14th International Conference
MIXDES 2007
Ciechocinek, POLAND
21 – 23 June 2007

ANALOG, CONTINUOUS TIME, FULLY PARALLEL, PROGRAMMABLE IMAGE PROCESSOR BASED ON VECTOR GILBERT MULTIPLIER

R. DŁUGOSZ[1,2,3]
[1] UNIVERSITY OF NEUCHÂTEL, SWITZERLAND
[2] UNIVERSITY OF ALBERTA, CANADA
[3] POZNAŃ UNIVERSITY OF TECHNOLOGY, POLAND

KEYWORDS: Analog parallel image processor, Gilbert multiplier

ABSTRACT: A conception as well as a CMOS implementation of the analog, ultra low power and fully parallel image processor have been presented in this paper. Proposed circuit bases on the 2-D FIR filters realized using the Gilbert vector multiplier. Proposed filter enables realization of various lowpass and highpass 2-D FIR filter masks. Both the mask dimensions and values of the filter coefficients can be programmed using several dozen digital signals and several DC currents. Proposed image processor does not use the clock generator, what simplifies the overall circuit's structure and reduces the noise level. An example (6x6) image processor that enables filtering with a 3x3 mask has been implemented in CMOS 0.18 μm process. This circuit calculates 36 pixels in parallel every 1 μs, dissipating power about 20 μW. The image resolution can be easily enlarged by a parallel connection of many designed 6x6 cells.

INTRODUCTION

Two dimensional filtration is commonly used in image processing systems in such operations as: decimation, interpolation, edge detection and many others [2, 3, 4]. The image filters are usually realized as digital systems, although many examples of the analog implementations have been also reported. The most common realization examples of the analog image processors base on the Cellular Neural Networks (CNN) [2]. In CNNs the neighbouring nodes are in common relation, exchanging data in the convergence process that finally leads to a stable 2D output signal. The very interesting CMOS implementation of the CNN image processor has been described in [2]. In this solution all pixels of the output image with resolution (64x64) are calculated in parallel. This processor is a voltage mode circuit designed in CMOS 0.5μm process. The circuit dissipates power of 1.5 W from the 3.3 V voltage supply for analog data rate that is equal to 1 MSamples/s.

In this paper a quite different approach to realization of the parallel analog image processors has been presented. Proposed circuit is a 2D FIR filter that bases on the vector Gilbert multiplier (VGM) operating in the current mode using transistors working in subthreshold region. The proposed image processor works with continuous time signals, without using the clock generator enabling fully parallel processing of the input signals. In this approach, as opposed to CNN realizations, there is no data exchange between the neighbouring nodes. Calculation of the output data does not need the recursive process, as it is in CNN solutions. As a result, proposed circuit enables higher data rates. The power dissipation depends linearly on number of the output pixels and for example case of circuit with output image resolution (6x6) is equal to about 20 μW for VDD = 0.65 V. This means that for the image resolution of (64x 64) power dissipation will be equal to about 3 mW. The data rate in designed circuit (6x6) is equal to about 36 MSamples/s. Time required to calculate a single pixel is constant and does not depend on the image resolution.

The paper is organized as follows. The principle of the 2-D FIR filtration has been briefly described in section 2. As this topic is widely described in literature, here only same necessary aspects have been provided to explain the idea of proposed circuit that has been described in section 3. The CMOS implementation with simulation results are described in section 4. Finally, the conclusions are drawn in section 5.

2D-IMAGE FIR FILTRATION

Image can be treated as a two-dimensional signal, where particular pixels are samples of this signal. Domain of this signal is defined as the x and y image coordinates. Brightness of a given pixel is treated as a value of this signal sample. In 2D FIR filtration, particular pixels are multiplied by the filter coefficients in 2D mask H and then summed producing samples of the output image. This principle is illustrated in Fig. 1. For the example filter mask with dimensions (3, 3) the output samples are calculated using the following equation:

