Concurrent focal-plane generation of compressed samples from time-encoded pixel values

M. Trevisi[(1)] , H. C. Bandala[(2)], J. Fernández-Berni[(1)], R. Carmona-Galán[(1)], Á. Rodríguez-Vázquez [(1)]
[(1)] Instituto de Microelectrónica de Sevilla (IMSE-CNM), CSIC-Universidad de Sevilla, Spain
[(2)] Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Puebla, Mexico
trevisi@imse-cnm.csic.es

***Abstract*— Compressive sampling allows wrapping the relevant content of an image in a reduced set of data. It exploits the sparsity of natural images. This principle can be employed to deliver images over a network under a restricted data rate and still receive enough meaningful information. An efficient implementation of this principle lies in the generation of the compressed samples right at the imager. Otherwise, i. e. digitizing the complete image and then composing the compressed samples in the digital plane, the required memory and processing resources can seriously compromise the budget of an autonomous camera node. In this paper we present the design of a pixel architecture that encodes light intensity into time, followed by a global strategy to pseudo-randomly combine pixel values and generate, on-chip and on-line, the compressed samples.**

Keywords*— compressive sampling; cellular automaton; time-encoded pixels;

I. INTRODUCTION

Sparsity in natural images can be exploited to compress the signal and then recover the content from a number of samples that is below the limit determined by Nyquist's theorem [1] [2]. In order to implement this compressing mechanism, a pair of matrices is applied to the coefficients of the original signal, image pixel values in our case. These matrices are the compressive strategy, Φ, and the sparsifying dictionary, Ψ. Each compressed sample is a linear combination of the elements of the original image. Then, from a set of compressed samples much smaller than the original image, convex optimization can lead to a unique solution. In order to achieve this, the product of Φ and Ψ must hold the restricted isometry property (RIP) [3].

Building a compressive strategy on-chip is not trivial. The most common approach is to obtain the elements of Φ from a random distribution. It has been implemented by using optical elements [4] and also by employing dedicated circuitry [5]. The simplest implementation consists in using a sub-Gaussian distribution, i. e. the elements of Φ are obtained from a normal distribution that is thresholded so they are either one or zero. In this way each compressed sample results directly from the addition of all the pixels selected by Φ. In practice, there are two major limitations to implement these techniques at sensor level. First of all, the compressive strategy must be known at both extremes of the communication channel. This is, not only the sensor must know which the compressive strategy is in order to generate the corresponding compressed samples, but it must be present also at the end of the channel, in order to reconstruct the original image. Therefore, either the compressive strategy is generated at the sensor and transmitted to the image reconstruction system or it needs to be stored at both ends. This represents an important burden as these options will introduce strong requirements on the transmission bandwidth and/or the in-sensor memory. The second question is dynamic range. The number of bits required to describe the linear combination of N pixels is $\log_2 N$ in addition to the bits required to describe one single pixel. This is a major limitation for an implementation of this addition in the analog plane, as dynamic range in analog circuits is limited by the available output range and noise level.

These two limitations can be alleviated with a widely extended approach: block-based compressive sampling [6] [7] [8]. The pixels of the complete image are divided into macro-blocks to which the compressive strategy is applied. This represents a considerable reduction of Φ and the infrastructure required for its generation, transmission and/or storage. At the same time, the required dynamic range to represent the compressed samples is noticeably reduced. In exchange for this, reconstruction departs from ideal and may require additional samples to achieve a prescribed accuracy.

Our approach is the on-chip generation of a full-frame compressive strategy [9] by means of a 1D cellular automaton (CA) for the pseudo-random selection of pixels [10]. This solution avoids both transmitting and storing the compressive strategy as it can be error-free reconstructed from the initial seed with an analogous CA implementation. In addition to this, pixel values need to be represented in a form that does not exhaust the available dynamic range when being added up. In order to do this, we are encoding pixel values in time. Addition is realized in the digital domain by asynchronously tagging events in a per-column basis. In this paper we are displaying the details of the design of the chip, starting with the individual pixel, and following with the pseudo-random selection of pixels, the time to digital conversion of the pixel value, the column addition and the final generation of the compressed sample.

