A NEURAL-TYPE POOL ARITHMETIC UNIT *

S. W. Tsay and R. W. Newcomb

Microsystems Laboratory
Electrical Engineering Department
University of Maryland
College Park, MD 20742
Tel: (301) 405-3708

Abstract

A neural-type pool arithmetic unit is presented. The implementation of this arithmetic unit is based on the describing equations of neural chemical pools that occur in biological neurons where the concentration of a chemical pool depends on the synthesis (inward current) and degradation (outward current) of the chemical materials. With different arrangements of inward and outward currents, this neural type arithmetic unit can perform functions such as addition, subtraction, sign inversion, square, and square root on voltages.

1 Introduction

A neural pool represents the chemical storage in a neural cell and can be used to simulate the primary and second messenger chemical-electrical interactions in a neural system [1][2]. Because of the synthesis and degradation of the chemical materials, a neural pool can be described by the following equation:

$$C\frac{dP}{dt} = I_{in} - I_{out} \qquad (1)$$

where C is the pool volume, P is the concentration of the material and I_{in} and I_{out} represent the rates of the materials flowing into and out of the pool, respectively. I_{in} and I_{out} can be modulated by some other pools' levels or by the same pool [1][2]. The equilibrium state of the pool concentration P is determined when the inward current I_{in} is equal to the outward current I_{out}.

Equation (1) can be realized as the simple circuit shown in Figure 1 where the voltage across the capacitor represents the pool concentration and the capacitance represents the pool volume.

2 Circuit Realization of Neural-Type Pool Arithmetic Unit

As mentioned in the introduction, I_{in} and I_{out} can be modulated by other signals. If the simplest modulation, linear modulation, is emploied, a pool can be realized as the circuit in Fig. 2 in which I_{in} is linearly proportional to the difference $V_a - V_b$ and I_{out} is linearly proportional to $V_c - V_d$. As next discussed, the circuit of Fig. 2 realizes various arithmetic functions and, consequently, is called a "neural-type pool arithmetic unit".

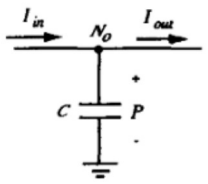

Figure 1: Basic circuit realization of pools.

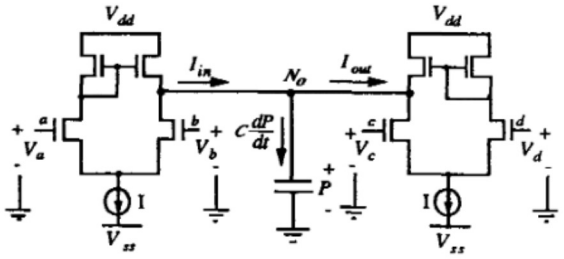

Figure 2: A pool with linear modulation on I_{in} and I_{out}.

2.1 Adder, Subtracter, and Sign Inverter

Three arithmetic functions, addition, subtraction, and sign inversion, can be easily obtained with the circuit of Fig. 2 where different arrangements of the nodes in Fig. 2 can result in different functions.

When we tie node b to N_o and c to GND, the inward and outward currents will be

$$I_{in} = K(V_a - P) \quad \text{and} \quad I_{out} = K'(0 - V_d) \qquad (2)$$

where K and K' are constants that depend on the sizes of the transistors and on the current I_{ss} in the differential pairs in Fig. 2. If these two differential pairs are identical, K and K' will be equal and can be cancelled out. The equilibrium state of this pool is determined when $(V_a - P) = (0 - V_d)$. In

* Research Supported by ONR Grant No. N00014-90-J-1114.

CH 3006-4/91/0000 – 2518 $1.00 © IEEE

this case we get

$$P = V_a + V_d \tag{3}$$

For the function of subtraction, if node b is tied to N_o and d is tied to GND, the inward and outward currents are then be described by

$$I_{in} = K(V_a - P) \quad \text{and} \quad I_{out} = K'(V_c - 0) \tag{4}$$

Again, when $K = K'$ the equilibrium state of this pool will be

$$P = V_a - V_c \tag{5}$$

Sign inversion can be achieved by tying node a in the subtracter to GND so that the output, according to (5), is

$$P = 0 - V_c = -V_c \tag{6}$$

These functions of addition, subtraction, and sign inversion are summarized in the Table 1.