$$B(x,y)=\sum_{n=1}^{3}\sum_{m=1}^{3}A(x+n-2,y+m-2)h(n,m) \qquad (1)$$

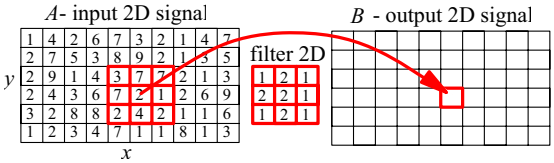

Fig. 1. 2-dimensional FIR image filtration

The mask shown in Fig. 1 is an example lowpass filter, which can be used as an anti-aliasing filter before image decimation or as an anti-imaging filter after image interpolation. On the other hand, the highpass filters are typically used in edge detection. The following example highpass FIR filters can be used in edge detection:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \qquad (2)$$

(a) (b)

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

(c) (d)

2-DIMENSIONAL VGM FIR FILTERS

The vector Gilbert multipliers (VGM) are standard circuits used in different applications. The important basic details of this class of circuits have been provided in [1]. In this case the VGM circuits with transistors working in weak inversion have been successfully implemented in CMOS technology in ultra low power analog decoders, where are used as soft XOR gates.

In presented work the VGM circuits are used in the parallel 2D FIR filtration. The VGM circuits provide normalization of the filter coefficients, what is very important in this case.

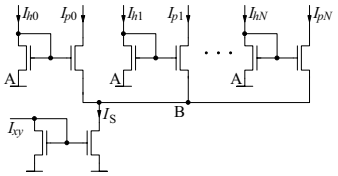

Fig.2 Multiplication of scalar and vector with normalization of vector (SVGM)

$$P_{xy} = \frac{I_{xy} H^{\mathrm{T}}}{\sum_{i=0}^{N} h_i} = I_{xy} \left| H \right|^{\mathrm{T}} \qquad (3)$$

The basic cell that is used in proposed image processor is the scalar-vector Gilbert multiplier (SVGM) that is shown in Fig. 2 and described by equation (3). This circuit calculates currents I_p, which are normalized products of a given input sample $A(x, y)$ delivered to the input I_{xy} and the vector H of currents I_h, which represent the filter coefficients. Length of the output vector P_{xy} is determined by number of the coefficients, which have unique values. For example, in case of filter masks that are shown in Fig. 1, and are given by equation (2), length of the vector P_{xy} is equal to only 2, as many coefficients in the mask have the same values.

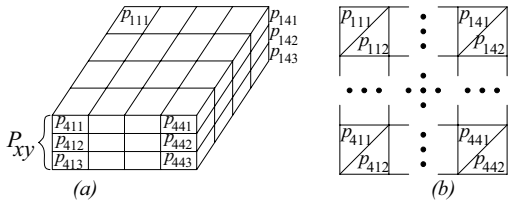

Fig. 3. Presentation of the VGM's output currents products "p" in two different ways.

Let us assume that the input image A has dimensions (X, Y) and length of vector P_{xy} is Z. In this case number of SVGM circuits is equal to $X \cdot Y$, and finally we get $X \cdot Y \cdot Z$ products I_p. These currents are for convenience denoted further as p_{xyz}, and are used as components in calculation of samples of the output image B.

Although particular SVGM circuits are independent one in respect to each other and their outputs can be presented in any way, this is very convenient to think about products p as they were organized into 3D matrix P with dimensions (X, Y, Z) as shown in Fig. 3 (a). This matrix will be further presented rather in form shown in Fig. 3 (b).

Assuming that filter mask has dimensions (N, M), then particular elements of each vector P_{xy} are used as components in no more than $N \cdot M$ output samples. This means that each product p_{xyz} must be copied into $N \cdot M$ independent output branches, using PMOS-type current mirrors. Currents flowing in these branches are summed, producing samples of the output image. Summing in current mode circuits is realized by use of simple junctions, what is an important advantage here.