II. PIXEL ARCHITECTURE

The output of a compressive sampling imager is given in the form of linear combinations of randomly selected pixel

contributions. These linear combinations are the compressed samples [1]. As we have already referred, the generation of the random pattern for the selection of pixels has been left to a one-dimensional cellular automaton that will be described later on. As for now, just consider that a set of devices delivering random logic zeroes and ones are employed to generate selection signals for the rows and columns of the pixel array.

Apart from the generation of random selection signals, the major difficulty on implementing a full-frame compressive strategy is providing enough dynamic range to accommodate compressed samples. Consider that each pixel value, either originally a voltage or a current, is represented by N_b bits. For an image of $M \times N$ pixels that can be chosen at random to compose a compressed sample, the number of bits required to avoid any kind of clipping in the generation of compressed samples is:

$$N_B = N_b + \log_2(MN) \tag{1}$$

Furthermore, even in block-based compressive sampling [11], where images are divided into small blocks to reduce N_B, compressive samples are generated using blocks that have a minimum practical size of 8×8 pixels. Smaller blocks would not be very sparse and as such one of the fundamental premises behind compressive sampling would fall. To preserve resolution, if each pixel value is encoded in 8b, we would still need 14b of resolution in order to properly encode compressed samples. Our approach to provide the number of bits prescribed by Eq. (1) is to time-encode the pixel values and employ time-to-digital conversion. The summation of pixels is then realized in the digital domain, avoiding the requirement of a wide dynamic range in the analog domain.

A. *Time-encoding of light intensity*

The elementary pixel (Fig. 1) contains an integrating photodiode that discharges node V_{pix} at a rate determined by the photocurrent. This is depicted inside the 'Time-encoding of light intensity' box in Fig. 1. When V_{pix} crosses a reference voltage V_{ref}, a voltage comparator flips its output, V_1. It time-encodes the magnitude of the light intensity, what is described as pulse-modulation imaging [12]. The pixel value is then contained in the period of time separating the reset of node V_{pix} and the moment in which V_1 turns from low to high. The lower (higher) the light intensity on the diode is, the longer (shorter) it takes to the comparator to switch. In this chip, both V_{rst} and V_{ref} can be adjusted on-line in order to adapt to different illumination conditions in real-time.

B. *Pixel selection circuit*

As previously mentioned, the contribution of each pixel to a particular compressed sample is determined by a combination of row and column selection signals, S_i and S_j, that are generated with the help of a one-dimensional cellular automaton positioned around the sensor array (Fig. 2). These two signals are combined by a XOR gate, implemented by 6 transistors in Fig. 1. The voltage V_2 is stuck at V_{dd} if selection signals S_i and S_j are equal. If not, V_2 is the inverse logic value of V_1. Using a XOR gate guarantees that the pixel contributes to the linear combination that constitutes a compressed sample in just half of the possible combinations of S_i and S_j. It is important to notice that this pixel selection unit is allocated right after the comparator because this helps reducing power consumption. If a pixel is not contributing to the compressed sample there is no reason to let the pixel activation front propagate, inducing changes in the subsequent nodes that are going to be discarded later.

C. *Propagation of the activation edge*

Signal V_2 is active in low, eliciting a rising edge in V_3 if this signal has not been activated before. If it has, the feedback of $\bar{V}_3$ locks V_3 to logic '1' until the pixel is reset again. Now consider that signal Q' is high. Later we will what makes Q' changes over time. If Q' is in logic '1', then V_4 is the inverse of V_3, i. e. if the pixel is activated and is selected to contribute to the compressed sample, V_3 goes from logic '0' to '1' and V_4 goes from '1' to '0'. If signal C_{in} is low, this falling edge in V_4 induces a rising edge in V_5 which is the signal controlling driving transistor M_2. The column bus, whose voltage V_o is pulled up to V_{dd} by default, experiences a pull down driven by M_2. V_o will remain low if it was not for the event termination circuit.

