A Low Power CMOS Imager based on Time-to-First-Spike encoding and Fair AER

Chen Shoushun and Amine Bermak
Electrical and Electronic Engineering Department
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong, SAR.
Email: dazui,eebermak@ust.hk

Abstract— **This paper presents a CMOS vision sensor based on a biologically inspired data representation referred to as Time-to-First-Spike (TFS) encoding combined with a fair Address Event Representation (AER) scheme. Our approach is different from conventional methods because the read-out of information is initiated by the pixel itself while access to the read-out bus is granted only once to each pixel after which it enters into a stand-by mode. This approach allows to greatly save dynamic power consumption and to extensively reduce inefficiencies due to periodical requests of the bus in the case of spiking pixels. Transmission bandwidth is thus significantly improved using the proposed circuitry. Each pixel includes only 14 transistors and occupies an area of $15 \times 15 \mu m^2$, with a fill factor of 33% using $0.35 \mu m$ process. The average current consumption is estimated to $10 nA$ per pixel, which is 3 orders of magnitude lower compared with that of the spiking pixel.**

I. Introduction

It is widely expected that Complementary Metal-Oxide Semiconductor (CMOS) image sensors would allow the realization of the next generation on-chip visual systems ultimately associating image capture devices with intelligent processing such as sensory adaptation, motion detection and on-line image compression. The main obstacles facing the designers of this next generation visual systems are the very high bandwidth requirements due to real-time processing and higher power consumption due to more complex processing and the scanning read-out used in traditional image sensors. In conventional CMOS camera, images are read-out using a clock, which switches the multiplexer from one sensor to another, reading a brightness value from each sensor at a fixed interval, hence called "scanner". Images are therefore produced by sequentially scanning the array using column and row scanners. Scanning read-out strategies will soon fall short of meeting higher resolution and fame rate requirements ($\geq 256 \times 256$ and 100 frames/s), and hence new approaches are therefore required to overcome these limitations. Address Event Representation (AER) [1] combined with the spiking pixel architecture was proposed in order to provide efficient allocation of the transmission channel to only active pixels [2]. While this concept has its own merits as it introduces for the first time the idea of pixel-driven parallel read-out, however the approach suffers from the inherent disadvantage of the spiking nature of the pixel, which constantly fires and periodically requests access to the bus. Recent biological studies [3] reviewed a number of arguments for taking into account the temporal information that can be derived from the very first spikes in the retinal spike trains. The study suggests that retinal encoding can be performed in the Time to First Spike (TFS) information rather than the frequency of the spikes. In building CMOS vision sensors the two approaches can be equally used to convert luminance into a measurable variable. In the TFS case the information is encoded in the delay in obtaining the first spike while in the spiking pixel case the information is encoded in the rate or the firing frequency. While both concepts provide a viable mean to build a vision sensor, both the operation of the pixel and the read-out strategy are fundamentally different. In the spiking pixel based AER, brighter pixels are favored because their integration threshold is reached faster than darker pixels. Consequently, brighter pixels request the output bus more often than darker ones. This results in an unfair allocation of the bandwidth as well as congested read-out bus because of the periodical request due to the spiking nature of the pixel. This imposes higher constraints on the AER processing speed and induces more dynamic power consumption and temporal jitter affecting the SNR. In this paper, we propose to overcome these problems by using Time-to-First-Spike (TFS) encoding combined with a fair and high speed Address Event Representation. Our approach is different from conventional methods because the read-out of information is initiated by the pixel itself while access to the read-out bus is granted only once to each pixel after which it enters into a stand-by mode. This approach allows to greatly save dynamic power consumption and to extensively reduce inefficiencies due to periodical requests of the bus in the case of spiking pixels. The paper is organized as follow: Section II introduces the imager architecture including the TFS based concept together with its simulation results. Section III introduces the AER architecture and its implementation. Section IV concludes the paper.

II. TFS based pixel

The key idea behind the TFS is to encode the illumination information into the latency recorded in order to obtain the first spike from an event generator. Only a single transition is thus required for each pixel which results in a number of benefits mainly related to saving in terms of power consumption as well as efficient utilization of the communication bandwidth with the peripheral circuits.

A. Pixel description

Figure 1 shows the schematic of the TFS based pixel. The circuit is composed of a photosensor (photodiode P_d) with its internal capacitance C_d, a reset circuit, composed of the parallel combination of the PMOS transistors *m1* and *m2* followed by a recently proposed [4] elegant current feedback event generator *(m3-m7)*. Transistors *(m8-m14)* are used in order to implement the asynchronous communication with the column and row AER circuit.