Each vector P_{xy} contains $(N \cdot M \cdot Z)$ PMOS output transistors, resulting in the same number of the output branches organized into connection map. This map can be programmed using the digital signals, which enable realization of various 2D filters. It is worth noting that typically only several branches are used in a given filter mask. Although each input sample $A(x,y)$ is multiplied by each element of a given vector P_{xy}, typically only one from the Z resulting products in the P_{xy} vector is used to calculate the output sample $B(x,y)$. Other branches are broken by the digital programming signals. Number of the output PMOS transistors per one pixel is relatively large, what is certain disadvantage, but these transistors have small dimensions. Proposed circuit has various important advantages. One of them is that map of connections is programmed off-line and is then fixed during the filter operation. This means that a given copy of a given product p_{xyz} is always connected to the same output sample. As a result, for chip programmed at the beginning there is no switching over during the circuit operation. The clock generator is not required, what simplifies the overall circuit structure and minimizes the level of noise that would be generated by clock.

Map of connections is constructed in the way that enables fully parallel data processing, what means that all output pixels are calculated in the same time. This is the second very important advantage of proposed circuit, which increases the data processing rate.

Programming the circuit

There are several parameters in the circuit, which must be programmed to enable realization of different filter masks. The first type of parameters are absolute values of the filter coefficients, which are represented by DC currents. Filter masks in 2D filtration are typically the symmetrical structures. As a result, the maximum number of the filter coefficients which have unique values is equal to $N \cdot M / 4$ for even values of N and M and to $(N+1) \cdot (M+1) / 4$ for odd values of N and M. Number of coefficients with unique values must be determined during design of the circuit.

The second element, which must be programmed, is the map of connections described earlier. This map is programmed using digital signals that are organized into the 4-dimensional matrix D (N, M, Z, 2). To explain the meaning of the first two dimensions we must remind that output samples $B(x,y)$ are calculated in accordance

with equation (1) as sums of *elements*, which are products of chosen input samples $A(x, y)$ and given filter coefficients in 2D mask, as shown in Fig. 1. These *elements* are here given products p taken from 3D matrix P. Now, if sample $B(x,y)$ is being calculated, then bits, which in matrix D are in a given position (n, m) determine that given products p in neighborhood of this sample are given *elements* from equation (1). When, for example, sample $B(2,5)$ is being calculated then bits $D(1,1,..,..)$ tell that products p in vector $P(1,4)$ form *element* (n=1, m=1) in equation (1). The same products $P(1,4)$ for other sample $B(2,4)$ form *element* (n=1, m=2) and are controlled by bits, which in matrix D are in the position $D(1, 2,..,..)$.

The third dimension Z determines which product p_{xyz} within a given vector P_{xy} is connected to a given sample $B(x,y)$. The last dimension, denoted here by 2, is necessary as each product p_{xyz} is described by two bits. The first bit determines if a given product p_{xyz} is added with a positive sign, whereas the second one determines if a given product is added with the negative sign. If both bits are zero then given product is not used in calculation of a given output sample. Chosen examples of matrix D are shown in Fig. 4. In these examples we assume that filter mask has dimensions (3, 3). In the first case (a) matrix D is set up to realize (2, 2) highpass edge detecting filter described by equation (2d), whereas in the second case (b) is set up to realize (3, 3) lowpass filter shown in Fig. 1. In both cases number of unique coefficients is only 2, resulting in Z being equal to 2. The first case illustrates that filter with dimensions, which are smaller than the maximum possible values can be realized by zeroing of chosen bits in matrix D.

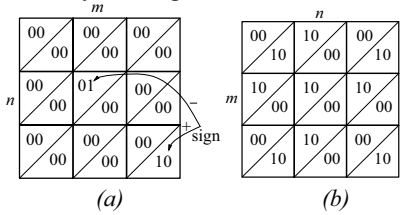

Fig.4. Examples of matrix D for two different filters

There exists an interesting way to increase the programming possibilities, as all bits d in matrix D are independent one in respect to each other. Each position (n, m) in the matrix D contains an area of $(Z \cdot 2)$ bits, where each of these bits controls a given connection in the map. So far we assumed that only one bit in each area $(Z \cdot 2)$ is equal to 1, but setting more than one bit to the value 1 enables increasing an effective number of unique coefficients. To illustrate this idea let us assume that in 3-element vector (Z=3) of the filter coefficients particular elements have the following values: 1, 3, 8. As the negated values are also available, this file can be extended to: 1, 3, 8, 0, -1, -3, -8. Now by combining these values in different ways we get the following coefficients (their negations are realized automatically):