D. *Event termination circuit*

The rising edge in V_5 is feedback to the event termination circuit, where it is inverted as long as Q, which is a global signal, is high. This causes Q' to fall to logic '0', switching

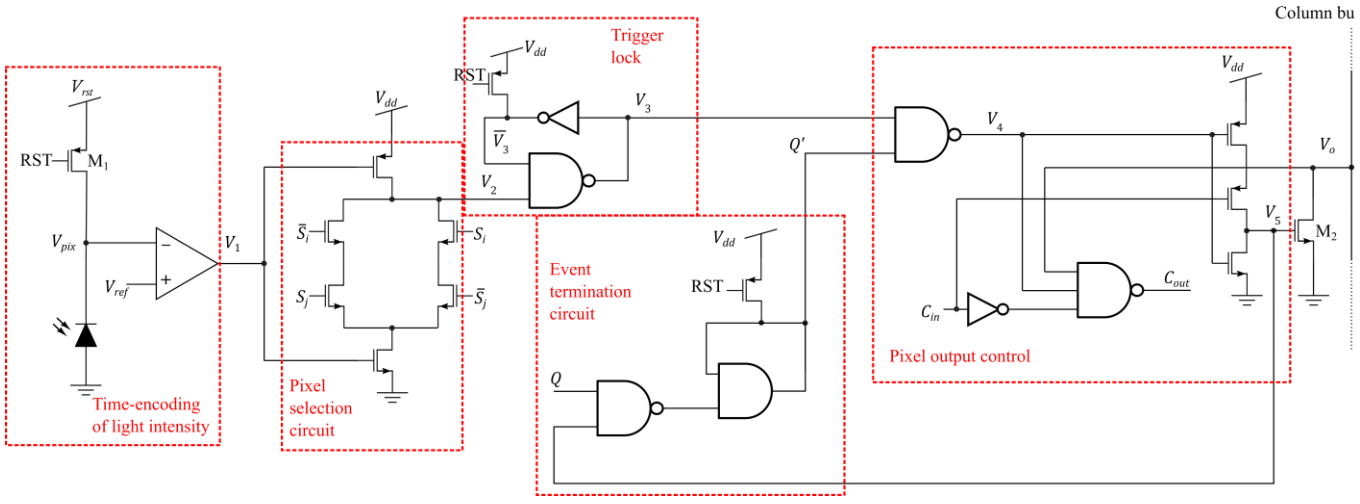

Fig. 1 Schematics of the elementary pixel

back V_4 to logic '1' and then V_5 to logic '0', terminating the pulse that started before after a short delay.

The motivation to use a global pulse termination signal to establish the duration of the events instead of a local delay unit is to provide global control without introducing area and/or power consuming elements in the pixel. In particular, Q is a signal provided by a control unit in each column of the pixel array. This unit senses the column bus and detects if it is being pulled down. Once the falling edge is detected, and after a user-controllable delay, Q rises enabling the termination of the pulse only in the pixel that has already turned M_2 on. This is verified by the NAND gate in the 'Event termination circuit' box (Fig. 1).

E. Pixel output control

As depicted in Fig. 1, all pixels in the same column of the array share the same column bus to transmit its output pulse. As will be explained later, the time-encoding of the pixel value will be converted to digital by means of a time-to-digital converter, which in this case will be built with a clock and a counter. Of course, there is no a priori knowledge on the proximity of the values of the pixels and, therefore, how close in time will be the pulses emitted by the pixels. What is clear is that each one of them needs to be taken into account if we do not want to introduce additional errors in the image reconstruction from its compressed samples. In order not to skip any of the pulses, a token protocol is established so pixels that are being triggered close in time are only allowed to emit their pulse one after the other. This blocking mechanism needs to be parallel to all pixels so that the first pixel that delivers its event puts all other pixels on hold until its event is over. The release mechanism on the contrary has to be sequential so that, if there is more than one pixel in queue waiting to deliver its pulse, it will be impossible to have more than one of them active at the same time. In order to do so, each pixel receives a signal C_{in} from the pixel immediately above (Fig. 1), and sends a signal C_{out} to the pixel immediately below it. If there is no preceding pixel waiting to deliver a pulse through the column bus, C_{in} will be low. This enables the propagation of a falling edge in V_4 when it occurs into a rising edge in V_5. If C_{in} is high, however, this propagation is retained.