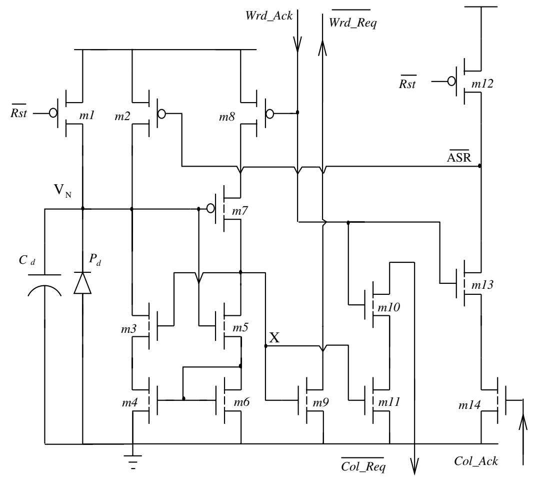

Fig. 1. TFS Pixel Schematic.

An active low $\overline{Rst}$ pulse is used to reset the pixels after which the integration process starts. The light falling onto the photodiode P_d will start discharging the internal capacitor of the photodiode C_d. This results in a linearly decreasing voltage V_N across the node of the photodiode. Once this voltage reaches the threshold voltage of the inverter *(m5, m7)*, an event will be generated at node "X" and a request signal is sent to the row AER through the global signal $\overline{WrdReq}$. The row AER will process all the row requests and will grant an acknowledgment signal *(WrdAck)* to a single row. At this stage, all pixels that generated an event within the acknowledged row will send a request $\overline{ColReq}$ to the column AER and will asynchronously self-reset the photodiode node by turning on transistor m2 once got acknowledged again by the column AER. The pixel will then enter into a stand-by mode until a new global start integration signal is received. The TFS based pixel was successfully simulated and simulation results are reported in Figure 2. The figure shows the sequence required in a full cycle which can be described as: *Start Integration* $\mapsto$ *Event Generation* $\mapsto$ *Row Request* $\mapsto$ *Row Acknowledgment* $\mapsto$ *Column Request* $\mapsto$ *Column Acknowledgment* $\mapsto$ *Self Reset*. It is very important to note that a global reset is omitted and instead each pixel is responsible to self-reset itself after which it enters a stand-by mode until a new start integration signal is received. The time to generate the first spike is used in order to encode the pixel brightness. The latency of time to first spike T_f is given by

$$T_f = \frac{(V_{dd} - V_{TH}) \times C_d}{I_d}, \tag{1}$$

where I_d and V_{TH} are the photocurrent and the threshold

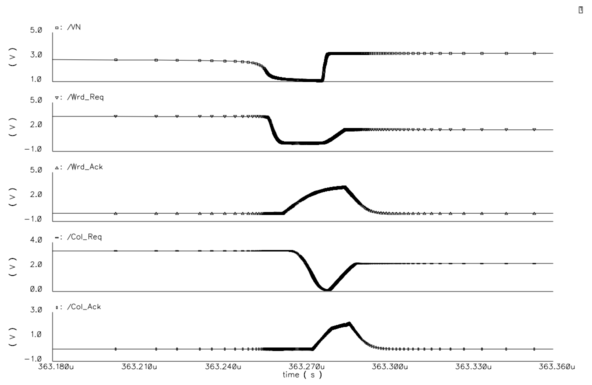

Fig. 2. TFS Pixel simulation results. Signals from top to bottom are: photodiode voltage V_N, Row request, Row Acknowledgment, Column request and Column Acknowledgment, respectively.

voltage of the inverter *(m5, m7)*, respectively. The row and column acknowledgements signals are encoded as an address data for the event. An asynchronous event driven imager is therefore realized based on *"single transition per pixel"* concept. It should be also noted that in this proposed scheme, the charge-up current required to reset the sensing node is kept minimum as the self-resetting operation prevents the discharge of the sensing node. The charge-discharge swing is kept constant ($V_{dd} - V_{TH}$) for all pixels within the array.

