

Asynchronous, event-driven readout for large-scale imaging devices

Prafull Purohit, Rajit Manohar

Computer Systems Lab, Yale University, New Haven, CT, USA

{prafull.purohit, rajit.manohar}@yale.edu

Abstract—Event-driven readout architecture offers an energy-efficient solution for reading out information from pixels when events are sparse. A frame-based readout, on the other hand, can quickly read a group of pixels without requiring multiple communication handshakes on the output bus. However, neither of the readout schemes performs well for large-scale imaging devices, with millions of pixels, when both event-based and frame-based imaging need to be supported. In this work, we propose a better readout mechanism based on token rings, which allows both event-based and frame-based readout. We introduced a shared channel to transmit additional data from the pixel, such as the integrated brightness intensity or the pixel address. We synthesized the design, verified the correct functionality using SPICE simulation, and demonstrated the use of a shared channel to communicate meaningful data from every pixel.

Index Terms—vision sensor, event-driven readout, address-event representation (AER), arbitration, neuromorphic, asynchronous

I. INTRODUCTION

In addition to computing and communication, the nervous system also performs the important task of sensing the physical environment through its primary receptors and converting the sensed stimulus into the asynchronous stream of spikes (events) which are then processed in the brain. A typical event-based sensing can be considered a data-dependent sampling where output is produced whenever the external stimulus changes by a pre-defined amount. The output in such sampling schemes can be either an asynchronous spike or some delta (difference in value compared to the previous output). The high efficiency of the event-based sampling approach used by the nervous system has inspired people to also use a similar approach for signal processing. As a result, several event-based sensors have been designed in the visual, auditory, and olfactory domains.

Event-based vision sensors, in particular, have gained a lot of attention recently due to their application in machine vision (inspection, monitoring), security & surveillance (motion/gesture detection), mobility (autonomous driving, drones), medicine (vision restoration), and scientific imaging. As a result, several event-based vision sensors using custom event-driven, asynchronous readouts have been designed.

Individual pixels in these sensors measure local changes in the incident light intensity (brightness) and produce an

asynchronous event. The events are then communicated through a shared output bus in the form of address-event representation (AER). As shown in Fig. 1, a typical AER-based readout circuit multiplexes the event location as an encoded address on the output bus.

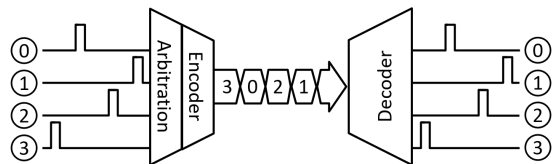


Fig. 1: AER Communication

The ability to read sparse events, low latency, and the low-power operation of such event-based readout mechanism is particularly important for large systems of detectors like those used in physical sciences. A typical image sensor in such a detector contains millions of pixels, each outputting application-specific data. Depending on the application, this data can be intensity information, peak amplitude, number of incident photons, time difference between two events, etc. Unfortunately, existing event-driven readout circuits only communicate an optional 1-bit *polarity* information from an event along with pixel location and are not designed to communicate a wide range of data. Brandli *et al.* [1] designed a vision sensor with dedicated readout circuits to send pixel events and intensity information concurrently.

In this paper, we present a scalable readout mechanism for image sensors using a ring of mutual exclusion elements. The proposed design can support sequential scanning of the frame-based readout as well as the sparse, event-based readout. Fig. 6 shows the conceptual diagram of the readout circuit. Instead of using tree-based arbitration logic outside the pixel array, a token ring connects all pixels in a row, and a second token ring connects all row-level rings. A single token travels through this mesh of token rings to process the event promptly. Once a pixel receives the token, a second shared communication channel can be used to perform supporting tasks, such as updating the address counter on the periphery of the chip, sending integrated intensity information, or event timestamps.