1, **2** (as 3-1) [d_{xy12}=1 and d_{xy21}=1], **3**,
4 (as 3+1) [d_{xy11}=1 and d_{xy21}=1],
5 (as 8-3) [d_{xy31}=1 and d_{xy22}=1],
6 (as 8+1-3) [d_{xy31}=1, d_{xy11}=1 and d_{xy22}=1],
7 (as 8-1), **8**, **9** (8+1), **10** (8+3-1), **11** (8+3), **12** (8+3+1)

As a result, for $Z = 3$ we get 24 different coefficients, what is sufficient in many filtering tasks.

Taking all above considerations into account, the output samples $B(x, y)$ are calculated using the following general equation:

$$B(x,y) = \sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{l=1}^{L} d_{nml1} \cdot p_{x+n-2,y+m-2,l} + \\ - \sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{l=1}^{L} d_{nml2} \cdot p_{x+n-2,y+m-2,l} \tag{4}$$

To illustrate described earlier 3D filter, an example map of connections has been shown in Fig. 5. Particular arrows are connections in the map controlled by bits d. In this example the filter mask has dimensions (2, 2). Length of vectors P_{xy} has been also selected to be equal to 2 ($Z = 2$). Resulting matrix P has dimensions (X, Y, 2) and programming matrix D has dimensions (2, 2, 2, 2). In example case when filter mask is given by equation (2d), then equation (4) can be simplified to:

$$B(x,y) = -1 \cdot p_{x,y,1} + 1 \cdot p_{y+1,y+1,2} \tag{5}$$

For example, the output sample $B(2,4)$ is calculated as:

$$B(2,4) = -1 \cdot p_{241} + 1 \cdot p_{352} \tag{6}$$

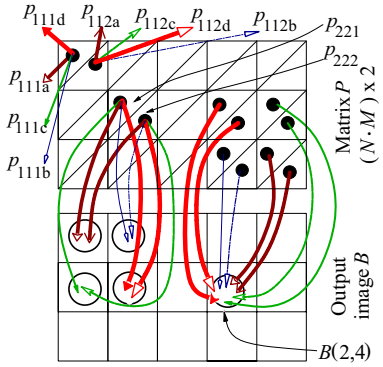

Fig. 5. Calculation of the output samples on the basis of matrix P in 2D filtration using the VGM filter

Fig. 5 illustrates two important aspects. One of them is the way, how particular output samples $B(x, y)$ are calculated.

The second aspect is that given products p_{xyz} are copied many times to different output samples $B(x, y)$. Particular copies of given products p are denoted by an additional letter (a, b, c, d). For example, element p_{221} as well as element p_{222} can be used in calculation of the output samples: $B(1, 1)$, $B(1, 2)$, $B(2, 1)$ and $B(2, 2)$. In case of sample $B(1, 1)$, the product p_{221} in used in calculation of this sample what is determined by bit d_{1111}. Bit d_{1112} determines if product p_{222} is used. In case of the sample $B(1, 2)$ product p_{221} is also used, but in this case this is determined by bit d_{1211}. Each output sample $B(x, y)$ uses another copy of the product p_{221}. For example, in case of sample $B(1, 1)$ the copy p_{111a} is used, whereas in case of sample $B(1, 2)$ the copy p_{111b} is used. Connections between particular copies of given products p_{xyz} and given output samples $B(x, y)$ are made using switches controlled by particular bits d, where each bit d controls many configuration switches in the

map. For example, bit d_{1211} controls connection between product p_{251} and the output sample $B(2,4)$, but also between product p_{151} and the sample $B(1,4)$ and so on. This is very important feature as for a given matrix D all connections in the map are fixed.

The 2D image filtration can be illustrated as moving the filter mask over the input image, where output samples are calculated sequentially. In typical digital systems samples are calculated just in this way. In proposed solution, on the other hand, filter mask does need to move over the input image, as all output samples are calculated in the same time.

