[6] P. Vorenkamp and R. Roovers, "A 12-b, 60 Msample/s cascaded folding and interpolating ADC," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1876–1886, Dec. 1997.
[7] U. Fiedler and D. Seitzer, "A high-speed 8 bit A/D converter based on a Gray-code multiple folding circuit," *IEEE Trans. Solid-State Circuits*, vol. SC-14, pp. 547–551, June 1979.
[8] P. E. Pace, P. A. Ramamoorthy, and D. Styer, "A preprocessing architecture for resolution enhancement in high-speed analog-to-digital converters," *IEEE Trans. Circuits Syst.*, vol. 41, pp. 373–379, June 1994.
[9] P. E. Pace, R. E. Leino, and D. Styer, "Use of the symmetrical number system in resolving single-frequency undersampling ambiguities," *IEEE Trans. Signal Processing*, vol. 45, pp. 1153–1160, May 1997.

A CMOS General-Purpose Sampled-Data Analog Processing Element

Piotr Dudek and Peter J. Hicks

***Abstract*—This brief presents the architecture and implementation of an analog processor, which in a way akin to a digital microprocessor, embodies a physical model of the universal Turing machine. The analog microprocessor (AμP) executes software programs, while nevertheless operating on analog sampled data values. This enables the design of mixed-mode systems which retain the speed/area/power advantages of the analog signal processing paradigm, while being fully programmable general-purpose systems. A proof-of-concept integrated circuit has been implemented in 0.8-μm CMOS technology, using switched-current techniques. Experimental results from fabricated chips are presented and examples of the application of the AμP's in image processing are given.**

***Index Terms*—Programmable analog circuits, switched-current circuits, VLSI processor arrays.**

I. Introduction

Analog signal processing circuits can offer advantages over their digital counterparts in terms of speed, power dissipation, and area consumed by the circuitry [1]. The desire to make these advantages more accessible to the system designer has resulted in interesting developments in the area of general-purpose analog circuits; for example, analog processors [2], [3] and field-programmable analog arrays (FPAA's) [4]. These devices operate either in continuous time or in sampled-data mode [5]–[7], but their programmability is based on a similar principle. An array of basic configurable cells is provided, and the functionality of the system is determined by establishing a signal path within the device. In other words, the processing algorithm is implemented in *hardware*.

However, as compared with reconfigurable circuits, digital signal processors, or microprocessors, offer far greater flexibility. They can be relatively easily programmed to perform diverse and complex tasks. This stems from the universal Turing machine paradigm [8]. The functionality of the system is determined by *software*. A stored-program computer comprises a microprocessor consisting of a register file, an execution unit and a control unit, and the external memory for holding data and instructions. The processor performs a sequence of iterative, algorithmic computations, which manipulate digital data according to the instruction-level program.

Manuscript received August 1999; revised January 2000. This work was supported by The Batory Foundation and UMIST. This paper was recommended by Associate Editor M. Helfenstein.

The authors are with the Department of Electrical and Electronic Engineering, University of Manchester Institute of Science and Technology (UMIST), Manchester M60 1QD U.K.

Publisher Item Identifier S 1057-7130(00)04202-6.

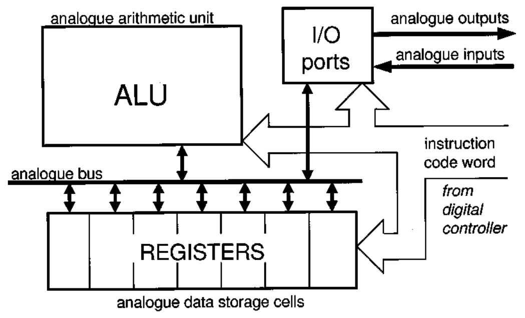

Fig. 1. The architecture of the AμP.

Yet, the instruction-level programmability can also be exploited in a system with an analog data path. Such an analog processor has been described in [9]. The architecture presented there was based on charge-domain operations and the circuitry involved high-gain amplifiers, capacitors and analog switches. Alas, a discrete-components level of implementation exhibited rather poor performance, and the idea seems to have been abandoned. More recently, the invention of the Cellular Neural Network Universal Machine (CNN-UM) [10] has attracted a great deal of interest [11]. The CNN-UM is an instruction-level programmable analog computer. However, since the CNN-UM concept is based on a specific connectivity model (cellular array) and algorithms (CNN transients with programmable templates) the architecture and the instruction set of the CNN-UM differ from that of a digital microprocessor. This can be beneficial in some cases, but sometimes a more conventional programming model would be desirable.

In this paper we describe the analog microprocessor (AμP), which has an architecture similar to a simple general-purpose digital microprocessor, and as such provides a platform for the execution of conventional "digital" algorithms. However, since the processing is performed in the analog domain, the AμP can achieve savings in terms of silicon area and power dissipation, as compared with digital processors.

The remainder of this paper is organized as follows. Section II introduces the architecture of the AμP and Section III describes the switched-current (SI) implementation of the AμP. Section IV discusses the issues involving the accuracy of processing, Section V presents experimental results obtained from the test chip, and Section VI concludes the paper.

