CMOS Image Sensor Featuring Current-Mode Focal-Plane Image Compression

Fernanda D. V. R. Oliveira, Hugo L. Haas, José Gabriel R. C. Gomes and Antonio Petraglia

Universidade Federal do Rio de Janeiro – COPPE / Electrical Engineering Program
e-mail: gabriel@pads.ufrj.br

ABSTRACT

The interest in focal-plane processing techniques, by which image processing is carried out at the pixel level, has increased since the advent of active pixel sensors in the middle 90's. By sharing processing circuitry by a group of neighboring pixels such techniques enable high-speed imaging operation and massive parallel computation. Focal-plane image compression is particularly interesting, because it allows for further reduction in data rates. The proposed approach also benefits from processing currents rather than voltages, which not only suits current-mode APS imagers, but also enables the circuits to operate at low voltage supply levels and achieve high speed. Moreover, arithmetic computations such as additions and scaling are easily implemented in current mode. Whereas current-mode imaging architectures produce higher fixed pattern noise (FPN) figures than their voltage-mode counterparts, low FPN can be achieved by applying correlated double sampling (CDS) and gain correction techniques. This work presents a 32×32 gray-level imaging integrated circuit featuring focal plane image compression, such that for each 4×4 pixel block, analog circuits implement differential pulse-code modulation, linear transform, and vector quantization. Other processing functions implemented in the chip are CDS and A/D conversion. Theoretical details are described, as well as the test setup of the chip fabricated in a 0.35 μm CMOS process. To validate the proposed technique, experimental results and captured photographs are shown. The CMOS imager compresses captured images at 0.94 bits/pixel for an overall power consumption below 40 mW (white image), which is equivalent to approximately 36 μW per pixel. Using photographs taken from bar-target pattern inputs, it is shown that details up to 2 cycles/cm are preserved in the decoded images.

Index Terms: cmos image sensor, compression, dpcm, focal plane, image processing, vector quantization.

I. INTRODUCTION

The main feature of CMOS image sensors is that they allow for the introduction of processing hardware at the pixel level. Using this characteristic, it is possible to extract from the image only the desired data, thereby reducing the output data rate. The analog processing also has the advantage of enabling high-speed operation. Owing to these features, in the last few years there has been a strong interest in CMOS image sensors, called here as imagers [1]-[8].

The imager described in this paper is capable of compressing a gray scale image using analog hardware [9]. The compression is lossy and the method used, which is based on DPCM (differential pulse-code modulation) and VQ (vector quantization), is briefly reviewed in Section II. To decode the bits produced by the chip, we use the decoder described in Section III. The pseudo-code necessary to generate the image is also shown in this section. Other experimental results from the chip can be found in [10], [11].

The main purpose of this paper is to report theoretical and experimental results that show the contributions of the DPCM and VQ analog circuits in producing the pictures taken by the chip. The pseudo-code in Section III describes the theoretical contributions and shows how the DPCM and VQ results are put together to create the final image. Section IV briefly describes the circuits that are used for the implementation of the DPCM algorithm. Experimental results are presented in Section V, and concluding remarks are made in Section VI.

II. IMAGE COMPRESSION ALGORITHM

A description of the image compression algorithm is shown next. Additional details can be found in [9]-[11]. To illustrate this description, Fig. 1 displays

an overall block diagram showing the interconnections of the three stages that compose the image compression algorithm.

A. Initial Setup

The imaging integrated circuit developed in this work produces a 32 × 32 pixel image that is divided into 4 × 4 pixel blocks. The 16 scalar values from each block are arranged into a 16 × 1 vector denoted as $\mathbf{y}(n)$. The sum of all the values is computed from $\mathbf{y}(n)$, thus generating the sum value $s(n)$. The index n varies from 1 to 64 and represents the pixel-block index.

B. Sum Value Prediction

Since $s(n)$ and $s(n+1)$ are correlated, $s(n)$ is used to compute $\hat{s}(n+1)$, which is applied as a prediction value for $s(n+1)$, that is,

$$\hat{s}(n+1) = \hat{s}(n) + \hat{e}(n), \tag{1}$$

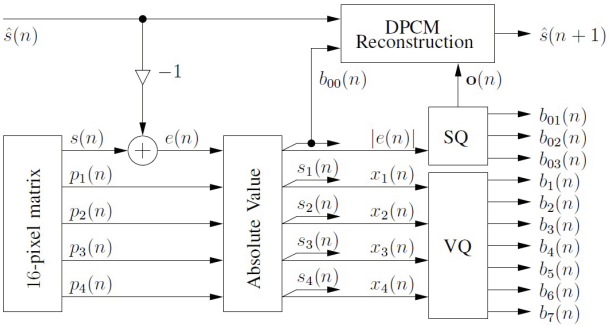

Figure 1. Image compression algorithm block diagram for a 4 × 4 pixel block.

where $\hat{s}(n)$ from the 4 × 4 pixel block will be equal to the quantized value of $s(n)$, and $\hat{e}(n)$ is the four-bit quantized representation of the prediction error

$$e(n) = \hat{s}(n) - s(n). \tag{2}$$

In this operation, $\hat{s}(n)$ from the first 4 × 4 pixel block is equal to the quantized value of $s(n)$.

