A low-cost application-specific neural network implementation with floating gate weights

Axel Thomsen, Martin A. Brooke
Microelectronics Research Center
Georgia Institute of Technology
791 Atlantic Drive
Atlanta, GA 30332-0269

This paper presents the technology and design philosophy for low-cost application-specific neural network implementations through MOSIS. The advantage of applying a low-cost custom-made special purpose network for a given problem over the application of available general purpose networks is discussed. The special purpose networks use floating gate devices for weight storage. The presented design approach allows the designer to implement any knowledge about the problem to be solved into hardware, thus reducing circuit complexity and training time. The example of a simple network for linear and non-linear voltage-to-current conversion illustrates the technique.

Introduction

Various neural-network designs have been presented, that offer a general purpose architecture with a certain number of fully interconnected neurons, resulting in enormous numbers of synapses and weights [1]. Often signal processing problems do not require the complexity of a general purpose network. Symmetries in pattern recognition problems for example allow a reduction of connectivity. Also most networks found in biology are not fully connected. The circuit complexity that comes with the general purpose approach has some significant drawbacks: the number of weights is much higher than required, causing long training times and potentially convergence problems in training. The connectivity is much higher than necessary, causing reduced speed compared to an application-specific network. Furthermore the chip area and power consumption are not optimized due to the number of unnecessary neurons and synapses. Knowledge about the signal processing problem cannot be implemented.

But the implementation and fabrication of application-specific neural networks poses serious problems: Implementations with fixed weights do not achieve the accuracy of trained circuits and have a very limited range of applications because they cannot adapt to variations in the environment. Floating gate devices or EEPROMs are the best adjustable weight storage devices available when compactness and accuracy are considered. Usually the fabrication of EEPROMs requires special processing to obtain thin oxide to build tunneling injectors. Specialized EEPROM fabrication processes are expensive, not generally accessible for the research community and not economical for small batch sizes. We present an EEPROM device in a process with general access and low fabrication cost even for small quantities. This is crucial to make application-specific networks economically feasible.

The application-specific neural network we suggest, offers floating gate weight storage in a user designed network that is optimized for a certain application at the low cost and fast turn-around that is offered by MOSIS [3]. The network connections and weights and the extent of trainability can be specified by the user. Often the connectivity and the weights of a neural network can be derived in a different way than a training simulation of a fully connected network with a back-propagation algorithm. For certain applications analytic and intuitive methods will yield useful networks that will rarely be fully connected.

The paper will explain the EEPROM device and its layout and characteristics, and show the design example of a simple one-dimensional network for linear and non-linear voltage-to-current conversion. The example circuit will show how custom design can improve the performance of a network vastly. It consists of 15 neurons, each with offset and weight adjustment possibilities. The offsets and weights are preset roughly to obtain a linear conversion but can be changed to either improve the linearity or approximate nonlinear functions. The presetting of weights allows a very simple training procedure.

0-7803-0559-0 /92 $3.00

Floating Gate Device In Standard Technology

A special layout in the ORBIT 2μm double polysilicon CMOS process allows the fabrication of a floating gate device with tunneling injector without additional processing steps [2]. The device layout is shown in fig. 1. Tunneling occurs at a crossover of two polysilicon lines due to field enhancement and oxide thinning when voltages of above 12V are applied. While the properties of this device are not optimized in terms of compactness, read/write endurance and programming voltage, the parameters significant for analog circuits applications, such as the achievable accuracy and the charge retention over time are very promising. A charge loss of less than 0.1% in 10 years has been reported, and achievable accuracy is limited by the programming algorithm and measurement accuracy. The yield is at 99.85%. Since the device is based on a parasitic effect, the parameters for the tunneling injector vary significantly. This requires consideration in the programming algorithm. Programming has to be done with a variable programming voltage for fast approximation with higher voltages in the beginning and lower programming voltage later to achieve high accuracy. The size of the programming voltage will differ for each device by up to 2V.
The layout technique was also successfully applied in the ORBIT 2μm low-noise analog CMOS process.

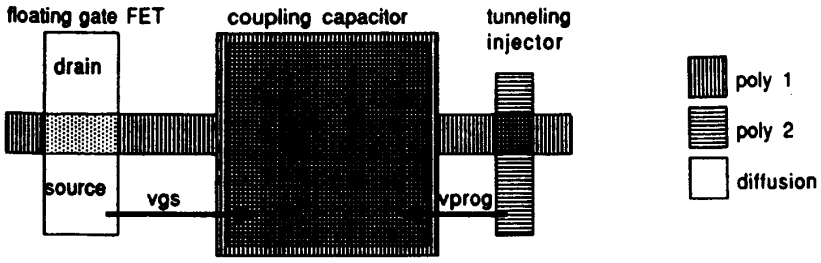

Fig. 1: Layout of the floating gate device with tunneling injector

The Voltage-To-Current Conversion Problem

Linear voltage-to-current conversion becomes a problem when the device modeling is complex or insufficient, because conventional approaches rely on simple device modeling equations. Here the problem is addressed by application of neural network design ideas. For this network the necessary weights, offsets and active ranges of each neuron could be determined analytically. Also, the number of required neurons could be determined from accuracy specifications. The knowledge about the active range of each neuron led to three slightly different neuron designs each for a different range of input values so that the circuit can exceed the input range possible with a conventional analog circuit design or a general purpose neural network. An implementation has to include trainability although the weights are known, because the technology does not provide sufficient accuracy with geometry-based weights.