II. Analog Microprocessor Architecture

The block diagram of the generic AμP architecture is presented in Fig. 1. The AμP consists of a register file (each register is an analog memory cell, capable of storing a sample of data), an analog arithmetic logic unit (ALU), and an analog I/O port. All the building blocks are interconnected via an analog data bus. The processing of information is performed entirely on analog values; however, the AμP operates in discrete time and executes a software program, performing consecutive instructions issued by a digital controller. These instructions may include register transfer operations, which move the analog samples of data between registers of the AμP, I/O operations, which move the data to and from I/O ports, arithmetic operations (addition, subtraction, multiplication, division), which modify the analog data, and comparison operations, which allow for conditional branching. The program is stored in the local memory of the controller, which is considered here as an external, purely digital device. The complete processor is

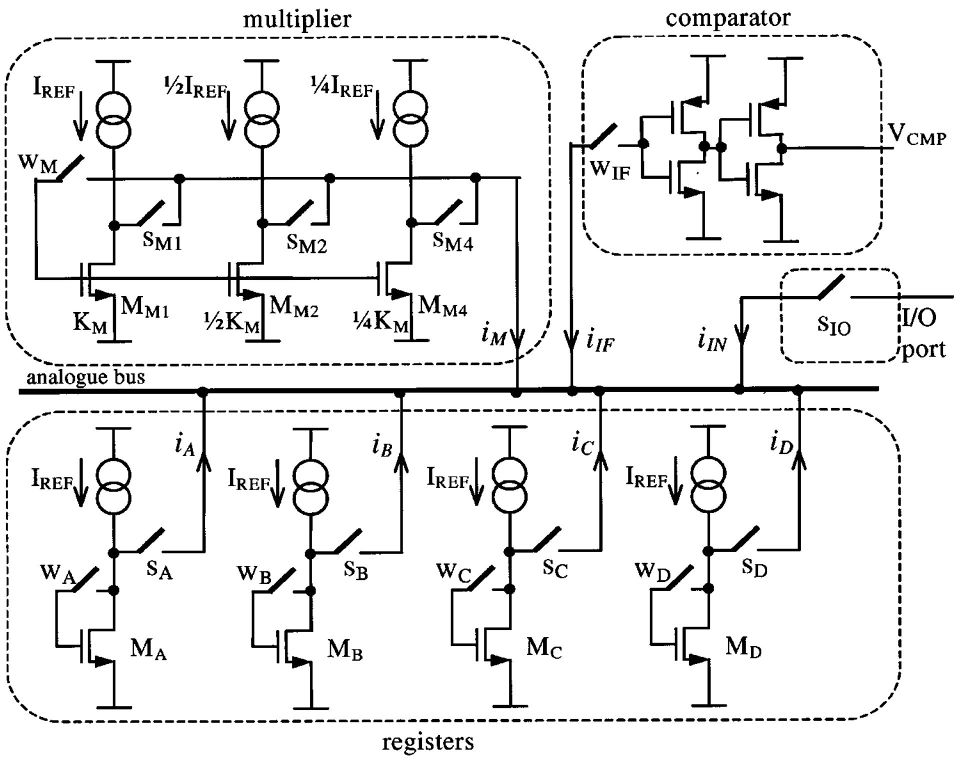

Fig. 2. Conceptual diagram of a simple switched-current AμP.

therefore a mixed-mode system, with an analog data path and a digital control path, which embodies a physical model of the universal Turing machine.

III. SI Implementation

We implement the AμP utilizing SI techniques [12]. SI circuits have become a feasible alternative to switched-capacitor (SC) circuits for the implementations of analog sampled-data systems. They can be realized in the standard digital CMOS process technology, and therefore are particularly suited for the design of mixed-mode systems. Simple structures of SI cells result in area- and power-efficient AμP implementations while offering adequate speed and accuracy to satisfy a wide range of applications. Moreover, using current-mode arithmetic circuits greatly simplifies the design of the ALU.

A. The Register File

The schematic diagram of a simplified AμP is presented in Fig. 2. The depicted register file comprises four registers (A–D), each of which is a basic SI memory cell [12] consisting of a memory transistor M_X $(X = A, B, C, D)$, a current source, and two switches W_X and S_X. Consider a simple register-transfer operation, which can be denoted as $\mathbf{A} \leftarrow \mathbf{B}$. To execute this instruction the switches W_A, S_A and S_B are closed, the remaining ones are open. Therefore, the only nonzero currents entering the analog bus node will be the current i_B, which is the current read out from the register B, and the current i_A, which is being written to the register A. Of course, it is also true that

$$i_A = -i_B. \tag{1}$$

The current i_A is stored in the register A following a basic SI memory cell operation. Since W_A is closed, the transistor M_A is diode-connected and therefore its gate-source voltage $V_{\mathrm{gs}\,A}$ will set itself to the value corresponding to the drain current, as described by the saturation-region equation

$$I_{\mathrm{REF}} - i_A = I_{\mathrm{ds}\,A} = K_A(V_{\mathrm{gs}\,A} - V_t)^2. \tag{2}$$

If the switch W_A is now opened, a quantity of charge will be stored at the gate capacitance of the transistor M_A, and the gate-source voltage $V_{\mathrm{gs}\,A}$ will hold its value (disregarding any error effects). By default, all switches revert to the open position after an instruction has been executed. For correct operation, it is also necessary to ensure that W-switches always open before S-switches.

For subsequent reading from the register, the switch S_A will be closed (W_A will remain open) and the drain current of M_A will be set by the gate-source voltage $V_{\mathrm{gs}\,A}$ (it is assumed that the memory transistors are in saturation when their corresponding S-switches are closed), so the stored current i_A can be read out.

B. Analog ALU

The analog ALU is required to provide the basic arithmetic operations of addition, subtraction, and multiplication/division. However, subtraction can be achieved by an inversion followed by an addition, and as can be seen from (1), the inversion is inherent in the basic current-transfer operation. Moreover, the addition operation in a current-mode system is performed on the analog bus with no area overhead, using current summation. For example, to execute instruction $\mathbf{D} \leftarrow \mathbf{B} + \mathbf{C}$, the switches W_D, S_D, S_B and S_C are closed, the remaining ones are open, and the current stored in register D is equal to

$$i_D = -(i_B + i_C). \tag{3}$$

Therefore, only a multiplier/divider needs to be physically implemented. Four-quadrant multipliers have been described in the literature [13], however we implement only a digitally controlled multiplier/divider, as illustrated in Fig. 2. The multiplier is constructed