C. DPCM Bits

The prediction error $e(n)$, which is given by the difference between $s(n)$ and $\hat{s}(n)$, is represented by a four-bit binary word. The sign of $e(n)$ is the first bit from the DPCM output vector. By quantizing $|e(n)|$ using three bits, we obtain the four bits that represent the sum of all the values from the pixel block:

$$e(n) = \sum_{i=1}^{16} y_i - \hat{s}(n), \tag{3}$$

$$\mathbf{b}_0(n) = \begin{bmatrix} \text{sign}(e(n)) \\ \text{SQ}(|e(n)|) \end{bmatrix}, \tag{4}$$

where sign(·) is the sign extraction operator and SQ($|e(n)|$) denotes 3-bit scalar quantization of $|e(n)|$.

D. Linear Transform

The components of vector $\mathbf{y}(n)$ are copied through current mirrors, so that the vector can be multiplied by a 4 × 16 linear transform matrix denoted as $\mathbf{H}$ [11]. The rows of $\mathbf{H}$ were chosen so that the result of the transform is a 4 × 1 vector with four principal components of the linear transform. We denote this principal component vector as $\mathbf{p}(n)$. The distribution of each of its components is Laplacian (i.e. a two-sided exponential distribution). The absolute value of each of its components is thus computed, and their respective signs are represented using four bits.

E. Vector Quantizer

The vector $\mathbf{x}(n)$, which contains the absolute values of the four components, is applied to a vector quantizer. In order to implement the vector quantizer using analog circuits with low complexity [12], we apply an approximation of the ideal quantizer. To accomplish that, $\mathbf{x}(n)$ is linearly transformed according to

$$\mathbf{f}(n) = \mathbf{W}\mathbf{x}(n). \tag{5}$$

The 4 × 4 matrix $\mathbf{W}$ is the optimal transform computed by principal component analysis out of the image database. The four elements of $\mathbf{f}(n)$ are quantized using one scalar quantizer for each element. The first element is represented with 3 bits, the second one with 2 bits, the third one with 1 bit and the fourth one with one bit, thus yielding the 7 bits obtained from the VQ output. The thresholds for the scalar quantizers were found using the same set of test images and a non-linear optimization routine [12].

F. Final Reshape

The 15 bits from each block are loaded into a shift register. This register sends, using a serial output, all the 15 bits to a micro-controller that in turn sends them to a digital computer that decodes the bits.

III. THE DECODER

To generate images from the output bits provided by the chip, we developed a MATLAB implementation of the decoder that is described in this section. For both the VQ and the DPCM, there are dictionaries that provide a 4 × 4 pattern, in the case of the VQ, or a scalar value, in the case of the DPCM, for a given word of bits.

The DPCM is dependent not only on the bits from a block of 4 × 4 pixels, but also on the values of the previous block. To find the current block decoded value, we use the four bits that came from the chip and search the corresponding reconstructed value in a dictionary, obtaining an estimate of the DPCM error, i.e. $\hat{e}(n)$. The error is then added to $\hat{s}(n-1)$, which is the estimate of the pixel sum in the $(n-1)$-th block. This estimate has been generated by the previous, $(n-1)$-th, block. As a result, we can compute the estimated average value for the current block out of $\hat{s}(n-1)$. In a loop with n increasing from 1 to 64, the decoder implements

$$\hat{s}(n) = \hat{s}(n-1) + \mathrm{SQ}^{-1}(\mathbf{b}_0(n)), \quad (6)$$

where $\hat{s}(n-1) = 0$ if n is a multiple of 8 (at the beginning of a block row), and the inverse scalar quantization operation is implemented by using the four bits of $\mathbf{b}_0(n)$ (one sign bit and three absolute value bits) to select the appropriate reconstruction value from a scalar codebook defined by

$$\mathbf{c}_0 = 10^{-3} \times [\,6.25, 25, 56.25, 100, 150, 225, 325, 468.7\,]. \quad (7)$$

After the reconstruction of all average values in all 4 × 4 pixel blocks, we obtain an image that describes the DPCM response for a given target. In other words, the DPCM decoded image is very coarse, and it only contains the estimated mean values of all blocks of the target. For example, some experimental DPCM results can be seen on the leftmost column of Fig. 6.