B. Imager Architecture

Figure 3 shows how the proposed TFS and AER concepts can be implemented in order to build an ultra low power vision sensor. The architecture comprises a pixel array of m rows and n columns, row and column buffers and a row and column AER organized in a tree structure. When one or more pixels inside a row fires this row will send a request $\overline{WrdReq}$ to the Row AER. The Row AER may receive several requests at the same time. After arbitration, only one row will be acknowledged. The fired pixels within that row then send request $\overline{ColReq}$ to the column AER. In order to avoid waiting for the Column AER to acknowledge all column requests one by one, the Column Buffer will hold the requests and acknowledge back concurrently. This improves the processing speed of Column AER by avoiding charging and discharging the large capacitance of the column buses. The Column AER will therefore process the requests held by the Column Buffers. An active low signal $\overline{CABusy}$ is sent from the Column AER to the Row Buffers to indicate that the row is now being processed. The Row Buffer now can safely kill the previous $\overline{WrdReq}$ by turning on transistor M17. This enables the Row AER to start another round of arbitration thus a parallel Row and Column AER processing is obtained. Once the row AER determines the next row to be processed, the new row acknowledgment signal will be blocked by the row buffer (M20 switched OFF) until the column AER has finished processing the current job. A pipeline processing between row and column AER is therefore achieved which greatly speeds-up the processing of the array.

From Equation 1, it is evident that the time-to-first-spike is inversely proportional to the photocurrent. A control circuit

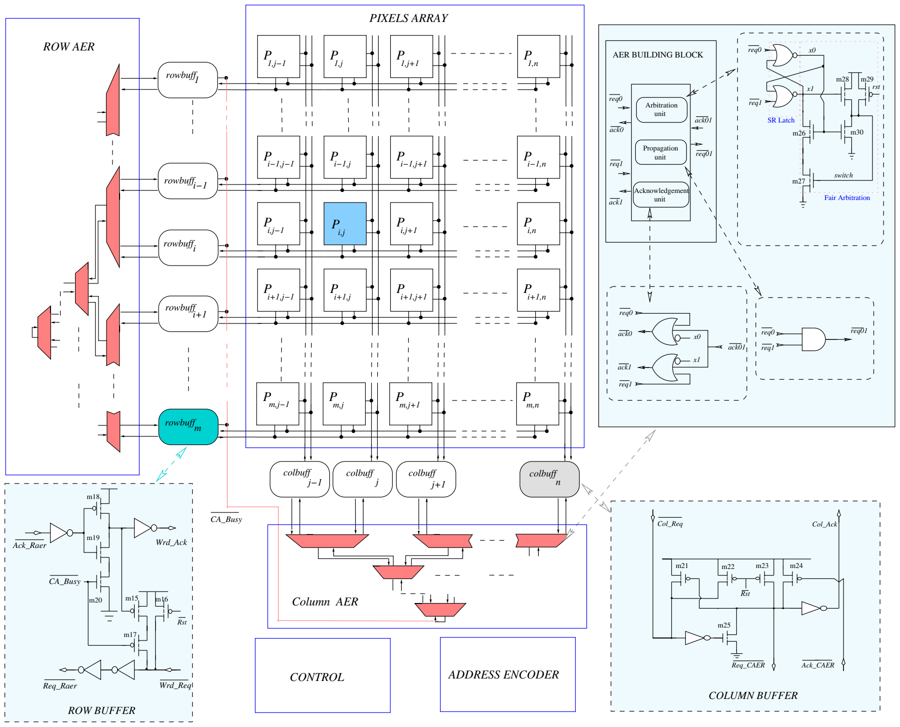

Fig. 3. Imager Architecture

(SRAM + counter) is added in our imager in order to compensate for this non-linearity by adjusting the quantization levels of a sampling counter circuit. The clock signal is modulated in order to adapt the conversion range and to compensate for the inverse relationship between the photocurrent, I_d and the time-to-first-spike, T_f, of the sensor. The combination of Row and Column acknowledgements will uniquely locates a pixel that fires. We then encode these two acknowledgments into address and output it together with the data provided by the control circuit.

III. AER Architecture

One of the major problems facing AER based imagers is the well known collision problem. Assume that at a given time, ρ pixels fire and request access to the bus. An arbiter will grant access to the bus to a given pixel and will place the remaining $\rho - 1$ pixels in a processing queue. A timing error is therefore induced which is proportional to the processing time of each request in the arbitration tree as well as the number of requests received at any given time. It is clear that using a spiking pixel, which periodically generates a spike and hence periodically requests the bus will certainly aggravate the timing error. In our circuit the timing error is reduced using three techniques (i) TFS encoding of the pixel (ii) pipelining strategy using the $\overline{CABusy}$ signal and (iii) hierarchical design. Indeed, in our 128×128 array sensor, we further divided the column AER tree into eight 16-input sub-trees. These sub-trees can process the blocks within one row in parallel thus greatly reduce the timing error compared to that associated to a single AER tree. Accordingly the eight sub-trees will have a separate address encoder while sharing one single data bus. In addition to this timing problem, the design of an arbiter responsible for fair allocation of the bus to the 2D array pixels is of primary importance and is not straightforward.