Function	V_a	V_b	V_c	V_d	Result
Adder	V_a	P	GND	V_d	$P = V_a + V_d$
Subtracter	V_a	P	V_c	GND	$P = V_a - V_c$
Sign Inverter	GND	P	V_c	GND	$P = -V_c$

Table 1: Input assignments for the neural-type pool arithmetic unit (Figure 2) to perform addition, subtraction, and sign inversion.

2.2 Square and Square Root circuits

By modifying one of the two differential pairs, the square and square root circuits can be achieved. These functions result when one of the currents I_{in} or I_{out} is generated by enhancement mode n-channel MOS transistor M1 which is in its saturation region. Figure 3 shows a circuit to generate the square function. The gate of M1 is tied to the output of a 3-input voltage adder as shown in Fig. 3 where P is the source voltage of M1 and V_t comes from a voltage source and is used to cancel out the threshold voltage of M1. M1 is in its saturation region when $V_{dd} \geq V_{in} + P$. The inward current, flowing from V_{dd} through transistor M1, is

$$I_{in} = C(V_{gs} - V_t)^2 = C(V_{in})^2 \tag{7}$$

where $C = \frac{K_p W}{2L}$ is the MOS constant with W/L the width to length ratio.

The outward current, marked as I_{out} in Fig. 3, is

$$I_{out} = C'(P - 0) \tag{8}$$

where C' is the gain of the differential amplifier at the right hand side. The equilibrium state of P in this circuit is at $I_{in} = I_{out}$. That is,

$$C(V_{in})^2 = C'(P) \quad \Rightarrow \quad P = \frac{C}{C'}(V_{in})^2 \tag{9}$$

We can change the W/L ratio of M1 and/or the current source I_{ss} to make $C = C'$, and in this case, P is the square of V_{in}.

The square root function can be achieved by the circuit in Fig. 4 which is something like a turned around Fig. 3. In Fig. 4 we assume $V_{in} \geq 0$ and denote by P the drain voltage of the n-channel enhancement mode transistor M1 whose source is tied to ground. M1 is in saturation because $V_{ds(M1)} = V_{gs(M1)} - V_t = P$. The outward current and inward current are

$$I_{out} = C(P)^2 \quad \text{and} \quad I_{in} = C'V_{in} \tag{10}$$

Because $I_{in} = I_{out}$, we get

$$C'V_{in} = C(P)^2 \quad \Rightarrow \quad P = K\sqrt{V_{in}} \tag{11}$$

where $K = \sqrt{\frac{C'}{C}}$ is a constant which can be set to be 1 by adjusting I_{ss} and/or changing the W/L ratio of M1.

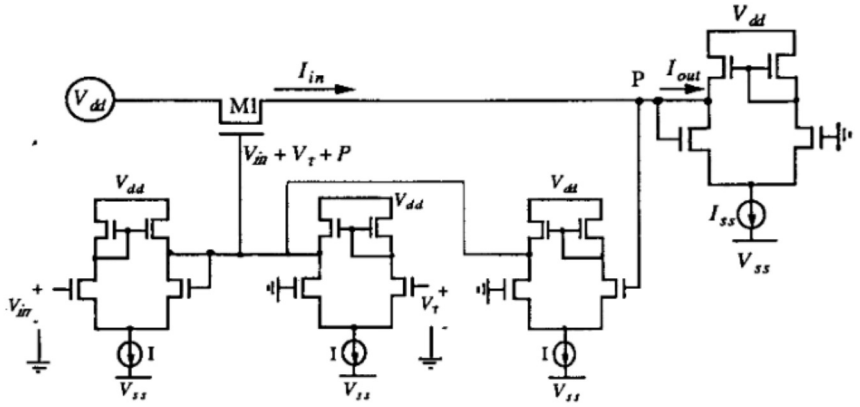

Figure 3: Simplified circuit for "Square" function.