This is worth noting that in case when filter would be designed as a non-programmable structure, then instead using the programmable matrix D and configuration switches, the connections between given products p and given output samples would be permanent. In this case connection would be in places where in matrix D are values 1. As a result, the non-programmable filters have simpler structures, resulting in reduced chip area.

Implementation of the interpolation filter

Proposed filter can be easily used in interpolation of the 2D signal. Assume for example, that an input image A must be increased by 200% in each dimension. In this case we get the new image B shown in Fig. 6. Samples denoted as B_{xyA} are original samples from the input image A, whereas B_{xyB}, B_{xyC}, B_{xyD} are new calculated samples.

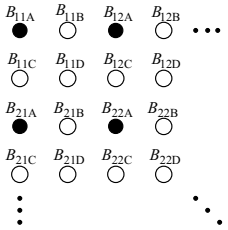

Fig. 6. Interpolation in the 2D domain.

Typically, first the new zero-valued samples must be inserted into the original signal, and then the anti-imaging filter with dimensions at least (3, 3) is used to calculate the new samples. When the VGM 2D FIR filter is used, the mask dimensions can be smaller e.g. (2, 2), and the interpolated image B is in this case calculated in four steps. In each step filter mask must be reprogrammed to different values. In proposed solution the zero-valued samples do not need to be inserted into the input signal, but the final effect is the same. In the first step the mask has coefficients given by equation (7a), what means that samples B_{xyA} are equal to the input samples A_{xy}. In the next step samples B_{xyB} are calculated using mask (7b), then samples B_{xyC} are calculated using mask (7c) and finally samples B_{xyD} are calculated using mask (7d). To enable reprogramming of the mask, particular bits in the matrix D instead constant values as it was in the previous case now must be the appropriate pulse signals. The additional problem here is that in each step current that represent the filter coefficient must have different value. In the first step current must be two times higher than in the second and in the third steps, and four times higher than in the 4th step. This can be realized by setting the input currents I_h to values that are in the relation: 4x 2x 1x. Now, in the first step bit d_{1111} is equals to 1. In the second step bits d_{1121} and d_{1221} are equal to 1. In the third step bits d_{1121} and d_{2121} are equal to 1 and in the last step d_{1131}=1, d_{1231}=1, d_{2131}=1 and d_{2231} must be equal to 1.

$$h_{11}=\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}\ \text{(a)} \qquad h_{12}=\begin{bmatrix}0.5 & 0.5\\ 0 & 0\end{bmatrix}\ \text{(b)} \qquad (7)$$

$$h_{21}=\begin{bmatrix}0.5 & 0\\ 0.5 & 0\end{bmatrix}\ \text{(c)} \qquad h_{22}=\begin{bmatrix}0.25 & 0.25\\ 0.25 & 0.25\end{bmatrix}\ \text{(d)}$$

CMOS IMPLEMENTATION OF EXAMPLE (6x6) IMAGE PROCESSOR

An example 2D FIR filter has been implemented in CMOS 0.18 μm technology and successfully tested in HSPICE simulations. Although the measurement results are not ready right now, but as the same VGM circuit was used in another application, which has been successfully measured [1], author believes that simulation results are credible in this case.

In this example circuit the filter mask has dimensions (3, 3) and is fully programmable. In this case the resolution of the input image has been selected as (6, 6), but this parameter can be easily increased. Structure of a single block that calculates products of a single vector P_{xy} (with all copies) is shown in Fig. 7.