One pixel's C_{in} corresponds to its upper neighbor C_{out}. In order to be '0', three different conditions must hold, namely: its C_{in} is low, what means that there is no pixel above it that wants to deliver a pulse; V_4 is high, what means that either the pixel has not been activated or it has already delivered a pulse; and V_o is high, what means that the column bus is available. If any of these three conditions is not true C_{out} will be stuck at the logic '1', thus preventing any of the pixels below it emitting a pulse through the column bus. A 3-input NAND gate is employed to combine the level at C_{in}, the pixel readiness to pull down the column bus and the feedback on the actual state of this column bus. This aggregated information is then sent as C_{out} to the pixels below. Using this logic each pixel will know that if $V_o = V_{dd}$ and no pixel above is waiting to pull it down it is allowed to release its own event. Since V_o is fed back to this control block, when a pull down occurs, each pixel will simultaneously block the pixel immediately below through C_{out}. The blocking mechanism is parallel. On the contrary, when an event is over, its C_{out} turns to '0', so the pixels will be released sequentially in a top down fashion.

III. Sensor Architecture

The pixel already described is part of an image sensor that implements a full-frame compressive strategy. The architecture of the chip is depicted in Fig. 2. The central element of the architecture is an array of 64×64 pixels. The peripheral circuit needs to implement the following functionalities: pseudo-random column and row selection, time-to-digital conversion of the pixel values, addition of the pixel values of the selected pixels. Let us describe the circuits implementing these functionalities one by one.

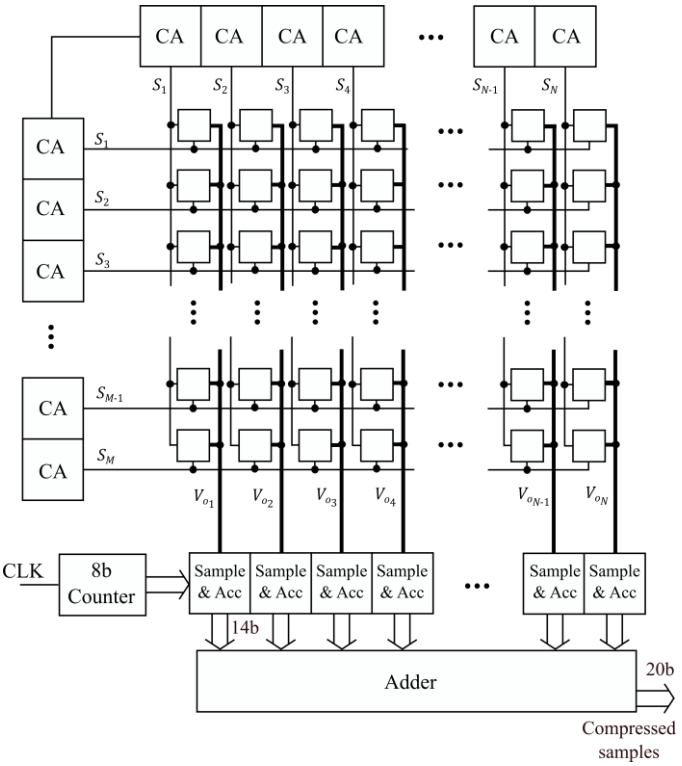

Fig. 2 Conceptual floorplan of the sensor chip

A. Pseudo-random generation of selection signals

The generation of pseudo-random patterns starting from a seed can be realized using different methods. Some of them have been employed before in the context of compressive sampling, like Hadamard vectors [13] or linear feedback shift registers [14]. As already mentioned, our approach consists in a 1-D cellular automaton, which has the advantage of being easily implemented in CMOS technology and its scalability, as the evolution of its cells only depends on their own state and those of their closest neighbors. Typically a linear CA with radius-1 interactions between its cells is defined by a truth table defined on the cell state (S) and the states of the left (L) and right neighbor (R). Table I shows the truth table for Rule 30,