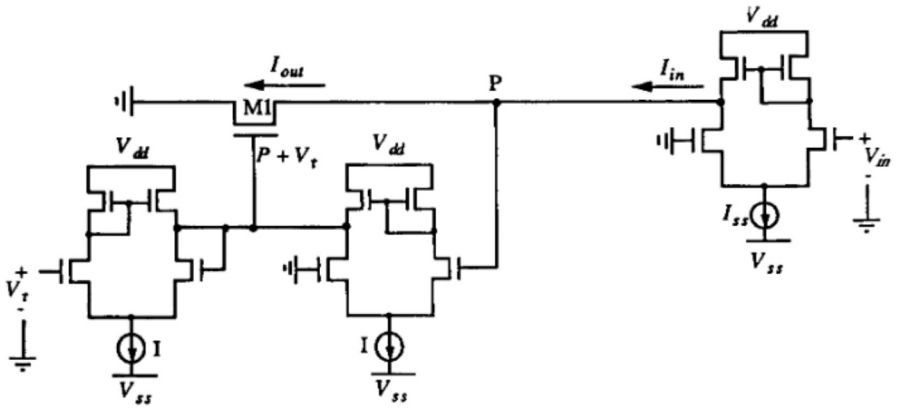

Figure 4: Simplified circuit for "Square root" function.

In equation (11) we rule out the possibility of $P=-K\sqrt{V_{in}}$ because when $V_{in}\geq 0$, the current is always flowing from right to left as indicated in Fig. 4. Thus, the drain voltage of M1, P, is always greater than the source voltage, 0 volt.

2.3 Simulation Results

The five arithmetic functions mentioned above have been simulated by PSPICE and the results are given in Figs. 5 - 9. To improve the accuracy of the results and increase the range of input voltages, we used the improved differential amplifier developed in [3] to replace all the differential amplifiers in Figs. 2, 3, and 4. All the current mirrors in the differential amplifiers were also replaced by a regulated current mirror because of the need for high output resistance [4].

3 Conclusions

A neural-type pool arithmetic unit has been introduced. This arithmetic unit is based on the structure of chemical pools in the biological neuron. Five functions, addition, subtraction, sign inversion, square, and square root have been implemented. This arithmetic unit works on analog voltages with no resistors needed. With this arithmetic unit, circuit which is equivalent to an adjustable threshold MOSFET can be made [5]. More applications are still under investigation.

References

[1] D. K. Hartline, "Simulation of Restricted Neural Networks with Reprogrammable Neurons," IEEE Transactions on Circuits and Systems, Vol. 36, No. 5, May, 1989, pp. 653-660.

[2] D. K. Hartline, "Synetsim 3.3 User's Manual," Bekesy Lab., Honolunu, HI, 1990.

[3] S. W. Tsay, N. El-Leithy, and R. Newcomb, "CMOS Realization of a Class of Hartline Neural Pools," Proceedings of the IEEE International Symposium on Circuits and Systems, New Orleans, May, 1990, Vol. 3, pp. 2417-2420.

[4] R. Geiger, P. Allen, and N. Strader, "VLSI Design Techniques for Analog and Digital Circuits," McGraw-Hill, New York, 1990.

[5] S. W. Tsay and R. W. Newcomb, "A Neural Arithmetic Unit and Adjustable Threshold MOSFET," manuscript prepared.

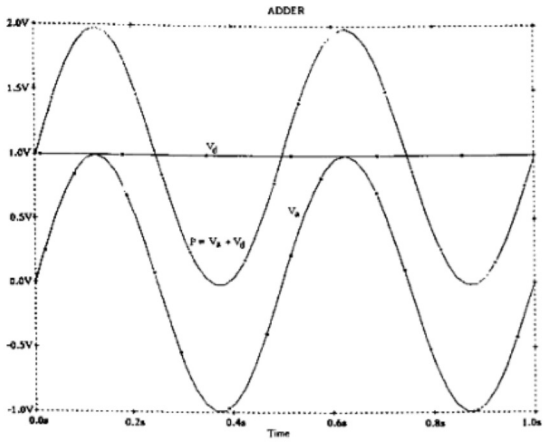

Figure 5: Simulation results for addition.

