by slowing the ramp has also been studied. Fig. 7(e) shows the measured resolution corresponding to different ramp currents, $I_{\text{ramp}}$. In this experiment, the current through the TIA is set to a fixed value producing a fixed voltage at the input of the ADC along with some noise. For a particular $I_{\text{ramp}}$, this input is digitized multiple times and the ratio of the mean of the codes and their standard deviation is considered as SNR for computing the effective resolution. It is seen for very large $I_{\text{ramp}}$, the LSB voltage step of the ADC is too large leading to a quantization noise dominated performance. On the other hand, for very slow ramps, the number of bits in the counter, CNT, is the limiting factor giving poor dynamic ranges. In between these two regimes, performance is limited by the noise of the system contributed primarily by the noise in the log-amplifier’s output and in the current source generating $I_{\text{ramp}}$. 9.5 bits of performance corresponds to a rms noise of around 1.5 mV at the output of the TIA and around 80 $\mu$V on the floating gate. Performance can be improved up to 11 bits by averaging a number of readings. This is done in FG programming when the measured current is close to the desired current.

![ADC Linearity](image.png)

Fig. 8 plots the INL of the ramp converter with respect to the LSB of a 9 bit ADC. The finite output resistance of the current source leads to nonlinearity in the slope of the converter. To ensure that the error in linearity of the ramp is less than 0.5 LSB, digital correction is used. The entire range of codes is divided into four sections and a separate gain and offset correction factor for each section is stored in the $\mu P$. To predict the correct input voltage for a certain code, one of these four gain and offset factors are used based on the range in which the code falls. Dividing the range into more subsections can improve INL performance even more, but is not needed since noise dominates the ADC performance in that case.

V. FLOATING-GATE MEASUREMENTS

In this section, we describe system test results showing measured change in floating-gate charge and using that information to program a desired amount of charge. Fig. 9 shows a die photo of the $3 \times 3$ $mm^2$ chip fabricated in 0.35 $\mu$m CMOS with the programming infrastructure occupying $1050 \times 250 \mu m^2$. A close up of the layout of the programming circuits is also shown. Fig. 10 shows data from an experiment where the FG device (in 0.5 $\mu$m CMOS) was subjected to 6 $V_{\text{clk}}$ pulses of pulse width equal to 100 $\mu$s. The current was monitored after every pulse using both the on-chip TIA and the off-chip ammeter. From the measurements, it is obvious that while the TIA can distinguish very fine
Fig. 9. Die Photo: Die photo and layout of the 0.35 \( \mu m \) chip showing the different sub-blocks of the programming infrastructure.

amount of hot-electron injection, the ammeter cannot. The left axes shows change in voltage from starting point while the right axes of the plot shows number of injected electrons based on a gate capacitor of value 750 fF.

Next we show programming floating-gates to specific target currents that span a wide range in magnitude. A set of forty floating-gates are programmed to currents ranging from approximately 6 pA to 20 \( \mu A \) using a version of the algorithm shown in [3]. The experiment was run fifteen times choosing a random set of devices from a pool of over thousand devices. The average of the absolute error in achieving the current targets is plotted in Fig. 11(a). The average error is 2.14\% for this range of currents and reduces to below 1\% if currents higher than 100 pA are considered. There could be several reasons for this error. First, there is an error associated in modeling the injection process over different source currents and \( V_{DS} \) values by a polynomial. This error is magnified particularly when the dynamic range is large. Second, there is a spread of the parameter values for injection across a large number of devices. Both these errors can be handled by slowing down the rate at which the target is approached and averaging the measurements to reduce noise. The increased error at low currents is primarily due to more noise, a property associated with logarithmic amplifiers followed by a fixed bandwidth low-pass filter [8]. This issue can also be addressed by averaging the measurements and trading speed for accuracy. Fig. 11(b) shows the error when a set of thirty different devices were programmed to a current of 100 nA while Fig. 11(c) depicts the resulting error when a single device is tunneled and programmed for twenty times to achieve a target of 100 nA every time. The source of this error is also the noise in the current being measured added to the noise in the measurement.