TABLE I
AN EXAMPLE PROGRAM. HIGH-LEVEL DESCRIPTION IS COMPILED TO OBTAIN A MACHINE-LEVEL CODE. THE BIT VALUES "0" IN THE INSTRUCTION CODE WORD CORRESPOND TO THE APPROPRIATE SWITCH OPENED, THE VALUES "1" DENOTE CLOSED SWITCHES

high-level program	instruction mnemonic	Instruction code word (ICW)														current transfers
		S_{IO}	W_A	S_A	W_B	S_B	W_C	S_C	W_D	S_D	W_M	S_{M1}	S_{M2}	S_{M4}	W_{IF}	
main () {	A←IN	1	1	1	0	0	0	0	0	0	0	0	0	0	0	$i_A = -i_{IN}$
VarA = inport(IN);	M←A	0	0	1	0	0	0	0	0	0	1	1	0	0	0	$i_M = -i_A$
VarB = 0.75 * VarA;	B←M(½+¼)	0	0	0	1	1	0	0	0	0	0	0	1	1	0	$i_B = -(½+¼)i_M = 0.75\, i_A$
VarC = VarA – VarB;	D←A	0	0	1	0	0	0	0	1	1	0	0	0	0	0	$i_D = -i_A$
}	C←B+D	0	0	0	0	1	1	1	0	1	0	0	0	0	0	$i_C = -(i_B + i_D) = i_A - i_B$

as a set of current-mirrors, with transistors M_{M1}, M_{M2}, and M_{M4} scaled in a geometric series with base 2^{-1}. This is enough to realize the multiplication of an analog value by a digital constant. The current i_M is stored in the multiplier, just like in another register, by closing switches W_M and S_{M1}. We get

$$I_{\mathrm{ds}M1} = I_{\mathrm{REF}} - i_M = K_M(V_{\mathrm{gs}\,M} - V_t)^2. \tag{4}$$

The current read out from the multiplier i'_M depends on the multiplication factor k. This is a binary word which selects the appropriate mirrors using switches S_{M1}, S_{M2}, and S_{M4}

$$i'_M = k \cdot I_{\mathrm{REF}} - k \cdot K_M(V_{\mathrm{gs}\,M} - V_t)^2 = k \cdot i_M. \tag{5}$$

The comparison operation is, in principle, also performed with no area overhead. If two or more register outputs are connected to the analog bus, which is kept in a high impedance state, then the bus node will be charged to a "high" voltage level or discharged to a "low" voltage level, depending on the sign of the total current. To sense the logic value, a simple CMOS buffer can be used. The output voltage of the buffer V_{CMP} provides the controller with the comparison result, allowing for a conditional execution of subsequent instructions.

The ALU is also capable of Boolean logic operations, if needed. By representing logic values by current levels, and combining arithmetic and comparison operations, logic operations of AND, OR, and NOT can be realized.

An input/output (I/O) port is simply realized by an analog switch S_{IO}, which connects the port node to the analog bus.

All current sources and analog switches can be implemented using CMOS transistors. In practice, basic SI circuits can be replaced by a more elaborate design, however the principle of the operation will remain the same.

C. Program Execution

All of the switches within the AμP are operated in response to logic-level voltages set by a digital controller. The complete set of these voltages, controlling all switches, forms the instruction-code word (ICW). The sequence of the ICW's issued by the controller constitutes a machine-level software program.

To illustrate the algorithmic operation of the AμP, let's consider the following example. As a part of a certain signal processing algorithm, the input value has to be stored in variable $\mathrm{Var}A$, then multiplied by 0.75 and stored in variable $\mathrm{Var}B$, and then the two variables should be subtracted, and the result stored in the third variable $\mathrm{Var}C$. The high-level description of the program, in the C-language notation, together with the resulting machine-level code, ICW's, and current transfer equations are presented in Table I. (Program compilation was done in this case manually; however, a high-level programming framework and a suitable compiler are under development). The machine-level program contains five instructions. The variables $\mathrm{Var}A, \mathrm{Var}B$ and $\mathrm{Var}C$ are assigned to the registers A, B and C of the processor. We will now follow the execution of this program on the analog microprocessor.

On the first instruction $\mathbf{A} \leftarrow \mathbf{IN}$, the switches S_A, W_A, and S_{IO} are closed, and the remaining ones are open. Therefore, the current $i_A = -i_{\mathrm{IN}}$ is stored in the register A. The second instruction in our example $\mathbf{M} \leftarrow \mathbf{A}$ is the first cycle of a multiplication operation. The switches S_A, S_{M1}, and W_M are closed, the remaining ones are open, and current $i_M = -i_A$ is stored in the multiplier. The third instruction $\mathbf{B} \leftarrow \mathbf{M(1/2 + 1/4)}$ completes the multiplication. Now switches W_B, S_B, S_{M2} and S_{M4} are closed. By selecting the multiplier mirrors with the weights of 1/2 and 1/4 the required multiplication factor is achieved and the result of the multiplication is stored in the register B; thus, $i_B = 0.75 \cdot i_A$. Finally, the subtraction. We need to use the register D for temporary storage. On the fourth instruction $\mathbf{D} \leftarrow \mathbf{A}$, the switches W_D, S_D, and S_A are closed, and the current is transferred from i_A to i_D, with sign inversion. On the fifth instruction $\mathbf{C} \leftarrow \mathbf{B} + \mathbf{D}$, switches W_C, S_C, S_B, and S_D are closed. Now currents i_B and i_D are summed on the analog bus, and the result is remembered in the register C, thus $i_C = i_A - i_B$. This completes the execution of the program. The sequence of ICW's dictates the way the samples of data are transferred and manipulated within the processor allowing the software implementation of the required processing algorithm.