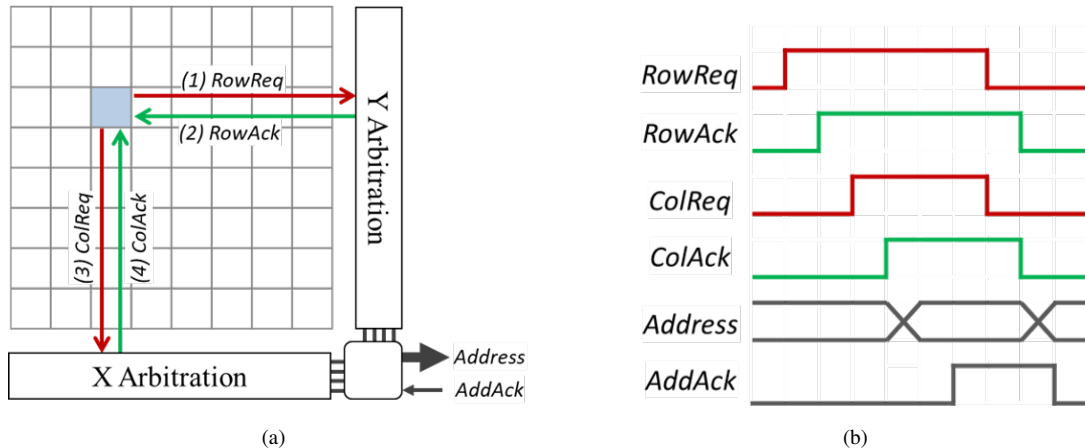


Fig. 2: Event-based readout

In section II we present some of the previously reported readout schemes for event-based vision sensors. Implementation details of the proposed readout circuit are provided in Section III. Section IV presents simulation results and a discussion on the benefits of the proposed design compared to existing work. Section VI concludes this paper.

II. EVENT-BASED READOUT

The main purpose of a readout circuit in an imaging device is to multiplex the pixel data onto a shared output bus. A range of readout schemes exists based on the application requirements with direct addressing and scanning approaches being the popular mechanism. In the direct addressing readout, a unique pixel index provides an address for each pixel that drives the output bus while keeping the driving circuit in other pixels disabled. As a result, this type of readout allows random access to any pixel in the array. The scanning readout uses a flip-flop based shift register chain to sequentially enable individual pixels which then drives the output bus.

The primary goal of an event-driven readout architecture is to offer an energy-efficient solution for reading out information from pixels when events are sparse. A simple implementation of such an event-based readout is shown in Fig. 2. In order to manage a large number of pixels, the event-based readout mechanism is divided into separate AER circuits for rows and columns. A pixel with event starts by requesting access from the row arbitration circuit. After getting the access in the form of acknowledgment, it then communicates with the column arbitration circuit. The acknowledgment from column arbitration grants the pixel exclusive access to the shared output bus for communicating pixel address. The pixel returns the access (token) to row and column arbitration after communicating the pixel address and the whole cycle repeats for another pixel with an event. The sequential order of the row and column arbitration before an actual readout communication provides an efficient

readout mechanism for sparse events while avoiding any issue related to contention on the output bus in situations where multiple pixels generate an event. However, the same sequential order resulted in a slow readout due to multiple unnecessary handshake communications for each event. As a result, recent implementations have focused on reducing the amount of redundancy in the communication of an event. In this section, we present some of the work done in the field to improve the data rate of such readout systems.

A. Word-parallel AER

Word-parallel AER [2] is the earliest adaptation of AER communication for vision sensors. In this type of AER, shown in Fig. 3, the X, and Y addresses of the pixel are communicated in parallel.

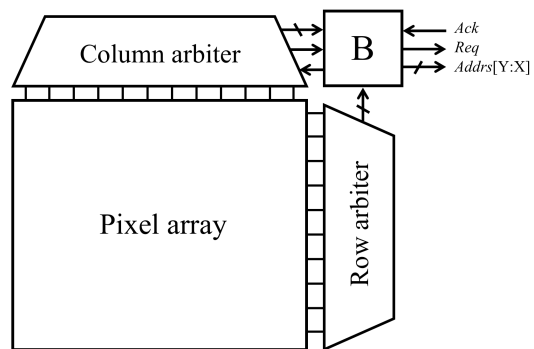


Fig. 3: Word-parallel AER

In a typical word-parallel AER, the communication between the vision sensor and receiver involves the following three steps:

1) Row selection

When a pixel detects an event, the readout logic initiates a row handshake, and the shared request signal from all the pixels in a row is communicated to

the row arbitration circuit. The row arbitration selects a row containing pixels with an event.

2) Column selection

When a row is selected by the row arbitration, all the pixels in the selected row initiate a column handshake. Similar to the row selection, an arbitration logic selects one column for readout.