VI. DISCUSSION

A. Long-Term Storage

The data retention capability of floating-gate transistors have been reported in multiple publications [11]–[13] and exhibits insignificant charge loss in tens of years. Both short-term and long-term drift of charge in our floating-gate devices have been shown to be less than 0.2\% over 16 days [14]. Moreover, the charge drift has been observed to reduce after an initial bake in the oven at elevated temperatures [13]. Finally, the effect of charge drift on circuit performance depends on the topology of the circuit and might be reduced if differential measurements are taken.

B. Temperature Dependence

The output voltage of the logarithmic TIA represents the floating-gate voltage and hence is relatively temperature independent even if the floating-gate current varies. Intuitively, the temperature behavior of the logarithmic amplifier is opposite of that of the FGMOS and exactly cancels the change in floating-gate currents. In practice, there is a minor temperature variation due to mismatch in sizes of the actual FGMOS and the feedback transistor in the logamp. In large systems using floating-gates, a floating-gate current reference, such as the one in [13] can be used to bias the array for temperature insensitive currents. The reference itself can be programmed based on floating-gate voltage measurements, which as we mentioned is relatively temperature insensitive.

C. Programming Time

The time needed to program an FGMOS device comprises the time needed to ramp the voltages (\( t_{\text{temp}} \)), injection pulse time (\( t_{\text{pulse}} \)), measurement time (\( t_{\text{mes}} \)) and time to transfer digital bits through SPI (\( t_{\text{SPI}} \)), all of which gets multiplied by
the number of pulses needed to achieve the target ($N_{\text{pulses}}$). So, we can write

$$T_{\text{prog}} = (t_{\text{ramp}} + t_{\text{pulse}} + t_{\text{mens}} + t_{\text{SPI}}) \times N_{\text{pulses}},$$

(11)

For the present implementation, average values of $t_{\text{ramp}} + t_{\text{SPI}}$ is limited to 0.8 msec. The ramp process is done in small steps to allow the bulk to stabilize and to prevent any chances of latchup. For programming large arrays, the time for ramping voltages up and down would be common to the whole array and its effect on total programming time shall be reduced. Average values of $t_{\text{pulse}}$ and $t_{\text{mens}}$ are 0.1 and 0.5 ms, respectively. The ADC conversion time can be reduced by employing larger ramp currents and higher clock speeds, the current limitation being set by the 20 MHz processor used. However, the bottleneck for measurement time is not the ADC conversion time but is the settling-time of the log-TIA for smaller currents. The average measurement time, though, can be reduced by using larger ramp currents while measuring larger target currents since the settling time of the log-TIA is reduced in those cases and is no longer the bottleneck. The average value for $N_{\text{pulses}}$ is around 35 for the simple programming scheme used. Hence, the average programming speed achievable is around 20 devices/s. It should be noted that the value for $N_{\text{pulses}}$ depends on the algorithm used and $t_{\text{mens}}$ depends on the accuracy desired (averaging might be necessary). We expect $N_{\text{pulses}}$ to reduce to around 25 with a better algorithm.

D. Future Improvements

In the current implementation, a large fraction of $N_{\text{pulses}}$ is needed to come within range of the desired target current. This might be improved dramatically by using a method similar to the one described in [16] for coarse programming. Moreover, we only store one set of injection parameters for the whole array in the LUT. Hence, to account for mismatch in the parameters, at every step the applied pulse width and $V_{\text{th}}$ value are reduced to a fraction of the actual one needed to achieve the target current. This leads to an increase in the number of required pulses. This can be avoided if characterization parameters are stored for each device in the array.

The other consideration in using an architecture like this might be the area overhead for the programming infrastructure. If only a few floating-gates are being used, using this whole infrastructure might be prohibitive. In that case, a simpler method might be using off-chip control and measurements. For a production environment, the tester time cost has to be compared with the cost for chip area to make a decision regarding this.

VII. Conclusion

Though initially the floating-gate device was used as a digital storage, in the recent past, there have been numerous instances of its usage in traditional analog applications such as data converters [18], imagers [19], analog memory [20], offset cancellation in amplifiers [12], low TC current references [10], and many more. This trend requires the accuracy and speed of programming the charge on the gate to increase drastically. A fully integrated architecture for programming floating-gate based systems with high accuracy, moderate speed and low-power is described in this paper. It achieves better dynamic range by utilizing a floating-point ADC that has a logarithmic transimpedance amplifier as a first stage followed by a linear ADC.