IV. ACCURACY

An important issue with analog circuits is the accuracy of processing. Apart from noise, the major error sources in SI circuits are charge injection effects in analog switches [14], voltage coupling through the gate-source capacitance of transistors, and the finite output conductance of the transistor. The errors in SI memory cells will cause the AμP instructions to be performed with a limited accuracy. Consider the transfer instruction $\mathbf{A} \leftarrow \mathbf{B}$. In the nonideal case, the current transfer is performed with an error, which consists of a systematic part $\Delta_S(i_B)$, and a random noise $\Delta_N(*)$. The systematic error can be split into the signal-independent part Δ_{SI} and the signal-dependent part $\Delta_{\mathrm{SD}}(i_B)$

$$i_A = -i_B + \Delta_S(i_B) + \Delta_N(*) \tag{6}$$

$$\Delta_S(i_B) = \Delta_{\mathrm{SI}} + \Delta_{\mathrm{SD}}(i_B). \tag{7}$$

A. Signal-Dependent Error Cancellation

Many methods have been proposed to reduce the error effects in SI circuits [12], and circuits with 11-bit equivalent accuracy have been reported [15]. However, the more sophisticated methods of error compensation in SI cells require more complex circuitry, and therefore, the design of an AμP will involve tradeoffs between accuracy, speed, area and power. These tradeoffs need to be resolved taking into account application needs. A particularly good compromise between cell area, power

TABLE II
MACHINE-LEVEL INSTRUCTION SEQUENCES FOR VARIOUS ARITHMETIC OPERATIONS WITH SIGNAL-INDEPENDENT ERROR CANCELLATION

operation	instructions	current transfers showing signal-independent error
Addition	D ← A+B	$i_D = -(i_A+i_B) + \Delta_{SI}$
C := A + B	C ← D	$i_C = -i_D + \Delta_{SI} = i_A + i_B$
Subtraction	D ← A	$i_D = -i_A + \Delta_{SI}$
C := A − B	C ← B+D	$i_C = -(i_B+i_D)+\Delta_{SI} = i_A - i_B$
Multiplication	M ← A	$i_M = -i_A + \Delta_{SI}$
A := k × A	A ← M(k)	$i_A' = -k i_M + \Delta_{SI} = k i_A + (1-k)\Delta_{SI}$
Multiplication	B ← A	$i_B = -i_A + \Delta_{SI}$
(complete Δ_{SI}	M ← B	$i_M = -i_B + \Delta_{SI} = i_A$
cancellation)	B ← M(k)	$i_B' = -k\, i_M + \Delta_{SI} = -k\, i_A + \Delta_{SI}$
A := k × A	A ← B	$i_A' = -i_B' + \Delta_{SI} = k\, i_A$

dissipation and accuracy can be achieved using the S^{2}I technique [16]. This technique uses a form of double sampling to compensate for the clock feedthrough errors, and at the same time provides an almost constant bus voltage at the end of the write phase, thereby reducing output conductance errors. The circuitry achieves very good overall reduction of the signal-dependent part of the error at a minimum cost, in terms of silicon area and power consumption.

B. Signal-Independent Error Cancellation

A signal-independent error cancellation can be easily achieved by appropriate sequencing of the instructions. For example, consider the simple variable assignment operation: $\mathrm{Var}\, A = \mathrm{Var}\, B$. The basic transfer instruction of the AμP performs an inversion, so the assignment will be performed in two steps, using an auxiliary register C. First transfer $\mathbf{C} \leftarrow \mathbf{A}$, followed by $\mathbf{B} \leftarrow \mathbf{C}$. Now, assume that each transfer instruction is performed with an identical signal-independent error Δ_{SI} (assuming an ideal signal-dependent error cancellation and neglecting noise and errors arising from device mismatches). For the first transfer, we get

$$i_C = -i_A + \Delta_{\mathrm{SI}} \tag{8}$$

and as a result of the second transfer, we get

$$i_B = -i_C + \Delta_{\mathrm{SI}} = i_A - \Delta_{\mathrm{SI}} + \Delta_{\mathrm{SI}} = i_A. \tag{9}$$

The errors cancel out and an assignment operation that is free of signal-independent errors is achieved. Similarly, as shown in Table II, addition and subtraction performed in two instructions achieve complete cancellation of the signal-independent error, whereas the multiplication performed in two cycles achieves only partial reduction of the signal-independent error, and for complete signal-independent error cancellation, it requires four instructions. However, on many occasions, the accuracy of two-instruction multiplication might be sufficient. Moreover, some optimization of the code is possible, to reduce the number of instructions for a given number of operations. Therefore, it can be assumed, that a typical assignment/arithmetic operation takes two instruction cycles on the AμP.

V. EXPERIMENTAL RESULTS

A. Test Chip

Various architectural choices need to be made in designing a particular implementation of the AμP. These include the number of registers, the type of arithmetic operations performed by the ALU (for instance, a complete four-quadrant analog multiplier could be implemented or the logarithm operation could be considered), the instruction set, I/O interface, etc. The design tradeoffs between processor area, speed, power dissipation, and accuracy also need to be resolved. Our implementation, apart from providing the proof-of-concept of the AμP architecture, serves as a prototype in the development of a processor array, intended for low-level image processing [17]. Analog solutions are efficient in performing various early-vision tasks [18], but even the programmable ones [19]–[21] lack the flexibility of digital processors. On the other hand, massively parallel SIMD (Single Instruction Multiple Data) arrays of mesh-connected digital processing elements have long been known to be efficient in executing a wide range of early-vision algorithms [22], [23]. The area-efficient implementation of a processing element is of primary importance, as it enables the integration of thousands of processors onto a single die, and thereby fully exploit the inherent fine-grain parallelism of early-vision tasks by realizing pixel-per-processor correspondence [24]. Also, the power consumption must be kept within certain limits. On the other hand, the mod-