which has been demonstrated to display aperiodic (class III) behavior [10]. The circuit employed to implement a cell of this cellular automaton is depicted in Fig. 3. The cell state is precisely the selection signal that is delivered to a rows or a column, depending on the position of the CA cell (Fig. 2).

Table I Truth table of Rule 30

L	S	R	NS
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	1
0	0	0	0

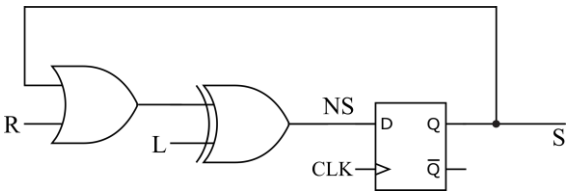

Fig. 3 Implementation of a Rule 30 cell of a cellular automaton

B. *Column-wise time-to-digital conversion*

Events generated at the pixels and transmitted through the column bus arrive to block 'Sample & Add' in (Fig. 2). These pulses encode the pixel value in the period of time that has passed between the pixels reset and the arrival to the 'Sample & Add'. A straightforward method to translate all this pulses into digital codes is to use the pulses to activate the sampling of a global time counter (Fig. 2) activated by a clock signal and started with the global pixel reset —allocating some initial delay to allow the pulses to reach the bottom of the array. Each time a pixel activation pulse arrives, the 8b of the counter are sampled and added to the already stored sum. After 256 clock periods, the pixel values have been accumulated at the 'Sample & Add', which delivers a 14b word containing this sum, as it is the result of adding up to 64 pixel values. After that, the 64 column sums are added up into a compressed sample of 20b.

Compressed samples need to be encoded in a much larger digital word, therefore there is an amount of compressed samples beyond which it is better to just deliver the uncompressed image. In our case, as pixel values are encoded by 8b and, and compressed samples in 20b, the compression ratio (R), i. e. the number of samples delivered divided by the total number of pixels in the image, needs to be below 0.4. This means that for a $M \times N$-pixel image, we will be always considering less than $0.4MN$ compressed samples.

In addition, as compressed samples are generated sequentially, it is necessary to operate the imager at a frame rate (f_{cs}) —considering that it is referred to the time it takes to deliver one single compressed sampling— that is at maximum $0.4MN$ times the original frame rate (f_s)

$$f_{cs} = R \cdot MN f_s \tag{2}$$

For $f_s = 30$fps, $R = 0.4$ and an image of 64×64 pixels, compressed samples can be generated at ≈ 50kHz at maximum. This is 20µs per compressed sample. It the duration of events is, for instance, 5ns, and the 64 pixels in a column are selected, there is a 6.25% chance that two events will randomly overlap. In order to avoid missing any pulses, we are delivering them one by one so that, if there is more than one pixel in queue waiting to deliver its event, it is impossible to have more than one of them active at the same time. As the time-to-digital conversion clock need to tick 256 times in the 20µs, it is possible that some pulses are detected in the following clock period, what will introduce a 1LSB error in the 20b compressed sample. Verification on the negligible influence of this error has been performed at system level.

IV. Chip Prototype

A prototype chip has been designed in a CMOS 0.18µm technology following the already described methodology. The die size including pads is 3.17×2.23 sq. mm (Fig. 4). It has 84 pads, of which one third is dedicated to power supply and ground connections. Table II contains a summary of the features of the prototype that is already in fabrication.