Table I presents a comparison of this work with other reported implementations. In Table I, “CHE injection” refers to channel hot-electron injection. It should be noted that the accuracy of this implementation increases to around 11 bits with averaging. The errors in modelling injection can be reduced if the dynamic range of operation is restricted to sub/above threshold regions, a fact validated by the results in [3]. This implies our implementation can be scaled to 12–13 floating point bits with around 9 bits of mantissa.

We present measured results of programming currents spanning more than six decades at speeds higher than 1 ms per measurement and average accuracy better than two percent. For programming switch elements, the time needed is around 100 $\mu$s per row of elements. The architecture is general and provides a solution to programming any system where a large number of floating-gates are required.
TABLE I

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Method</th>
<th>Range</th>
<th>Programming Accuracy</th>
<th>Programming Time</th>
<th>Level of integration</th>
<th>Array Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>This work</td>
<td>Discrete pulses, CHE injection</td>
<td>6 pA - 20 µA ($\Delta V_{fg} \approx 0.63$ V)</td>
<td>9.5 bits (floating point)</td>
<td>50 ms</td>
<td>Fully on-chip</td>
<td>Enabled</td>
</tr>
<tr>
<td>[3]</td>
<td>Discrete pulses, CHE injection</td>
<td>500 pA - 1 µA ($\Delta V_{fg} \approx 0.3$ V)</td>
<td>9 bits</td>
<td>-</td>
<td>Only I-V on-chip</td>
<td>Enabled</td>
</tr>
<tr>
<td>[15]</td>
<td>Discrete pulses, CHE injection, Differential only</td>
<td>$\Delta V_{fg} = 2$ V</td>
<td>10 bits</td>
<td>50 ms</td>
<td>Off-chip</td>
<td>No</td>
</tr>
<tr>
<td>[16]</td>
<td>Continuous time, CHE injection</td>
<td>10-640 pA ($\Delta V_{fg} \approx 0.13$ V)</td>
<td>&lt; 8 bits</td>
<td>-</td>
<td>Off-chip</td>
<td>Enabled</td>
</tr>
<tr>
<td>[17]</td>
<td>Pulse-width modulation, Tunneling</td>
<td>$\Delta V_{fg} = 1$ V</td>
<td>6.5 bits</td>
<td>75 µs</td>
<td>Fully on-chip</td>
<td>No</td>
</tr>
</tbody>
</table>

REFERENCES


Mr. Basu was a recipient of the Prime Minister of India Gold Medal in 2005 from I.I.T. Kharagpur, the JBNSTS Scholarship in 2000, the Best Student Paper Award in Ultrasystems Symposium 2006, and was nominated for the Best Student Paper Award in ISCAS 2008.

Mr. Basu received the B.Tech. and M.Tech. degrees in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in 2004 and 2005, respectively. He is currently pursuing the Ph.D. degree in electrical engineering from the Georgia Institute of Technology, Atlanta.

His research interests include nonlinear dynamics and chaos with applications to modelling neurons, low power programmable analog IC design, and bio-inspired circuits.

Paul E. Hasler (S’87–M’95–SM’04) received the M.S. and B.S.E. degrees in electrical engineering from Arizona State University, Tempe, in 1991, and the Ph.D. degree in computation and neural systems from California Institute of Technology, Los Angeles, in 1997.

He is an Associate Professor with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta. His current research interests include low power electronics, mixed-signal system ICs, floating-gate MOS transistors, adaptive information processing systems, "smart" interfaces for sensors, cooperative analog-digital signal processing, device physics related to submicrometer devices or floating-gate devices, and analog VLSI models of on-chip learning and sensory processing in neurobiology.

Dr. Hasler was a recipient of the NSF CAREER Award in 2001, the ONR YIP Award in 2002, the Paul Raphorst Best Paper Award, IEEE Electron Devices Society, 1997, CICC Best Student Paper Award in 2006, ISCAS Best Sensors Paper Award in 2005, and a Best Paper Award at SCI 2001.