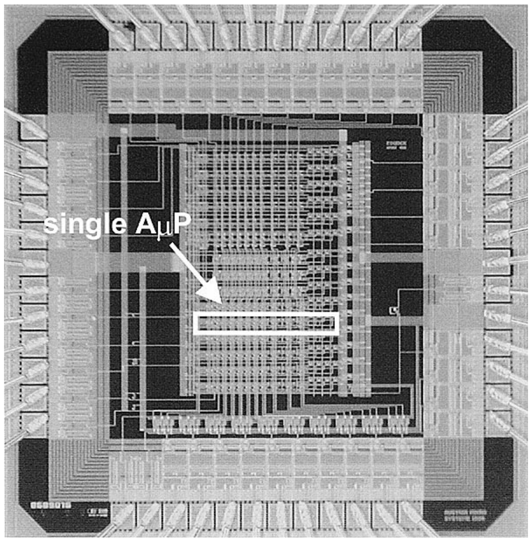

Fig. 3. Chip microphotograph. Shown in a box is a single processor employing the S^{2}I error cancellation scheme. To simplify routing of control signals to various types of processors, the layout of each processor is distributed within the 600 μm × 70μm area, which also includes some additional test structures. However, for the S^{2}I processor, the effective area occupied by the circuitry and routing is equal to 11200 μm^2

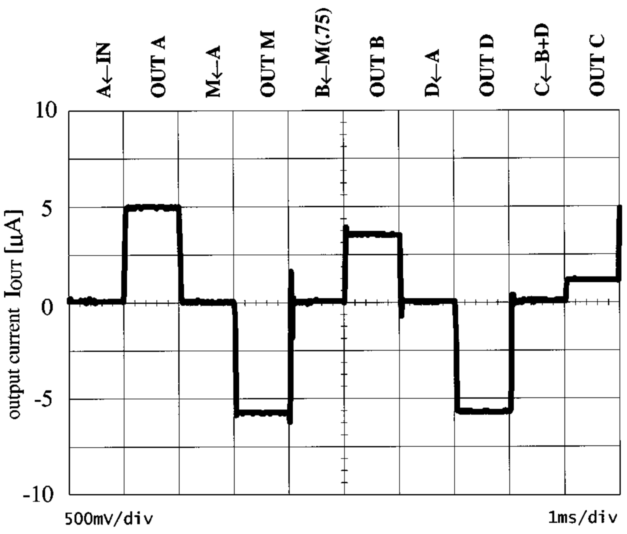

Fig. 4. Example program running on the AμP. The operations being performed are denoted above the output waveform. The result of each consecutive operation was fed to the output port and observed on the digitizing oscilloscope after I/U conversion.

TABLE III
PERFORMANCE/AREA AND PERFORMANCE/POWER RATIOS FOR VARIOUS PROCESSORS AND THE AμP

Processor	Ref.	MOPS / mm^2	GOPS / W	Processor type /technology	Technical data from which figures are derived
AμP	-	110	12.5	S^{2}I analogue microprocessor (0.8μm CMOS)	Area 11200μm^2 (AμP with 6 analogue registers), power 100μW, speed 1.25 MOPS for two-cycle analogue arithmetic operations
CNN-UM	[29]	81.8	4.37	Single cell from the CNN-UM array with grey-scale I/O (0.8μm CMOS)	Area 232x263.5μm^2, power 0.3W per 262 cells, peak performance 200ns per CNN-transient operation
Pixel-Parallel	[24]	64.1	9.34	Bit-serial PE from a massively parallel SIMD processor array (0.6μm CMOS)	Area 10661.76 μm^2 (PE with 128-bits storage), power 0.3W per 64x64 processors chip, performance 2.8GOPS/chip for 8-bit additions.
IMAP	[27]	36.3	3.67	8-bit PE from a processor-in-memory SIMD array (0.55μm BiCMOS)	Area 419x2628μm^2 (PE with 17 registers), power 0.7W per 64 PE's, performance 40MIPS/processor (for 8-bit instructions)
DEC Alpha 21064	[28]	1.72	0.013	High-performance 64-bit & floating point RISC microprocessor. (0.7μm CMOS)	Area 16.8x13.9mm^2, power 30W, performance 0.4GOPS. These figures describe the complete chip, i.e. include floating point unit, control unit, I/O, etc.

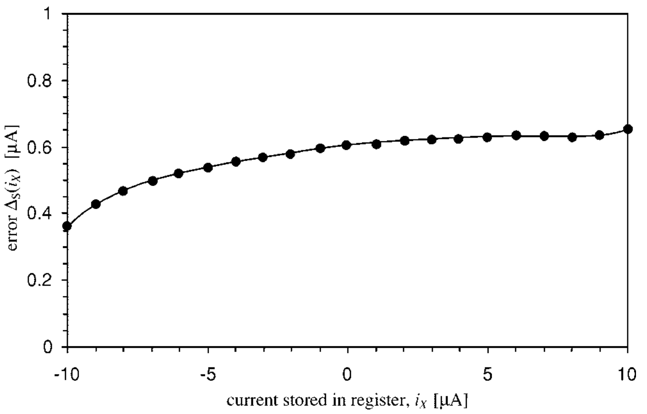

Fig. 5. Error of the single transfer operation as a function of signal value for the S^{2}I AμP.

erate accuracy, equivalent to 5 or 6 bits, is adequate for the majority of low-level image processing tasks. Our analysis of various image processing algorithms, including convolution, smoothing, edge detection, segmentation, median filtering, motion detection, etc., shows that usually only multiplication by a constant is required, and even a multiplier resolution of 3 to 4 bits can be sufficient. Six to eight registers are also thought to be sufficient.