Weight And Neuron Circuits

The neuron for voltage-to-current conversion is a differential transconductance, that has the input voltage and a tunable offset voltage as inputs (fig.4). Neurons for input values near the power supply rails include level shifter circuits. The transconductance output gets clipped by two unidirectional current mirrors to a region of high linearity around the center point of the curve. The current output is shown in figure 3. The tail-current I_{tail} controls the weight, which in this case is the transconductance g_m, according to

$$g_m = 2*(\beta * I_{tail})^{1/2}.$$

The weight and offset storage circuit is a programmable bi-directional current source. The circuit diagram is shown in fig. 2. The floating gate device is part of a differential pair. In this arrangement,

thermal variation of the current is minimized. The current flow is limited by the tail-current of the differential pair to avoid damage during a positive programming pulse.
The circuit implementation already includes an approximation of the weights and offsets for a linear conversion. The weight current is added to the tail-current to adjust the weight g_m. Changing the offset requires a voltage shift of the approximate offset. A transconductance in a feedback configuration can generate a voltage difference depending on the input current (fig.5). With the input current being programmable from the offset storage circuit above, offset adjustment is possible. The advantages of starting the training from an approximation will be discussed later.

The Complete Network And Its Training

This simple network consists of 15 neurons in one layer, one input and one output variable. 30 floating gate current sources are implemented to adjust 15 weights and 15 offsets. Circuitry is included to generate approximate weights and offsets for a linear conversion. These circuits are based on simple device models and geometry and do not provide sufficient accuracy to achieve low error. But an approximate generation of offset voltages and weights is important because it facilitates training. Instead of starting without any knowledge, the network is already close to its optimum state at the start of the training procedure. In the case of this network, it is known, which weight must be changed to reduce the error in a certain area. It requires only one or two data-points to determine the incorrect weight and the direction of change. Thus the number of iterations to find the solution reduces dramatically. This is an important issue when dealing with the poorly modeled floating gate devices that are used here. Any training algorithm has to be robust enough to converge in spite of the unpredictability of the size of a weight change. Training will become much more complex if the target of a weight change has to be determined, too. Furthermore the approximation of weights and offsets will improve accuracy, since the effect of charge loss from a floating gate is reduced. Only the amount of correction will change, not an absolute value.
Programming is done in the following manner: first all offset voltages are programmed to approximately the right position. Since the offset voltages are arranged in a chain, the order of programming is fixed. Then the weights of the individual modules are adjusted. Finally the offset voltages are finely tuned to their correct value. The accuracy of the offset voltages is most crucial to the overall accuracy of the transfer curve. The programming algorithm for the individual device has two steps: First, the programming voltage is ramped up and down to find suitable voltages for significant weight changes in both directions. Second, the programming voltages are reduced slowly to achieve high accuracy with small weight changes. A typical programming sequence is shown in fig. 6.

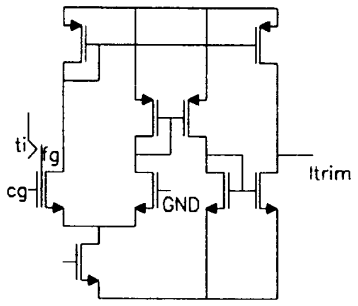

Fig. 2: Circuit diagram of the programmable current source with floating gate device

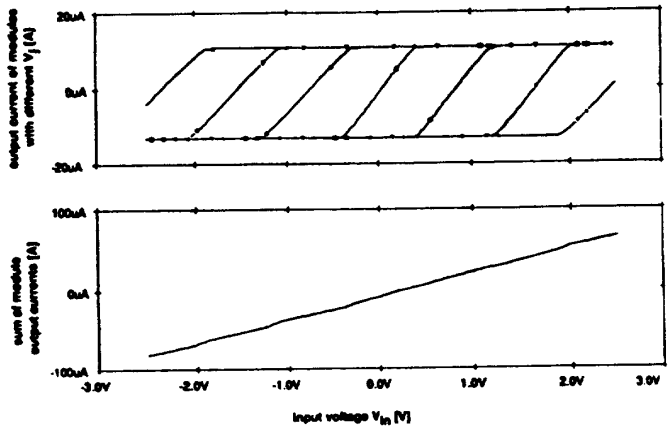

Fig. 3: Output current of several neurons: separately (top) and added (bottom)