Although the VQ encoder is implemented as a linear transform followed by scalar quantization, the inverse operation minimizing reconstruction mean-squared error is not the inverse of the scalar quantization followed by the inverse of the linear transformation. Since the vector coding procedure is VQ indeed, to perform the inverse operation we must use the seven VQ bits to directly look up $\hat{\mathbf{x}}(n)$ in the VQ dictionary, which appears in Fig. 2.

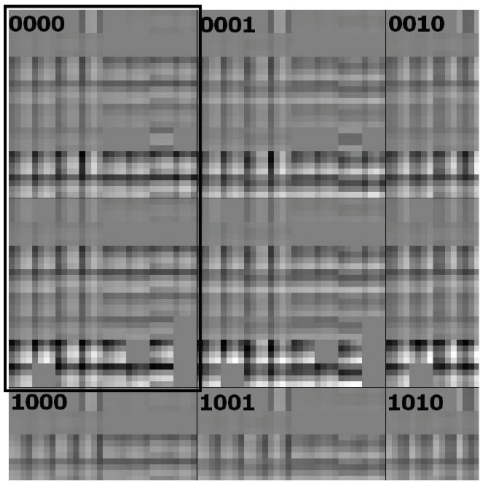

Figure 2. VQ codebook (inside the black box).

In a loop with n running from 1 to 64, the VQ decoder uses the seven bits in $\mathbf{b}(n)$ to compute

$$k(n) = [\,64\,;\,32\,;\,16\,;\,8\,;\,4\,;\,2\,;\,1\,]\mathbf{b}(n), \quad (8)$$

and then $\hat{\mathbf{x}}(n) = \mathbf{c}_k(n)$, where $\mathbf{c}_k(n)$ is the k-th column, selected for block n, out of the matrix $\mathbf{C}$, which represents the VQ codebook. The bits $s_1(n)$ to $s_4(n)$ from the linear transform vector signal are then used to define the pattern phase: for example, a black-to-white transition is positive, whereas a white-to-black transition is negative. This results in an estimated 4 × 1 principal component vector, denoted as $\hat{\mathbf{p}}(n)$. In a loop with n running from 1 to 64, the decoder computes the reconstructed value of $\mathbf{p}(n)$, which is denoted by $\hat{\mathbf{p}}(n)$, according to

$$\hat{p}_m(n) = (2s_m(n) - 1)\hat{x}_m(n), \quad (9)$$

for $m = 1, \ldots, 4$, where $s_m(n)$ denotes the m-th component of 4 × 1 sign vector $\mathbf{s}(n)$.

To complete the image decoding operation, we need to invert the linear transform $\mathbf{H}$ in order to transform the 4 × 1 vector $\hat{\mathbf{p}}(n)$ back into the 16 × 1 vector that represents the 16 values of the 4 × 4 pixel block. This is accomplished by applying $\mathbf{H}^{\mathrm{T}}$ to $\hat{\mathbf{p}}(n)$. Rearranging the vector back into the 4 × 4 pixel block, we reconstruct the details of that pixel block. Similarly to what was carried out with the DPCM procedure, after the reconstruction of all the 4 × 4 pixel blocks we obtain an image that incorporates the VQ operation for the given target. Thus the reconstructed pixel block, still in 16 × 1 format, which is denoted by $\hat{\mathbf{y}}(n)$, is obtained at the decoder side by means of a loop with n from 1 to 64 in which the following operation is executed:

$$\hat{\mathbf{y}}(n) = \mathbf{H}^{\mathrm{T}}\hat{\mathbf{p}}(n) + \hat{s}(n)[1, 1, \ldots, 1]^{\mathrm{T}}. \quad (10)$$

The resulting vectors have all the details properly estimated from the target image.

The final image is obtained with the sum of the DPCM and the VQ images. Finally, a block-wise operation reshapes all 16 × 1 vectors into 4 × 4 blocks, and the blocks are organized into an 8 × 8 matrix with 64 blocks, so that the final resolution is 32 × 32.

IV. CIRCUIT DESIGN AND FABRICATION

Figure 3 shows the schematic diagram of the DPCM reconstruction circuit. The M_A transistors have width equal to 1 μm and length equal to 2 μm. These are also the dimensions of M_2 to M_4, and M_7 to M_{11}. The M_B, M_5, and M_6 transistors have minimum size (width equal to 1 μm and length equal to 0.35 μm). The width and length of M_1 are equal to 2 μm.