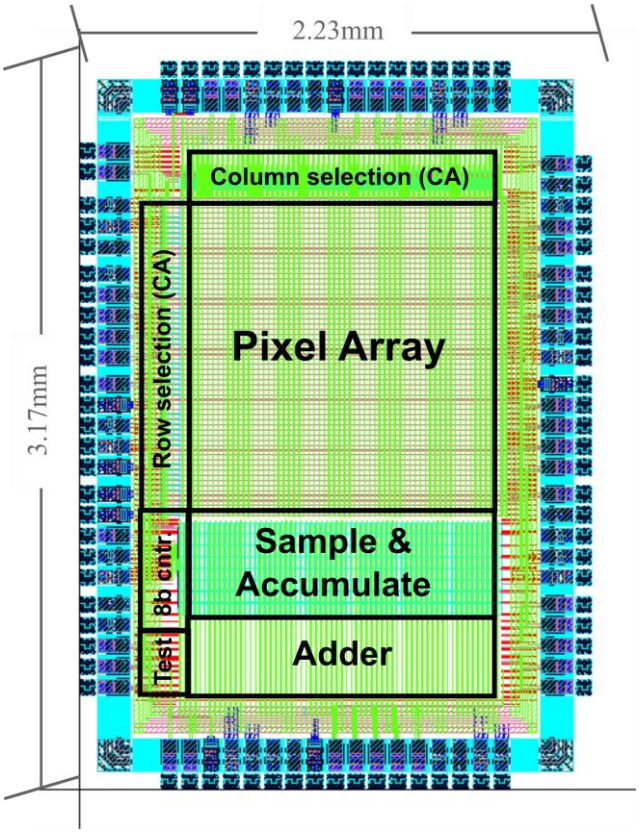

Fig. 4 Layout of the prototype sensor chip

Table II Summary of chip features

Technology	CMOS 0.18μm 1P6M
Die size (w. pads)	3174μm × 2227μm
Pixel size	22μm × 22μm
Fill factor	9.2%
Resolution	64 × 64
Photodiode type	n-well/p-substrate
Power supply	3.3V-1.8V
Predicted power consumption	<100mW
Frame rate	30fps
Max. compressed sample rate	50kHz
Clock Freq.	24MHz

The central part of the chip is the array of 64 × 64 pixels. Fig. 5 depicts the layout of the elementary pixel. The blocks described in Sect. II can be identified. In order to reduce the influence of the offset of the comparator, an auto-zeroing scheme has been implemented using a MiM capacitor on the top metal layers (not showing in the picture). Formal verification of the chip performance has been realized with post-layout simulation.

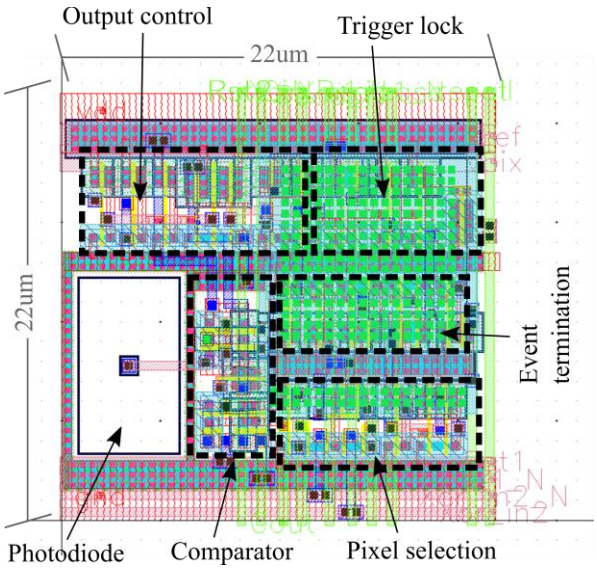

Fig. 5 Layout of the elementary pixel

V. Conclusions

The design of compressive sampling image sensor prototype based on the on-chip generation of a full-frame compressive strategy has been completed. Major design trade-offs are related with accuracy of the reconstruction and frame rate, because of compressed samples being too few or inaccurate. Time-encoding of the pixel values and tagging of asynchronous pulses coming from a collection of pulses is the methodology employed to overcome dynamic range limitations in the construction of the compressed samples. Experimental characterization of the prototype will allow verifying the advantages of full-frame compressive strategies versus block-based compressed sampling.