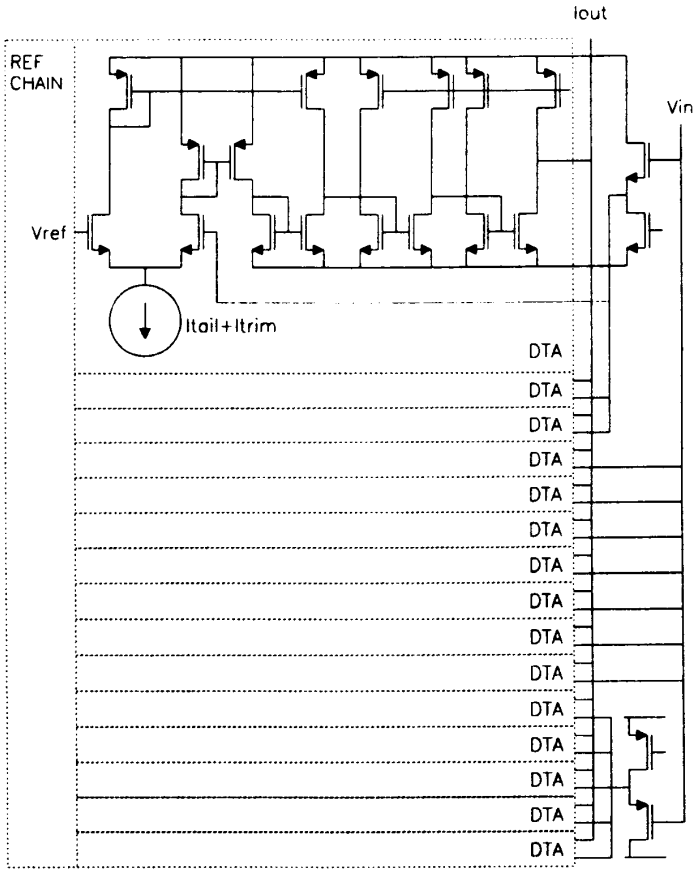

Fig. 4: Circuit diagram of the complete system: 15 neurons, level shifter circuits and the reference generation circuit

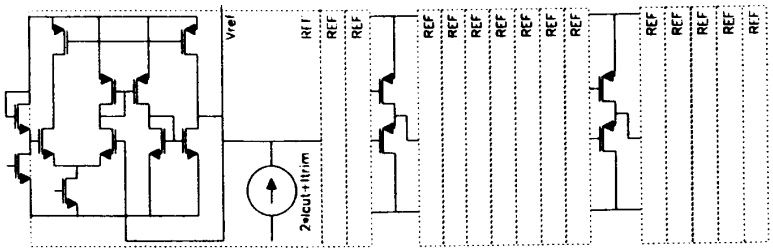

Fig. 5: The reference-generation circuit

Experimental Results

The circuit was implemented in a 2µm double polysilicon p-well process. All devices have minimum drawn channel length of 2µm. This implementation consisted of fifteen neurons. With a tail-current of 30µA and a cutoff current of 5µA an input range of the full power supply voltage swing from 0 to 5V was achieved. The circuit was programmed to a transfer curve of

$$I_{out} = 110\ \mu A - 36\ \mu A / V * V_{in}$$

with tolerances of 1µA/V in transconductance and 200nA for reference outputs. The transfer curve before and after trimming is shown in figure 7. The deviation from the desired curve is shown in fig. 8. The accuracy observed was 0.5µA or 0.3% of the output swing. The output resistance is in the order of 100kΩ. The output dynamic range is 0.5V to 4.5V. The measured 3dB frequency for operation into a 50 Ohm load is about 15MHz.
The same circuit was then programmed to a nonlinear function given by

$$I_{out} = 20\ \mu A - 70\ \mu A * \arctan ((V_{in} - 2.5\ V) / 1.6\ V).$$

The parameters were chosen to fit the input and output ranges of the given bias point. The error ΔV was calculated from application of the inverse function to the output.

$$\Delta V = V_{in} - 2.5V + 1.6\ V * \tan ((I_{out} - 20\ \mu A) / 70\ \mu A)$$

A deviation of 50 mV or 1% of the input swing was measured (fig. 9). In addition to the programming tolerances an error is inherent in the piecewise linear approximation approach. It strongly depends on the shape of the function and is least for low variations in the first derivative. It can be reduced by increasing the number of modules.