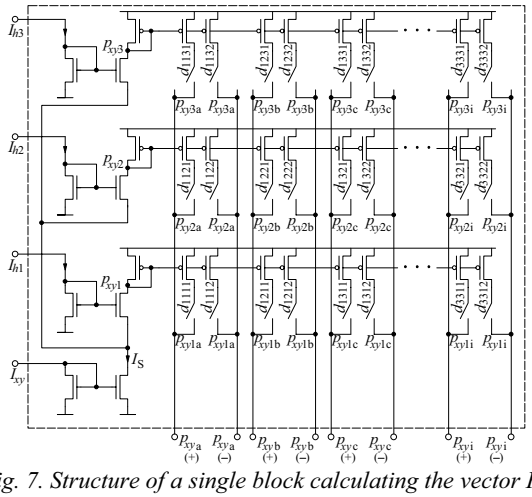

Fig. 7. Structure of a single block calculating the vector P_{xy}

The example processor contains 36 such cells that form the cluster (6, 6). Each cell is provided with the same filter coefficients represented by currents I_{hz}, as well as with an individual input pixel represented by current I_{xy}, producing output signals that are copies of particular products p_{xyz}. The output signals are denoted as (+) and (-), as part of them are used as negative or as positive components in calculated output samples $B(x, y)$.

This is worth noting that each sample $B(x,y)$ uses only one NMOS-type current mirror to turn sign of the signal in case of negative coefficients. All products p_{xy} (+) are connected to one node, whereas all products p_{xy} (-) to the second node and after turning the current direction, connected also to the first node p_{xy} (+). This approach minimizes number of transistors used in the circuit.

One additional problem here is related to realization of the high-pass filters (equation 2). Such filters remove the DC component from the output signal. The problem is important as the Gilbert multiplier must be biased by some DC current. To solve this problem after the highpass filtration the new DC current must be added directly to the output signal. This is realized by use of an additional current source.

0	0	0	0	0	0	0	0
0	200	200	200	200	200	200	0
0	200	200	200	200	200	200	0
0	200	200	500	500	200	200	0
0	200	200	500	500	200	200	0
0	200	200	500	500	200	200	0
0	200	200	200	200	200	200	0
0	0	0	0	0	0	0	0

Fig. 8. An example input image A surrounded by zeros used in verification of designed processor. Values are given in [nA]

Designed filter has been verified by implementation of different filter masks. Two example experiments have been selected for illustration. In both experiments the input signal shown in Fig. 8 is used. To enable testing the dynamic parameters of designed processor, pixels marked as "500" are the periodic pulse signals, with the frequency equal to 200 kHz. Signals oscillate between the values of 200 nA and 500 nA.

In the first experiment the lowpass (3, 3) filter, shown in Fig. 9 has been used. The resulting output image *B* is shown in Fig. 10. The time domain simulation results are shown in Fig. 11, where each panel illustrates pixels from a single column of the output image *B*.

1	2	1
2	4	2
1	2	1

Fig. 9. Example (3,3) filter mask used in the first experiment

193	255	255	255	255	193
255	365	418	418	365	255
255	418	574	574	418	255
255	445	653	653	445	255
255	418	574	574	418	255
193	282	335	335	282	193

Fig. 10. Output image B after filtration of example image A by the filter mask shown in Fig. 9. Values are given in nA

As we can see, all output pixels are calculated in parallel. The convergence time required to get the stable output signal is for a given input image equal to about 1 µs. This parameter can be additionally controlled by adjusting values of the input currents. This is worth noting, that presented circuit enables calculation also those signals, which are in direct neighbourhood of the samples that are on the border of the image i.e. $B(1,1)$ to $B(6,6)$. The additional calculated samples, denoted as: $B(0, 0)$ - $B(0, 7)$, $B(0, 0)$ - $B(7, 0)$, $B(0, 7)$ - $B(7, 7)$, $B(7, 0)$ - $B(7, 1)$, use products p_{xyz} that are in the border of the matrix *P*. These signals are not used when processor works as a single block, but when this circuit is used as a component of the bigger structure then signals coming from neighbouring blocks are combined into the additional pixels $B(x,y)$, what prevents the border effect visible in Fig. 10 (e.g. pixels $B(1,1)$, $B(6,1)$ - $B(6,6)$).