A. Fair Arbiter

Figure 3 shows the building block in our AER tree. Each building block of the tree can be divided into 3 units: arbitration unit, propagation unit and acknowledgement unit. The arbitration unit is constituted of an *RS* latch composed of two cross-coupled *NOR2* gates and five additional transistors used to provide fair arbitration. Figure 4 shows the principle behind our fair arbitration concept. Initially *M27* is turned

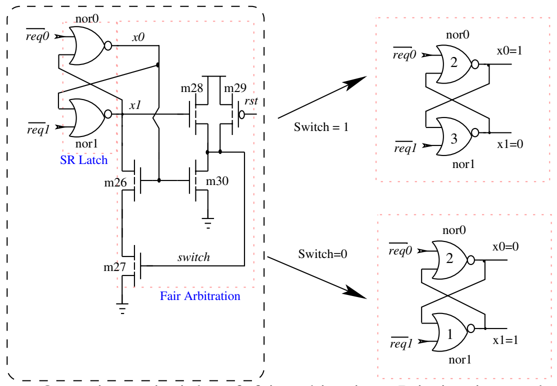

Fig. 4. Operation principle of fair arbitration. Priority is toggled after arbitration has taken place as the pulling down capability of the top *NOR2* gate depends on the *switch* signal.

ON by the global reset, providing the top *NOR2* gate a larger pulling down capability compared to the bottom *NOR2* gate. If the two requests "$\overline{req0}$" and "$\overline{req1}$" arrive at the same time, competition will occur and the top *NOR2* gate will gain priority, i.e. "$x_0 = 1$" and "$x_1 = 0$". The result is maintained until the arbiter receives an acknowledgement from higher stages and then "$\overline{ack0}$" will be activated. At this stage, transistor *M27* is turned off and the bottom *NOR2* gate gains priority over its counterpart. The priority is therefore toggled as the pulling down capability of the top *NOR2* gate depends on the *switch* signal, which is toggled after each arbitration process. This makes the arbitration a fair process. It should be also noted that the two *NOR2* gates always have different pulling down capability and this allows to avoid meta-state of the *SR* latch. The simulation results of this fair arbitration process is shown in Figure 5. One can note from this figure that initially "$\overline{req0}$" and "$\overline{req1}$" arrive at the same time and "$\overline{req0}$" is acknowledged first ($\overline{ack0} = 0$) followed by "$\overline{req1}$" ($\overline{ack1} = 0$). A second "$\overline{req0}$" is received and processed. The priority is hence toggled to "$\overline{req1}$" which explains why "$\overline{req1}$" is processed first in the third cycle.

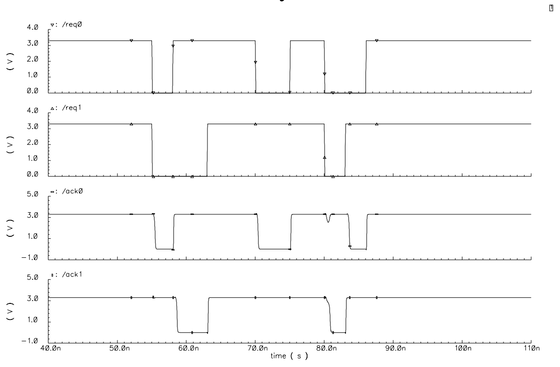

Fig. 5. Simulation results of the fair arbitration process.

B. Higher radix arbiter tree

In order to reduce the effect of timing errors due to collision problem, it is extremely important to reduce the depth of the arbiter tree and the delay of the arbiter building block. The delay in the tree arbiter can be expressed as $\Theta = \theta \times \log_r m$ where θ, m and r are the delay of the basic building block, the array size and the radix (or the number of inputs per building block), respectively. By increasing the radix r the depth of the tree $\log_r m$ is hence reduced which would improve the global delay Θ if we can maintain θ at acceptable level.

TABLE I
DELAYS OF A SINGLE BUILDING BLOCK AND AN ARBITER TREE FOR DIFFERENT RADIX (r), DIFFERENT ARRAY SIZE (m) AND FOR FIXED (*FP*) AND INTERCHANGEABLE PRIORITY (*IP*).