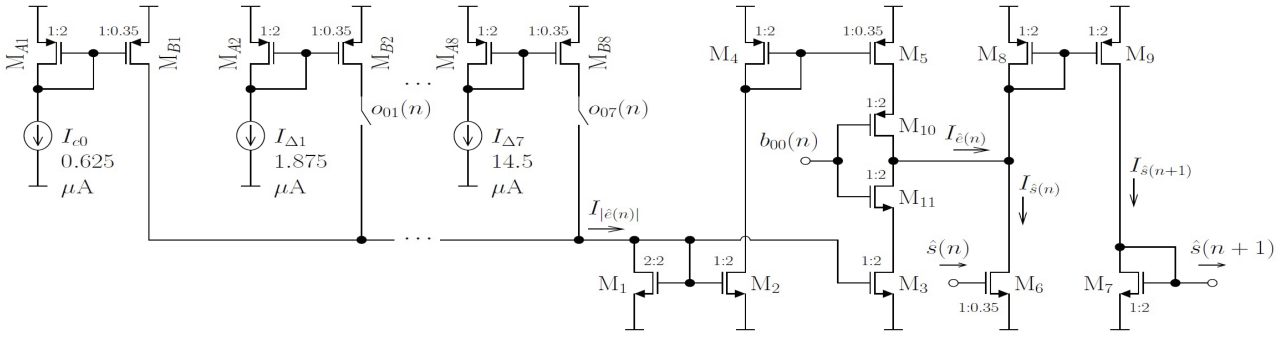

Figure 3. DPCM reconstruction circuit schematic diagram.

This circuit has been designed so that the comparator output bits [9], [10], denoted as $o_{01}(n)$, …, $o_{07}(n)$, are a quantized description of the absolute value the prediction error, i.e. $|e(n)|$. This description indicates how many quantization thresholds have been exceeded by $|e(n)|$, and hence defines which of the numbers in $\mathbf{c}_0$ (Section III) must be reconstructed in current mode for the computation of the next $\hat{s}(n)$. The I_{c0} current represents the first value in $\mathbf{c}_0$. The difference between the second and the first $\mathbf{c}_0$ values is represented by the current $I_{\Delta1}$. The difference between the third and second values is represented by $I_{\Delta2}$, and so forth. The summation of the I_Δ currents thus allows the current-mode reconstruction of any of the values in $\mathbf{c}_0$. After the quantized value of $|e(n)|$ is computed, which is represented by $I_{|\hat{e}(n)|}$ in Fig. 3, this value is either added to or subtracted from $I_{|\hat{s}(n)|}$. The selection between addition or subtraction is controlled by $b_{00}(n)$. The resulting current, $I_{|\hat{s}(n+1)|}$, is copied into the next pixel block, where the same set of DPCM operations take place. The linear transform, absolute value, and VQ operations have been designed and implemented with similar hardware that has been described in [9]-[11].

Table I presents some characteristics of the fabricated chip. Figure 4 shows the layout of a 4 × 4 pixel block and Fig. 5 shows a photograph of the fabricated chip containing a pixel array with 32 × 32 pixels or, equivalently, eight pixel blocks along each direction. Besides these 64 pixel blocks, we included current reference sources and some test structures (individual photodiodes with and without CDS controls, and a single pixel block) in this layout.

Table I. Chip Characteristics.

Technology	AMS CMOS 0.35 μm
Chip Area	1.61 mm × 1.28 mm
Array Size	32 × 32
Block Area	150 μm × 150 μm
Pixel Area	37.5 μm × 37.5 μm
Photodiode Area	10 μm × 10 μm
Fill Factor	7%
Power Supply	3.3 V
Power Consumption	37 mW maximum (white image)

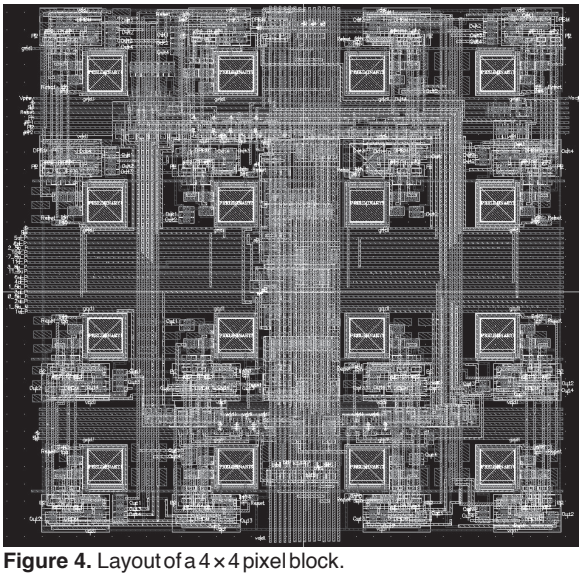

Figure 4. Layout of a 4 × 4 pixel block.