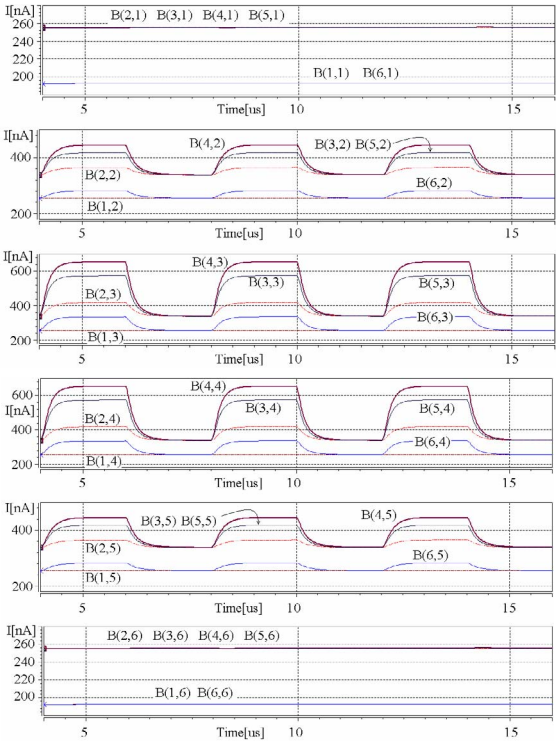

Fig.11. Pixels of the output image in the first experiment. The particular panels contain pixels from single image column.

0	0	0	0	0	0
0	1	3	3	1	0
0	3	9	9	3	0
0	4	12	12	4	0
0	3	9	9	3	0
0	1	3	3	1	0

Fig. 12. Output image B after filtration of a normalized image A using the filter mask shown in Fig. 9

There are also some issues related to normalization of the output data. If the input currents having values equal to 200 nA are normalized to the value 0, and those having values of 500 nA are normalized to the value 1, then we get the image shown in Fig. 12. Normalization of the output image *B* can be performed using equation (8). In this experiment parameter *a* is equal to 26 [-] while the parameter *b* is equal to –340 [nA]. As a result, we get the image shown in Fig. 13.

-5,65	-3,27	-3,27	-3,27	-3,27	-5,65
-3,27	0,96	3,00	3,00	0,96	-3,27
-3,27	3,00	9,00	9,00	3,00	-3,27
-3,27	4,04	12,04	12,04	4,04	-3,27
-3,27	3,00	9,00	9,00	3,00	-3,27
-5,65	-2,23	-0,19	-0,19	-2,23	-5,65

Fig. 13. Normalized output image B

$$B_{2norm}(x, y) = a \cdot B_2 + b \qquad (8)$$

Differences between the theoretical output image *B* and the normalized image are on the level of 0.35%. This

theoretically enables obtaining an 8 - bits grey scale (dynamic range 48 dB). This parameter will need to be confirmed in measurements after chip manufacture, as mismatch between transistors will affect to a certain degree the dynamic range. This may not be a serious problem, as in various applications e.g. in robotic and in medicine dynamic range about 25-30 dB is sufficient.

In the second experiment the high-pass filter, shown in Fig. 14, has been implemented. The resulting output image is shown in Fig. 15. The output image that is calculated theoretically on the basis of normalized input image A is shown in Fig. 16, whereas the normalized image B, calculated using equation (8) ($a = 47$ [-], $b = -196$ [nA]), is shown in Fig. 17. Time domain simulation results are shown in Fig. 18.

-1	0
0	1

Fig. 14. Highpass filter given by equation (2d) used in the second experiment.

196	196	196	196	196	150
196	243	243	196	196	150
196	243	193	147	196	150
196	243	193	147	196	150
196	196	147	147	196	150
150	150	150	150	150	150

Fig.15. Output image B after filtration of example image A by the filter mask shown in Fig. 14. Values are given in nA.

+

0	0	0	0	0	0
0	1	1	0	0	0
0	1	0	-1	0	0
0	1	0	-1	0	0
0	0	-1	-1	0	0
0	0	0	0	0	0

Fig. 16. Output image B after filtration of a normalized image A using the filter mask shown in Fig.14.

0,00	0,00	0,00	0,00	0,00	-0,98
0,00	1,00	1,00	0,00	0,00	-0,98
0,00	1,00	-0,06	-1,04	0,00	-0,98
0,00	1,00	-0,06	-1,04	0,00	-0,98
0,00	0,00	-1,04	-1,04	0,00	-0,98
-0,98	-0,98	-0,98	-0,98	-0,98	-0,98

Fig. 17. Normalized output image B.