To allow the evaluation of different design tradeoffs, we have designed and fabricated an integrated circuit containing 15 AμPs using various error-cancellation methods and different transistor sizes. These include a processor with no compensation, processors with half-size dummy devices [25], processors with regulated-cascode output stages [26], processors with an op-amp feedback loop [12], and processors utilizing the S^{2}I technique. The silicon area occupied by a basic register cell varies therefore from 17 μm $\times$ 39 μm to 54 μm $\times$ 57 μm. The chips were fabricated through EUROPRACTICE using the standard 0.8-μm CMOS process from AMS. The chip microphotograph is presented in Fig. 3. The AμP circuits operate with a 3.3-V power supply voltage and were tested, performing various algorithms, using a laboratory data generator as an external controller.

B. Measurements

For different processor designs, we have obtained magnitudes of the signal-dependent error of a single instruction from 0.2% to 3.5%, with processors operating at speeds from 70 kHz to 4 MHz. Below we report the measurement results for one of our AμPs, built using the S^{2}I error-compensation technique. The processor has six registers and a 3-bit multiplier. The nominal reference current level is equal to 10 μA. Fig. 4 shows the output of the example program, considered in Section III, running on this AμP. The result of each instruction is fed to the output port and the output currents are observed on the oscilloscope after a current–voltage conversion. Note the signal-independent error cancellation. It is apparent that even with a fairly significant signal-independent error of approximately 0.6 μA (visible on negative currents) there is relatively little overall error in the results of arithmetic operations (positive currents). For illustration purposes, in order to obtain clear waveforms without settling oscillations of the I/U converter used, the clock frequency was reduced to 1 kHz. However, the processor works satisfactorily with a clock frequency of up to 2.5 MHz. The total power dissipation within the processor is less than 100 μW. (To reduce power consumption only current sources that are required by a particular instruction are enabled). The effective area occupied by the processor is equal to 11 200 μm^2. As the typical assignment or arithmetic operation will take two clock cycles, the performance/area ratio for this processor is equal to 0.11 giga operations per second (GOPS) per square millimeter. The performance/power ratio is equal to 12.5 GOPS/W.

In Table III, these figures of merit are compared with ones derived for digital processing elements of SIMD arrays [24], [27], a RISC microprocessor [28], and a CNN-UM [29]. Comparing "operations per second" for different architectures can be difficult as, for example, the RISC microprocessor is obviously a much more complex and versatile device than the AμP. Nevertheless, simple digital processing elements intended for massively parallel SIMD arrays provide similar levels of functionality to the AμP (i.e., small number of registers, simple ALU, off-chip controller). The implementation of the CNN-UM chosen for comparison supports analog I/O. (Most other implementations perform only transformations of binary arrays). To facilitate a fair comparison, details of how the performance/area and performance/power figures were derived are given in Table III. As can be seen, the AμP compares quite favorably with other approaches. It has to be recognized, however, that the AμP has the drawback of a limited accuracy.

The error measured for a register transfer instruction, for the S^{2}I processor, is plotted as a function of signal value in Fig. 5. The magnitude of the signal-dependent error $\Delta_{\rm SD}$ is equal to 250 nA, that is 2.5% of the maximum signal level of 10 μA. Random errors associated with a single transfer $\Delta_N(*)$ were measured to be equal to 40-nA rms, that

```
A <- in Pixel[x-1,y-1]
B <- in Pixel[x-1,y]
C <- in Pixel[x-1,y+1]
F <- A+B+C
D <- F
A <- in Pixel[x+1,y-1]
B <- in Pixel[x+1,y]
C <- in Pixel[x+1,y+1]
F <- A+B+C
B <- F + D
E <- B
IF (B)
   E <- F + D
ENDIF
A <- in Pixel[x-1,y-1]
B <- in Pixel[x,y-1]
C <- in Pixel[x+1,y-1]
F <- A+B+C
D <- F
A <- in Pixel[x-1,y+1]
B <- in Pixel[x,y+1]
C <- in Pixel[x+1,y+1]
F <- A+B+C
B <- F + D
A <- B
IF (B)
   A <- F + D
ENDIF
C <- A + E
```

Fig. 6. Sobel edge-detection program for serial image processing on a single AμP. The above program is performed for each pixel in the input image.

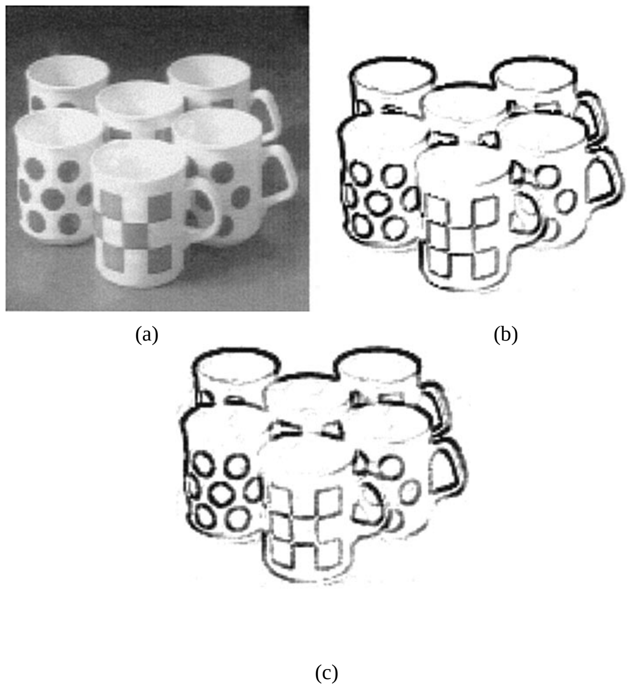

Fig. 7. (a) An input image. (b) Results of the ideal Sobel edge detection. (c) Results obtained by the execution of the Sobel algorithm on the AμP.

is 0.4% of the maximum signal level. At room temperature, the analog value held in the register decays due to the leakage currents at a rate of 0.5% per 100 ms.