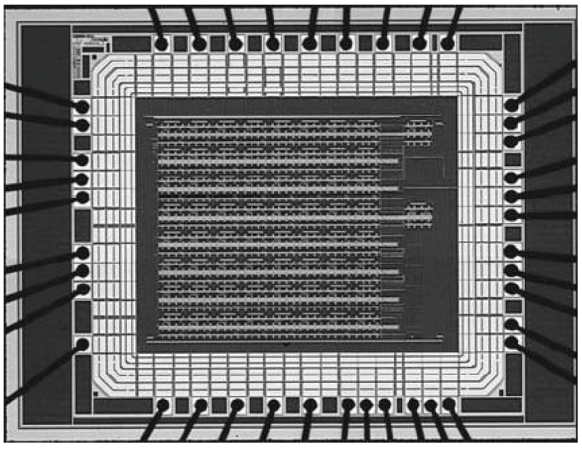

Figure 5. Chip photograph captured through an optical microscope.

V. Experimental Results

We designed a test board to evaluate the fabricated imager. To implement an interface between the imager and a digital computer, we use a PIC 18LF4550 micro-controller from Microchip. It has USB communication and operates on a 3.3 V power supply.

Every time an image is requested, the micro-controller executes three tasks: generating control

signals, recording the imager output bits, and sending them to a computer running the image decoding software. A lens was fixed on a suitable mounting part immediately above the imager socket, so that targets with size around 4 cm × 4 cm can be conveniently placed 10 to 20 cm away from the optical system. The integration period between the photodiode reset and pixel readout was adjusted to 800 μs. To measure power consumption, we use a small resistor (10.7 Ω) connected between the power supply and the VDD pin of the imager chip.

Figure 6 displays a set of pictures taken by the integrated circuit. In this figure we can see column-wise, from left to right: DPCM result, VQ result, and overall snapshot (instant image capture) combining the DPCM and VQ results, and an average image computed out of 500 snapshots for the suppression of temporal noise. Temporal noise can be easily observed in individual snapshots. It corresponds to errors that appear occasionally in regions with photodiode groups that should clearly have shown a different photocurrent reading. Although these pixel errors are frequent, they appear randomly at different locations. They can therefore be suppressed by computation of an average over several images. The error reduction can be illustrated for instance by the second line of Fig. 6, where the snapshot sample has a few saturated pixels that appear correctly in the average picture.

This type of error also appears clearly in the third, fourth, seventh, eighth and ninth rows of Fig. 6. Particularly when the light excess causes the pixel read-out currents represeting the $y_l(n)$ pixels (l = 1, 2, …, 16) to be too large, the DPCM circuit fails in correctly tracking the pixel sum. The failure usually takes place when the current summation stays, for several pixel blocks, above the highest current summation value for which the adder and DPCM reconstruction circuits (Fig. 1) were designed to operate. In this situation, we observe that the DPCM state $\hat{s}(n)$ is not able to return to lower values after, spatially, the current sum returns to lower values in the next pixel blocks. In order to avoid DPCM errors, the integration time, environment illuminance, and lens aperture were adjusted so that pixel saturation was not frequently observed. However, a few DPCM errors are still observed, which is due to temporal noise.

In our tests, the environment illuminance is low and the lens aperture is small, so that the raw image appearance is rather dark, as shown in the first column of Fig. 6. To obtain lighter images, we shift the mean value of the decoded image down to 0.0, then apply a hyperbolic tangent function $0.5\tanh(x)$ to the pixel values, and then shift the mean value back up to 0.5. This yields images with increased contrast. The pixels in our imager were acquired with CDS but, for further spatial noise suppression, we also apply a low-pass spatial filter. This leads to decoded images featuring softer object boundaries.

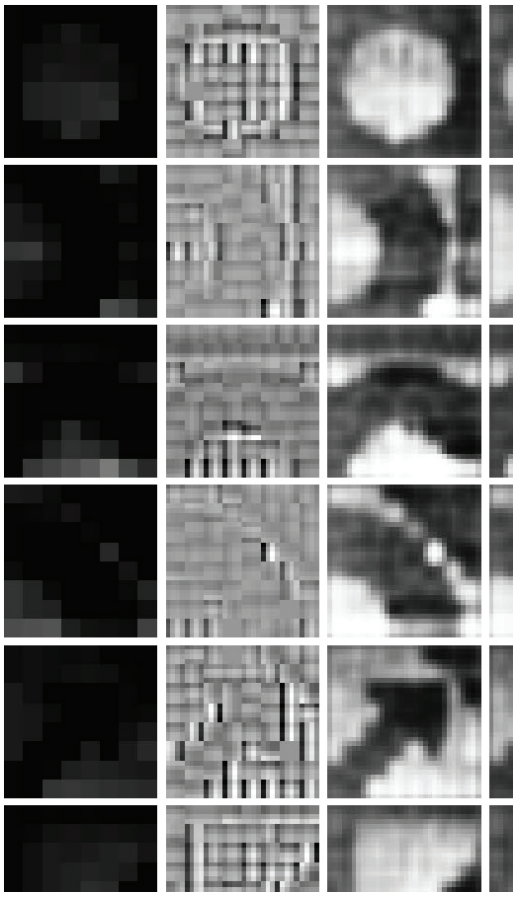

Figure 6. Some experimentally reconstructed images. Column-wise, from left to right: DPCM result, VQ result, overall snapshot, and an average image output computed out of several snapshots for temporal noise reduction.