SUMARRY

The new analog current mode image processor has been proposed, implemented in CMOS 0.18 µm process and verified in HSPICE simulations. Proposed circuit bases on the Gilbert vector multiplier, where transistors work in weak inversion. As a result, circuit features very low power dissipation on the level of 500 nW / pixel, enabling processing the data with the rate equal to about 1 Mpixel/s. Low power dissipation results from the fact, that sum of all currents p_{xyz} within a single vector P_{xy} is equal to the input current, typically being on the level of several hundreds nA. Energy used for calculation of a single pixel is below 1 pJ. Each cell (1 pixel) contains c.a. 120 transistors and occupies area equal to about 1000 µm^2. The proposed circuit is a fully programmable structure. The filter mask can be adjusted using several dozen digital signals and several DC currents (filter coefficients). The circuit enables easy implementation of anti-imaging filters.

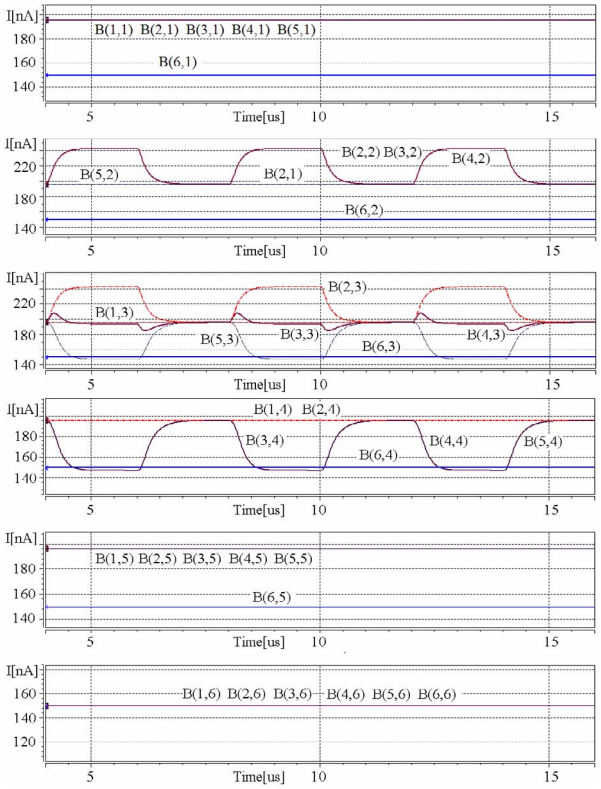

Fig.18. Pixels of the output image in the second experiment.

THE AUTHOR

Dr. Rafał Długosz
University of Neuchâtel, Institute of Microtechnology, Rue A.-L. Breguet 2, CH-2000, Neuchâtel, Switzerland; University of Alberta, Department of Electrical and Computer Engineering, 114 St – 89 Ave, Edmonton Albert-a, T6G 2V4, Canada, rdlugosz@ualberta.ca;
fellow of the MC EU Outgoing International Fellowship

REFERENCES

[1] Ch. Winstead, "Analog Iterative Error Control Decoders", Ph.D dissertation, University of Alberta, ECE Department, Edmonton, Alberta, 2004

[2] G. Linan, P. Foldesy, S. Espejo, R. Dominguez-Castro, A. Rodriguez-Vazquez, "A 0.5µm CMOS 10^6 transistors analog programmable array processor for real–time image processing", Proc. of the 25th European Solid-State Circuits Conference, 1999. ESSCIRC '99, pp:358 - 361

[3] K. Korekado, T. Morie, O. Nomura, T. Nakano, M. Matsugu, Iwata, A., "An image filtering processor for face/object recognition using merged/mixed analog-digital architecture", Symposium on VLSI Circuits, 2005 pp.220 – 223

[4] T. Hanyu, M. Arakaki, M. Kameyama, "2-Transistor-Cell 4-Valued Universal-Literal CAM for a Cellular Logic Image Processor" International Solid State Circuit Conference (ISSCC) 1997