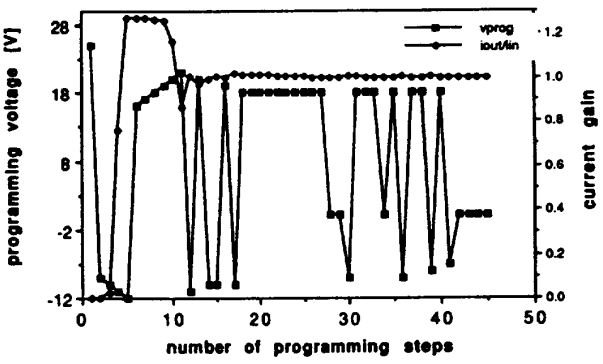

Fig. 6: Output value and programming voltage for a typical programming sequence of a single device

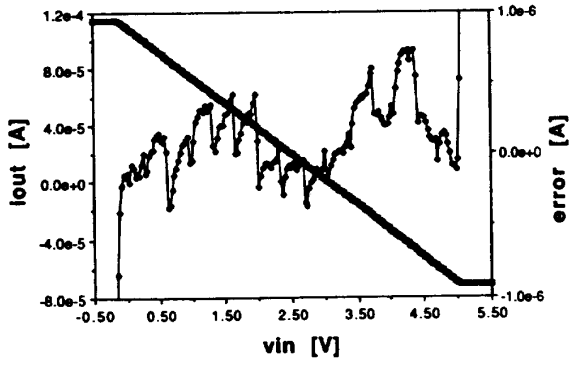

Fig. 7: Output current and error for a linear transfer-function

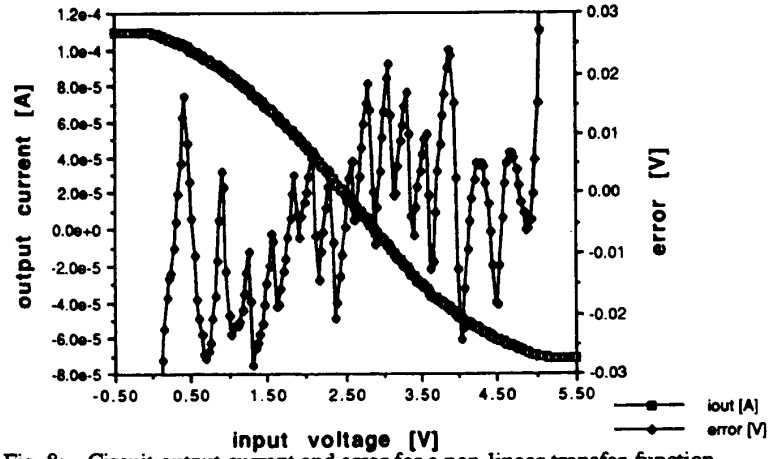

Fig. 8: Circuit output current and error for a non-linear transfer-function

Discussion

To solve this simple signal processing problem to the accuracy and speed presented, there is currently no alternative to custom design. Custom design is the only method to achieve the input range. Any general purpose network will be too complex to match the input-range and speed specifications achieved with this design. The example shown is a fairly simple, so a generalization to more complex problems seems appropriate. To the user, that is interested in the best performance of a network for a given problem, a general purpose approach can never be optimal. The design approach presented here can implement all knowledge the designer has about the problem such as zero weights, results from a training simulation, analytically derived weights etc.. The designer can build in the trainability that he thinks is required to handle the uncertainty inherent in any implementation. The trade-in is a reduced flexibility of the network, reduced fault tolerance, and reduced correction possibilities. Furthermore does the floating-gate device and its parameter-spread introduce a new difficulty to the training process. The device itself requires more area than a special process device. A decision on which approach is advantageous will have to be made based on how much the required network differs from a fully connected general purpose implementation. Given the suitable requirements the application specific approach will be a significant performance improvement. The technology presented here will make this approach economical.

Conclusion

A low-cost application-specific neural network design technique and philosophy is presented. It offers floating gate weight storage and user specified connectivity to avoid the disadvantages of general purpose neural network implementations. Its main advantages are the reduced complexity, reduced training time, and improved speed. Disadvantages are reduced fault tolerance and the use of a poorly modeled floating gate device, that requires a robust training algorithm. The design example of a one-dimensional network for voltage-to-current conversion with 30 weights achieved an error of less than 1% and a speed of 15 MHz.

References

[1] M. Holler etal., "An electrically trainable artificial neural network (ETANN) with 10240 floating gate synapses," *Proc. of IJCNN*, v. II, pp. 191-196, Washington DC, June 1989

[2] A.Thomsen, M.A.Brooke, "A floating gate MOSFET with tunneling injector fabricated using a standard double polysilicon CMOS process," *IEEE Electron Device Letters*, v. EDL-12, no. 3, pp. 111-113, March 1991

[3] The MOSIS Project, USC/ISI, 4676 Admiralty Way, Marina del Rey, CA 90292-6695