The first row of Fig. 6 corresponds to a circular white target. As expected, the DPCM and VQ stages complete each other in generating the final image. Textures and details on object boundaries and transitions where pixels are darker on the outside of the circle – and brighter on the inside – are provided by the VQ. This can be seen in the second column, for all examples in this figure. The pictures in the second, third, and fourth rows of this figure were taken using a black ring target seen from different orientations. The fifth row shows a black arrow target. The overall imaging system, even with the significant amount of data compression, still has enough resolution to represent the thin line that exists between the end of the arrow and the beginning of the black triangle in the top right corner of the fifth row. The sixth and seventh rows show a white triangle and a black one. Rows 2 to 7 illustrate the VQ response, i.e. the use of the VQ dictionary, for the representation of lines at different orientations that are present in the same image.

A. Photographs Captured from Natural Images

We also used, as a target, a 32 × 32-pixel tile cropped from the Lena image, which shows her eye, eyebrow, and part of her hat (Fig. 7). This image was

also employed in Cadence electrical simulations that were shown in [9]. Figure 8 shows two results obtained with the Lena target. Averages were computed over 500 images. The first column of Fig. 8 presents the DPCM average result, the second one shows the VQ average result and the last one shows the average image taking into account both DPCM and VQ. In the first row, we used a small lens aperture to capture the images. In the second row the aperture is larger, but it was carefully adjusted to avoid pixel saturation. One can notice that the optical setup influences the final imaging result. Additional details such as the eyebrow can be enhanced (in the top image) even though the amount of captured light is smaller.

Figure 7. Tile with 32 × 32 pixels taken from the Lena image, and used as a target in the experiments.

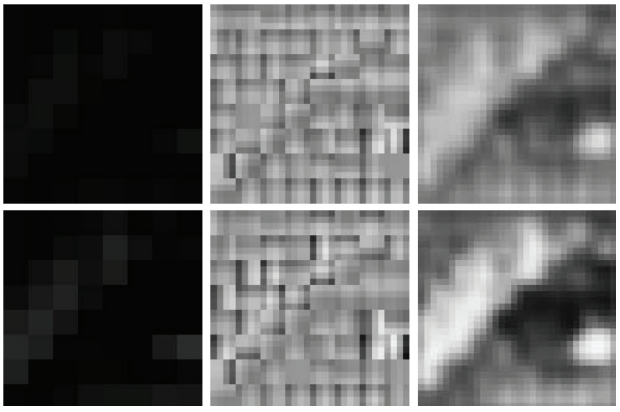

Figure 8. Experimental results obtained with 32 × 32 tile from the Lena image, under two different illumination conditions: smaller aperture (top) and larger aperture (bottom). From left to right, column-wise: DPCM, VQ, and overall results.

B. Modulation Transfer Function (MTF) Measurements

In this section, we focus on the overall modulation transfer function (MTF) [13] of the proposed imaging system, as it operates at bit rates below 0.94 bpp.

The MTF describes the spatial magnitude response of an overall imaging system, based on the hypothesis that input images corresponding to sinusoidal functions with different spatial frequencies are applied to the imaging system. In this section, we provide results of experimental tests that were performed with bar-target pattern inputs featuring black and white stripes with the same width. It can be shown [13] that, except for a scale factor with respect to the case in which the input is a continuous sinusoid, the bar-target method provides the correct MTF estimates. Our goal is to estimate the highest spatial frequency that can be detected by the proposed imager. This measurement is related to the narrowest detail that can be decoded from the imager output, which is represented by the point-spread function (PSF).