Operation type	θ		Θ			
	$r = 2$	$r = 4$	$r = 2$		$r = 4$	
			$m = 16$	$m = 64$	$m = 16$	$m = 64$
FP single Request	0.33n	0.49n	1.32n	1.98n	0.98n	1.47n
FP multi. Requests	0.39n	0.55n	1.56n	2.34n	1.1n	1.65n
IP single Request	0.37n	0.55n	1.48n	2.22n	1.1n	1.65n
IP multi. Requests	0.41n	0.59n	1.64n	2.46n	1.18n	1.77n

An AER building blocks with $r = 4$ was designed and its delay was evaluated and compared with the case where $r = 2$ for both interchangeable (*IP*) and fixed priority (*FP*). In addition the global performance of the tree based on the two building blocks and for different array sizes are reported in Table 1. One can note that for array size of 64, higher radix arbiter tree reduces the global delay by more than 25%.

The vision sensor was implemented in Alcatel $0.35\mu m$ CMOS technology. Figure 6.A shows the pixel layout while figure 6.B shows the whole imager layout. Each pixel occupies an area of $15 \times 15\mu m$ with a fill factor of 33%.

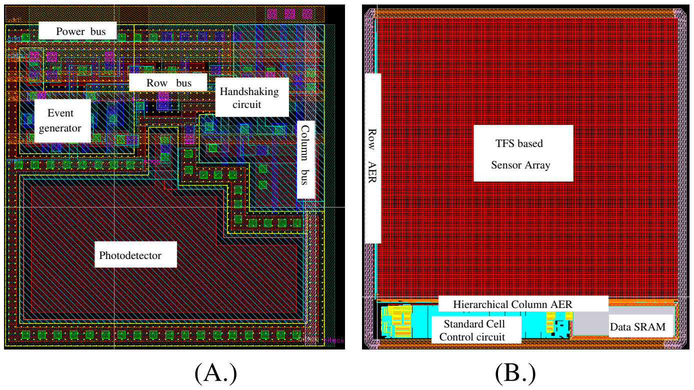

Fig. 6. (A.) Layout of TFS Pixel (B.) Layout of Imager

IV. Conclusion

This paper presents a vision sensor based on TFS and fair AER. In contrast to a recently reported biomorphic digital image sensor [2], access to the read-out bus is granted only once to each pixel after which it enters into a stand-by mode. This approach allows to greatly save dynamic power consumption and to extensively reduce inefficiencies due to periodical requests of the bus in the case of spiking pixels. Transmission bandwidth and timing error due to collision are greatly improved. In addition the delay of the AER is also reduced by using three strategies namely: (i) pipelining, (ii) hierarchical column tree and (iii) higher radix arbiter. The paper also proposed a very simple and yet a reliable way to implement a fair arbitration by only using 5 extra transistors in the AER building block. The imager was designed in $0.35\mu m$ CMOS technology. Each pixel includes only 14 transistors and occupies a very compact area of $15 \times 15\mu m^2$, with a fill factor of 33%. The average current consumption is estimated to $10nA$ per pixel, which is 3 orders of magnitude lower compared with that of the spiking pixel [5].

Acknowledgment

The work described in this paper was supported by a grant from the Research Grant Council of HK SAR, China (Project No HKUST6148/03E).

References

[1] Kwabena A.Boahen, "Point-to-point connectivity between neuromorphic chips using address events", IEEE Trans CAS II, Vol. 47 , No. 5 , 2000, pp.416 - 434.
[2] Culurciello, E.; Etienne-Cummings, R.; Boahen, K.A., "A biomorphic digital image sensor ", IEEE JSSC , Vol. 38, No. 2 , 2003, pp. 281 -294.
[3] Van Rullen, F.; and Thorpe, S. J., "Rate Coding Versus Temporal Order Coding: What the Retinal Ganglion Cells Tell the Visual Cortex", Neural Computation, 13, 2001, 1255-1283.
[4] Culurciello, E.; Etienne-Cummings, R., "Second generation of high dynamic range, arbitrated digital imager", ISCAS, pp. 828-31, 2004
[5] Jens Doge, et Al, "An HDR CMOS imager Sensor with Spiking Pixels, Pixel Level ADC, and Linear Characteristics", IEEE Trans. Circuits and Systems II, Vol 49, No.2, P155-188. Feb 2002