Acknowledgment

This work has been funded by the Spanish Government through projects TEC2015-66878-C3-1-R MINECO (European Region Development Fund, ERDF/FEDER), by Junta de Andalucía through project TIC 2338-2013 CEICE and by the Office of Naval Research (USA) through grant N000141410355 and CONACYT (Mexico) through grant MZO-2017-291062.

References

[1] Candès. "Compressive sampling". *Int. Congress of Mathematics*, pp. 1433-1452. Madrid, Spain. August, 2006.

[2] J. Romberg. "Imaging via Compressive Sampling". *IEEE Signal Processing Magazine*, Vol. 25, No. 2, pp. 15-20. March, 2008.

[3] R. G. Baraniuk, V. Cevher, M. B. Wakin. "Low-Dimensional Models for Dimensionality Reduction and Signal Recovery: A Geometric Perspective". *Proc. of the IEEE*, Vol. 98, No. 6, pp. 959-971. Jun 2010.

[4] M. B. Wakin, J. N. Laska, M. F. Duarte, D. Baron S. Sarvotham, D. Takhar, K. F. Kelly, R. G. Baraniuk. "An Architecture for Compressive Imaging". *IEEE International Conference on Image Processing*, pp. 1273 – 1276. Atlanta, USA. October, 2006.

[5] V. Majidzadeh, L. Jacques, A. Schmid, P. Vandergheynst and Y. Leblebici. "A (256x256) Pixel 76.7mW CMOS Imager/Compressor Based on Real-Time In-Pixel Compressive Sensing". *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2956 – 2959. Paris, France. May, 2010.

[6] Y. Oike, A. El Gamal. "CMOS Image Sensor With Per-Column ΣΔ ADC and Programmable Compressed Sensing". *IEEE Journal of Solid-State Circuits*, Vol. 48, No. 1, pp. 318 - 328, January. 2013.

[7] B. Kaliannan, V S. Rao Pasupureddi. "A Low Power CMOS Imager Based on Distributed Compressed Sensing". *27th International Conference on VLSI Design and 13th International Conference on Embedded Systems*, pp. 534 – 538. Mumbai, India. January. 2014.

[8] M. Dadkhah, M. Jamal Deen, S. Shirani. "CMOS Image Sensor With Area-Efficient Block-Based Compressive Sensing". *IEEE Sensor Journal*, Vol. 15 No. 7, pp 3699-3710, July, 2015.

[9] M. Trevisi, R. Carmona-Galán, Á. Rodríguez-Vázquez, "Compressive Image Sensor Architecture with On-Chip Measurement Matrix Generation". *IEEE Int. Conf. PhD Research in Microel. and Electronics (PRIME 2017)*, pp. 25-28, Taormina, Sicily (Italy), June 2017.

[10] E. Jen. "Aperiodicity in one-dimensional cellular automata". *Physica D: Nonlinear Phenomena*, Vol. 45 No. 1-3, pp 3-18. September, 1990.

[11] Lu Gan "Block Compressed Sensing of Natural Images", *15th International Conference on Digital Signal Processing*, pp. 403-406. Singapore, Republic of Singapore. July, 2007.

[12] D. G. Chen, D. Matolin, A. Bermak, C. Posch, "Pulse-Modulation Imaging—Review and Performance Analysis". IEEE Trans. on Biomedical CAS, Vol. 5, No. 1, pp. 64-82, February 2011.

[13] G. Sudhish, P. Deepthi. "A Secure LFSR Based Random Measurement Matrix for Compressive Sensing". *Sensing and Imaging*, New York, Springer, 2014, Vol. 15 No. 1.

[14] L. Wang, Y. Zhao Z. Dai. "A Random Sequence Generation Method for Random Demodulation Based Compressive Sampling System". *International Journal of Signal Processing, Image Processing and Pattern Recognition*, Vol. 8 No. 1, pp 105-114, 2015.