The MTF of a conventional imager is described by a single scalar function that is isotropically replicated along every direction. However, in the case of the proposed imager, a different behavior is expected along the horizontal direction and the vertical direction, because from one column to the next column we expect slope overload and other predictive quantization artifacts that are created by the DPCM stage. Since each block-row (i.e. a set of four pixel-rows) is, in terms of DPCM, encoded independently of the previous or next four rows, we do not expect predictive quantization artifacts from one row to the next row. And since the imager, lens and object are placed so that each pixel (37.5 μm × 37 μm in silicon) is illuminated by a 1 mm × 1 mm pixel from the target, these statements are particularly true for bar-target inputs with period above 8 mm (i.e. 1.25 cycles/cm), because that frequency corresponds to the peaky MTF situation in which a 4 × 4 pixel block is entirely white, the next block is black, and so on. For frequencies above 1.25 cycles/cm, the falling MTFs along every direction will tend to converge until most of the details are significantly lost at sufficiently high frequency. This behavior is expected to be particularly similar for the vertical and horizontal directions, because of the codebook symmetry imposed by the separate coding of the transform coefficient signs. Since high-frequency components were eliminated after the application of the linear transform that was mentioned in Section II.D, we expect the maximum frequency detectable without aliasing by the imager to be below 2.5 cycles/cm.

Along the diagonal direction, we expect to find a mixture of the properties seen along the vertical and horizontal directions. So all tests were performed for three directions: vertical, horizontal, and the 45-degree diagonal appearing in Fig. 9.

For a given bar-target input, the MTF at a given spatial frequency is computed according to the following procedure:

i) compute the pixel average along the "DC" direction, i.e. the constant-pixel-value direction which is orthogonal to the direction along which the square wave is seen. The vector containing the 32 numbers representing the pixel averages is denoted as $\mathbf{a}(f)$;

ii) the MTF at frequency f is defined as

$$M(f) = \frac{\max(\mathbf{a}) + \min(\mathbf{a})}{\max(\mathbf{a}) - \min(\mathbf{a})}; \quad (8)$$

iii) to suppress undesired variation that is due to temporal noise and possible changes in illumination, we average the $M(f)$ values over 500 pictures from the same target under constant illumination conditions.

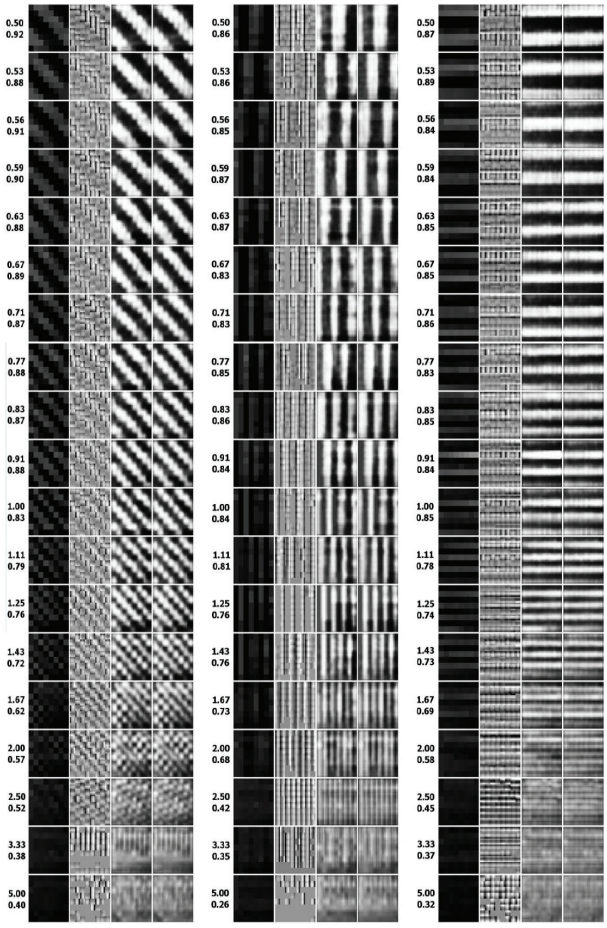

Figure 9. Experimental data available for the evaluation of the imager MTF.

Experimental MTF results are shown in Fig. 9. The data are organized into columns (diagonal, vertical and horizontal bar-target inputs). The DPCM stage output appears in the leftmost sub-column inside each column, the VQ output appears in the second sub-column, and the overall imager output appears in the third sub-column. To obtain the overall output, we add the contributions from the DPCM and VQ stages as described before. Correlated double-sampling has been applied. However, for additional suppression of fixed-pattern noise, the overall output has been lowpass filtered by a 3 × 3 matrix with entries equal to 1/9. To improve contrast, a sigmoidal non-linear function centered at 0.5 and with gain equal to 10 has been applied to every pixel in the filtered image. An average output image was computed over 500 pictures, as mentioned before, and the results appear in the fourth sub-column. Immediately to the left of each column, a pair of numbers indicates the spatial frequency in cycles/cm (top number) and the MTF at that frequency (bottom number). When the object is placed 20 cm away from the imager, each imager pixel corresponds to a 1 mm × 1 mm pixel at the object. We note that details can still be clearly perceived at 2.0 cycles/cm. Most details are preserved along the vertical and horizontal directions at 2.5 cycles/cm, and the imager does not respond to frequencies above 3 cycles/cm, which was expected because of the removal of high-frequency components after the linear transform.