The design tradeoffs were resolved in favor of small cell size, which resulted in small memory-transistor capacitances and in consequence relatively large errors. Nevertheless, many applications, particularly in low-level image processing, are not very sensitive to the errors introduced by the AμP. As an example application, consider the edge-detection problem. A software for the AμP, implementing the Sobel edge-detection algorithm [30] has been written and executed on the S^{2}I processor. The program listing is presented as Fig. 6. Fig. 7(a) contains a test image. The theoretical Sobel edge-detection result and the edge map obtained by executing the algorithm on the AμP are depicted in Fig. 7(b) and (c), respectively. The pixel-wise difference of the edge maps is equal to 9.64% (rms value). However, in the context of computer vision, detecting the overall location of edges is more important than having a low absolute error as compared with an "ideal" algorithm. It can be seen that the locations of the edges are accurately detected by the AμP. In this example, the image was processed serially on a single processor clocked at 2.5 MHz. Pixel values were fed to the processor using a D/A converter, and the result read out using an A/D converter. The processing speed was therefore relatively low. However, small cell size and low power dissipation are the key features that enable massive parallelism. A very high performance system could be built by integrating a large number of processors. An SIMD array of 128$\times$ 128 such AμPs could be feasibly accommodated on a single die and, when clocked at 2.5MHz, perform algorithms with a speed of above 20 GOPS while dissipating less than 2 W of power. The parallel version of the Sobel edge-detection program, intended for the AμP array, contains 49 instructions. The edge detection would be performed over the entire image within 20 μs.

VI. Conclusion

We have presented a general-purpose analog microprocessor whose architecture is analogous to that of its digital counterpart. The AμP executes software programs while operating on analog data values. The AμP paradigm will find application in areas that can benefit from employing analog signal processing techniques, but where nevertheless the flexibility of a software-programmable device is needed.

The operation of the AμP is characterized by a limited accuracy. However, the design tradeoffs may be resolved in various ways. For example, an AμP intended for general-purpose signal filtering applications could use SI circuits of greater complexity, but much reduced errors [15]. In our implementation, we have considered a processor array, targeted at image processing applications. The high-performance/area and performance/power ratios exhibited by the SI AμP will allow for a great number of processors to be integrated onto a single chip, resulting in the development of low-cost high-performance systems.

References

[1] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.
[2] L. H. Lu and C. Y. Wu, "The design of the CMOS current-mode general purpose analog processor," in *Proc. Int. Symp. Circuits and Systems (ISCAS'94)*, vol. 5, New York, 1994, pp. 549–552.
[3] D. L. Grundy, "A computational approach to VLSI analog design," *J. VLSI Signal Processing*, vol. 8, pp. 53–60, 1994.
[4] D. R. D'Mello and P. G. Gulak, "Design approaches to field-programmable analog integrated circuits," *Anal. Integr. Circuits Signal Processing*, vol. 17, no. 1–2, pp. 7–34, Sept. 1998.
[5] E. K. F. Lee and W. L. Hui, "A novel switched-capacitor based field-programmable analog architecture," *Anal. Integr. Circuits Signal Processing*, vol. 17, no. 1–2, pp. 35–50, Sept. 1998.
[6] H. Kutuk and S. M. Kang, "A switched-capacitor approach to field-programmable analog array (FPAA) design," *Anal. Integr. Circuits Signal Processing*, vol. 17, no. 1–2, pp. 51–65, Sept. 1998.
[7] A. Bratt and I. Macbeth, "DPAD2—A field programmable analog array," *Anal. Integr. Circuits Signal Processing*, vol. 17, no. 1–2, pp. 67–89, Sept. 1998.
[8] R. Herken, Ed., *The Universal Turing Machine. A Half-Century Survey*, Oxford, U.K.: Oxford Univ. Press, 1988.