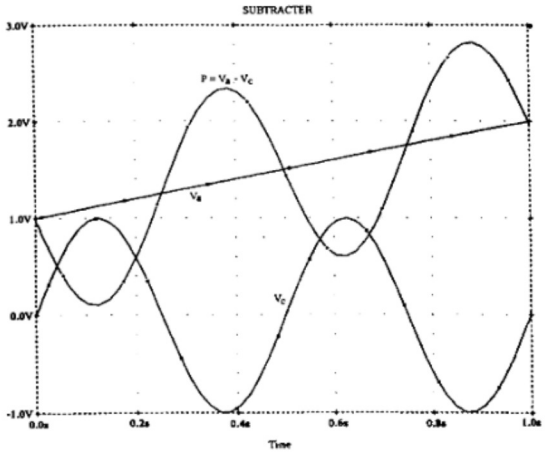

Figure 6: Simulation results for subtraction.

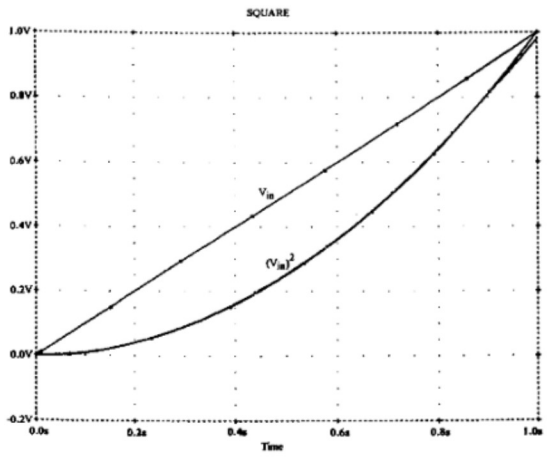

Figure 8: Simulation results for square.

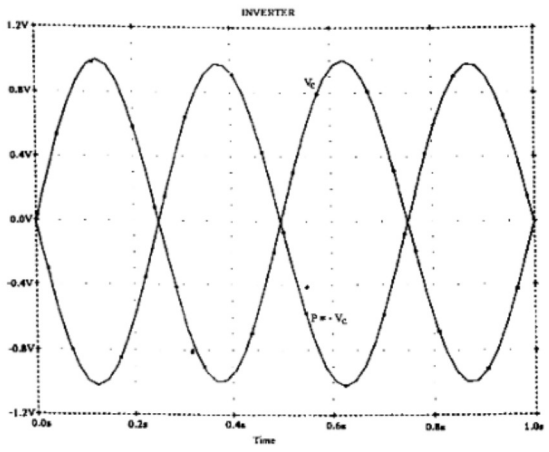

Figure 7: Simulation results for sign inversion.

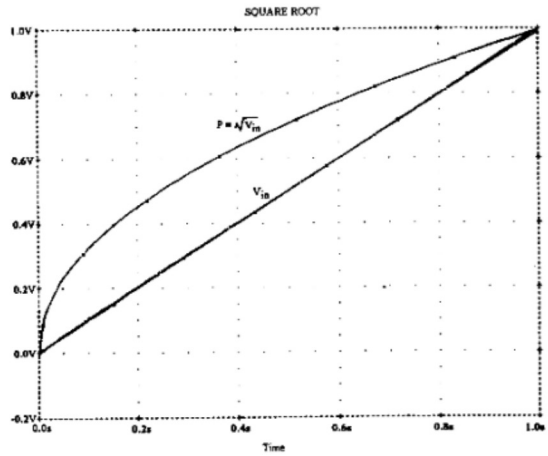

Figure 9: Simulation results for square root.