Performance comparisons with other CMOS imagers that implement focal plane compression can be found in Ref. [11], where relevant quantitative metrics are considered.

VI. CONCLUSIONS

We presented a 32×32 imaging integrated circuit that captures and compresses gray scale images at the focal plane level. We adapted DPCM, linear transform, and VQ techniques in order to obtain low-complexity solutions that are suitable for analog hardware implementation. Theoretical details were presented for the encoder and decoder, as well as experimental results. The experimental results validate the proposed integrated circuit and show that focal-plane image compression was accomplished. For each block, the image compression algorithm results in a binary output vector with 15 bits. The data rate is thus 15/16 = 0.94 bits/pixel at most. If an entropy coding algorithm is further applied, lower bit rates can be obtained.

ACKNOWLEDGEMENTS

This work was supported by Brazilian research funding agencies: CNPq, CAPES, FAPERJ and FUJB/UFRJ. The authors also thank Gustavo Haubrich and Luis Henrique Haubrich, currently with Satisloh do Brasil Ltda. For having built the optical setup.

REFERENCES

[1] E. Artyomov and O. Yadid-Pecht, "Adaptive multiple-resolution CMOS active pixel sensor," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 53, no. 10, pp. 2178-2186, October 2006.

[2] W. D. León-Salas, S. Balkir, N. Schemm, and M. W. Hoffman, "A CMOS imager with focal plane compression using predictive coding," *IEEE J. Solid-State Circuits*, vol. 42, no. 11, pp. 2555-2572, Nov. 2007.

[3] Z. Lin, M. H. Hoffman, N. Schemm, W. D. León-Salas, and S. Balkir, "A CMOS image sensor for multi-level focal plane image decomposition," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 55, no. 9, pp. 2561–2572, October 2008.

[4] Y. M. Chi, A. Abbas, S. Chakrabartty, and G. Cauwenberghs, "An active pixel CMOS separable transform image sensor," in *Proc. IEEE Int. Symp. Circuits and Systems*, Taipei, Taiwan, May 2009, pp. 1281–1284.

[5] A. Nilchi, J. Aziz, and R. Genov, "CMOS image compression sensor with algorithmically-multiplying ADCs," in *Proc. IEEE Int. Symp. Circuits and Systems*, Taipei, Taiwan, May 2009, pp. 1497–1500.

[6] M. Zhang and A. Bermak, "Architecture of a digital pixel sensor array using 1-bit Hilbert predictive coding," in *Proc. IEEE Int. Symp. Circuits and Systems*, Taipei, Taiwan, May 2009, pp. 1501–1504.

[7] L. Camunas-Mesa, C. Zamarreno-Ramos, A. Linares-Barranco, A. J. Acosta-Jiménez, T. Serrano-Gotarredona, and B. Linares-Barranco, "An event-driven multi-kernel convolution processor module for event-driven vision sensors," *IEEE J. Solid-State Circuits*, vol. 47, no. 2, pp. 504–517, April 2012.

[8] G. Wan, X. Li, G. Agranov, M. Levoy, and M. Horowitz, "CMOS image sensors with multi-bucket pixels for computational photography," *IEEE J. Solid-State Circuits*, v. 47, no. 4, pp. 1031–2042, April 2012.

[9] F. D. V. R. Oliveira, H. L. Haas, J. G. R. C. Gomes, and A. Petraglia, "A circuit for focal-plane image compression using vector quantization," in *Proc. IEEE Int. Symp. Signals, Circuits and Systems*, Iasi, Romania, June 2011, pp. 181–184.

[10] F. D. V. R. Oliveira, H. L. Haas, J. G. R. C. Gomes, and A. Petraglia, "A CMOS imaging system featuring focal-plane compression based on DPCM and VQ," in *Proc. IEEE Latin American Symp. Circuits and Systems*, Playa del Carmen, Mexico, February 2012.

[11] F. D. V. R. Oliveira, H. L. Haas, J. G. R. C. Gomes, and A. Petraglia, "CMOS imager with focal-plane analog image compression combining DPCM and VQ," *IEEE Trans. Circuits and Systems I: Regular Papers*, v. 60, no. 5, pp. 1331-1344, May 2013.

[12] J. G. R. C. Gomes and S. K. Mitra, "A comparative study of the complexities of neural network based focal-plane image compression schemes," *IEICE Trans. Fundamentals of Electronics, Communications, and Computer Sciences*, Invited Paper, vol. J88–A, no. 11, pp. 1185–1196, Nov. 2005.

[13] G. D. Boreman, *Modulation Transfer Function in Optical and Electro-Optical Systems*. Bellingham, WA: SPIE Press, 2001.