[9] S. Masuda, S. Yoneda, and T. Kasai, “Sampled-data charge processor,” *Int. J. Electron.*, vol. 58, no. 5, pp. 743–760, May 1985.
[10] T. Roska and L. O. Chua, “The CNN universal machine: An analogic array computer,” *IEEE Trans. Circuits Syst. II*, vol. 40, pp. 163–173, Mar. 1993.
[11] *Anal. Integr. Circuits Signal Processing, Special Issue*, vol. 15, no. 3, Mar. 1998.
[12] C. Toumazou, J. B. Hughes, and N. C. Battersby, Eds., *Switched-Currents: An Analogue Technique for Digital Technology*, London, U.K.: Peregrinus, 1993.
[13] D. M. Leenaerts, G. H. M. Joordens, and J. A. Hegt, “A 3.3 V 625 kHz switched-current multiplier,” *IEEE J. Solid-State Circuits*, vol. 31, pp. 1340–1343, Sept. 1996.
[14] G. Wegmann, E. A. Vittoz, and F. Rahali, “Charge injection in analog MOS switches,” *IEEE J. Solid-State Circuits*, vol. 22, pp. 1091–1097, Dec. 1987.
[15] W. Guggenbühl, J. Di, and J. Goette, “Switched-current memory circuits for high precision applications,” *IEEE J. Solid-State Circuits*, vol. 29, pp. 1108–1116, Sept. 1994.
[16] J. B. Hughes and K. W. Moulding, “S^2I: A switched-current technique for high performance,” *Electron. Lett.*, vol. 29, no. 16, pp. 1400–1401, Aug. 1993.
[17] P. Dudek and P. J. Hicks, “An SIMD array of analogue microprocessors for early vision,” in *Proc. Conf. Postgraduate Research in Electronics, Photonics and Related Fields (PREP’99)*, Manchester, U.K., 1999, pp. 359–362.
[18] A. Moini, *Vision Chips*. Norwell, MA: Kluwer, 1999.
[19] P. Kinget and M. Steyaert, “A programmable analogue CMOS chip for high speed image processing based on cellular neural networks,” in *Proc. IEEE 1994 Custom Integrated Circuits Conf.*, New York, 1994, pp. 570–573.
[20] T. Spirig, P. Seitz, O. Vietze, and F. Heitger, “A smart CCD image sensor with real-time programmable parallel convolution capabilities,” *IEEE Trans. Circuits Syst. I*, vol. 44, pp. 465–468, May 1997.
[21] D. A. Martin, H. S. Lee, and I. Masaki, “A mixed-signal array processor with early vision applications,” *IEEE J. Solid-State Circuits*, vol. 33, pp. 497–502, Mar. 1998.
[22] K. E. Batcher, “Design of a massively parallel processor,” *IEEE Trans. Comput.*, vol. 29, pp. 837–840, Sept. 1980.
[23] M. J. B. Duff and T. J. Fountain, Eds., *Cellular Logic Image Processing*. New York: Academic, 1986.
[24] J. C. Gealow and C. G. Sodini, “A pixel-parallel image processor using logic pitch-matched to dynamic memory,” *IEEE J. Solid-State Circuits*, vol. 34, pp. 831–839, June 1999.
[25] C. Eichenberger and W. Guggenbühl, “Dummy transistor compensation of analog MOS switches,” *IEEE J. Solid-State Circuits*, vol. 24, pp. 1143–1146, Aug. 1989.
[26] C. Toumazou, J. B. Hughes, and D. M. Pattullo, “Regulated cascode switched-current memory cell,” *Electron. Lett.*, vol. 26, no. 5, pp. 303–305, Mar. 1990.
[27] N. Yamashita, T. Kimura, Y. Fujita, Y. Aimoto, T. Manabe, S. Okazaki, K. Nakamura, and M. Yamashina, “A 3.84 GIPS integrated memory array processor with 64 processing elements and a 2-Mb SRAM,” *IEEE J. Solid-State Circuits*, vol. 29, pp. 1336–1343, Nov. 1994.
[28] D. W. Dobberpuhl, R. T. Witek, R. Allmon, R. Anglin, D. Bertucci, S. Britton, L. Chao, R. A. Conrad, D. E. Dever, B. Geiseke, S. M. N. Hassoun, G. W. Heoppner, K. Kuchler, M. ladd, B. M. Leary, L. Madden, E. J. McLean, D. R. Meyer, J. Montanaro, D. A. Priore, V. Rajagopalan, S. Samudrala, and S. Santhanam, “A 200-MHz 64-b dual-issue CMOS microprocessor,” *IEEE J. Solid-State Circuits*, vol. 27, pp. 1555–1567, Nov. 1992.
[29] J. M. Cruz and L. O. Chua, “A 16×16 cellular neural network universal chip: The first complete single-chip dynamic computer array with distributed memory and with gray-scale input-output,” *Anal. Integr. Circuits Signal Processing*, vol. 15, no. 3, pp. 227–237, Mar. 1998.
[30] E. R. Davies, *Machine Vision: Theory, Algorithms, Practicalities*. New York: Academic, 1990.

Four-Quadrant CMOS Current-Mode Multiplier Independent of Device Parameters

Koichi Tanno, Okihiko Ishizuka, and Zheng Tang

***Abstract*—In this brief, we present a four-quadrant CMOS current-mode multiplier based on the square-law characteristics of an MOS transistor operated in the saturation region. One advantage of this multiplier is that the output current is independent of MOS transistor device parameters; another, that the input resistance is independent of the input current. Simulations of the multiplier demonstrate a linearity error of 1.22%, a THD of 1.54%, a −3-dB bandwidth of 22.4 MHz, and a maximum power consumption of 0.93 mW. Operation of the multiplier was also confirmed through an experiment using CMOS 4007 IC’s.**

***Index Terms*—Analog integrated circuits, circuit theory and design, CMOS LSI, current-mode, multiplier.**

I. Introduction

Analog multipliers are composed of the essential circuit elements of analog signal processing systems, and include a modulator, frequency doubler, rectifier, variable gain amplifier, etc. Recently, some CMOS analog multipliers using current-mode technique have been proposed for fuzzy controllers and neural networks [1]–[5]. Currently, there are four principal approaches to realize current-mode analog multipliers.

The first approach is to use the current-controlled resistor, which consists of only three MOS transistors. However, this requires that the transconductance parameter of the pMOS transistor be identical to that of the nMOS transistor, which is very difficult because the values of these transistors strongly depend on the fabrication process and vary lot by lot. The second approach is to use the translinear principle of MOS transistors operated in the subthreshold region [2], [3]. This approach has the advantage of low power consumption. However, the dynamic range of this circuit is very small and operation speed is slow. In an effort to eradicate shortcomings, Liu *et al.* proposed a third approach, which uses the current-mode square root and squarer circuits based on the square-law characteristics of MOS transistors operated in the saturation region [4]. However, the power consumption by this approach is large; the multiplier requires bias currents for all input signals to make MOS transistors operate in the saturation region. The last approach uses the class AB current-mode cell, which consists of only two MOS transistors [5]. Because this circuit operates as the class AB multiplier, it does not need bias currents for input signals. However, similar to the first approach, it requires that the transconductance parameter of the pMOS transistor be identical to that of the nMOS transistor.

In this brief, a novel four-quadrant CMOS current-mode analog multiplier is proposed. It is based on the square-law characteristics of an MOS transistor operated in the saturation region. The advantages of this multiplier are that the output current is independent of MOS transistor device parameters and that bias currents for input signals are not required. An additional and important advantage is that the input resistance is independent of the input current. The performance of this multiplier is characterized through HSPICE simulations with 2.0-μm p-well CMOS parameters (BSIM3 parameters). Its operation is also confirmed through an experiment using CMOS 4007 IC’s.

Manuscript received March 1999; revised January 2000. This paper was recommended by Associate Editor H. Tanimoto.

The authors are with the Department of Electrical and Electronic Engineering, Faculty of Engineering, Miyazaki University, Miyazaki-shi 889-2192, Japan.

Publisher Item Identifier S 1057-7130(00)04203-8.