3) Receiver communication

After a pixel is granted access to the output bus, the RowAck and ColAck (acknowledge signals for row and column) are used to encode the x and y coordinates of the selected pixel as address. The encoded X, and Y address is communicated to the receiver on the address channel through a buffer (B).

At the end of step 3, the whole cycle repeats for another pixel with an event.

B. Word-serial AER

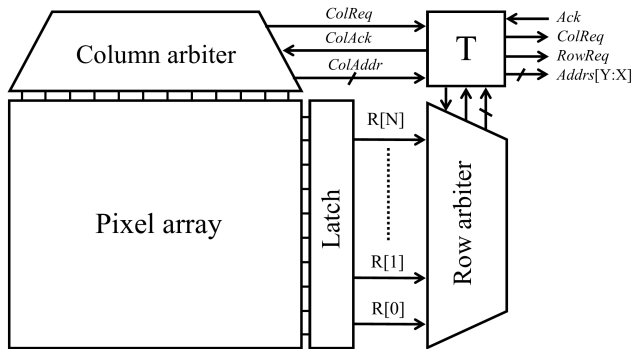


Fig. 4: Word-serial AER

The word-parallel AER provided a simple and easy-to-design mechanism for reporting pixel events in a vision sensor but it has some drawbacks. The separate address bus for row and column addresses requires more I/O pins. Another issue with the word-parallel AER is the redundant handshake for row address, i.e. need for communicating the row and column address for every pixel. When a column is selected through arbitration, we know that the column address for all pixels in the column is going to be the same. This observation is utilized in the *Burst-Mode Word-Serial Address-Event Link* [3]. In a word-serial AER, shown in Fig. 4, when a column is selected by the column arbitration, all the pixel in that column initiates the row handshake. A latching mechanism on the edge records all the row requests, and a two-way multiplexer (T) sends out the row and column address of pixels with an event. In situations where multiple pixels report an event, their common column address is transmitted only once followed by the row address of all pixels with an event. The word-serial AER thus improves communication bandwidth compared to word-parallel AER.

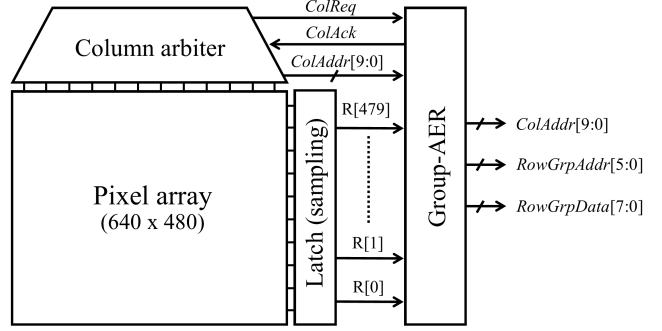


Fig. 5: Group-AER

C. Group-AER

In order to further increase the spatiotemporal resolution, event-based vision sensors need to reduce pixel size and increase the data rate for faster readout. The advances in microelectronics have allowed a reduction in the pixel size for vision sensors. As a result, the resolution has increased from VGA to QuadVGA and larger. For supporting higher event rates, the readout mechanism in such sensors has migrated towards a group- or vector-based, sparse readout of active pixels. In a typical implementation [4], the pixels within a row are divided into group(s) or vector(s) and addressed together to reduce the number of bits required for pixel address. As shown in Fig. 5, after the column arbitration circuit selects a column, events from all the pixels in that column are sampled by a special sampling circuit on the periphery. The sampled events are then processed by Group-AER encoder, which divides the list of active pixels into smaller groups and outputs the Group Address (*RowGrpAddr*) along with event status from each pixel in the group as Group Data (*RowGrpData*). In a more recent implementation [5], the column arbitration is also replaced by a sequential row selection circuit similar to the one used in traditional frame-based image sensors.

III. DATA-AER

In recent years, the event-driven vision sensor has attracted a lot of attention from scientific imaging communities in academia and industry. The primary reasons for such interest are their promising properties compared to standard frame-based cameras, such as low power consumption, sub-ms latency, sparse output, and increased availability of such sensors. Despite the efficiency of data processing by the sparse output of an event-driven vision sensor, traditional computer vision algorithms cannot be readily applied because no static scene information is encoded [1]. By combining the event detection capabilities of event-based vision sensors and intensity measurement of frame-based image sensors, one can track fast-moving objects and read full images for further analysis using well-established machine vision techniques such as object recognition and

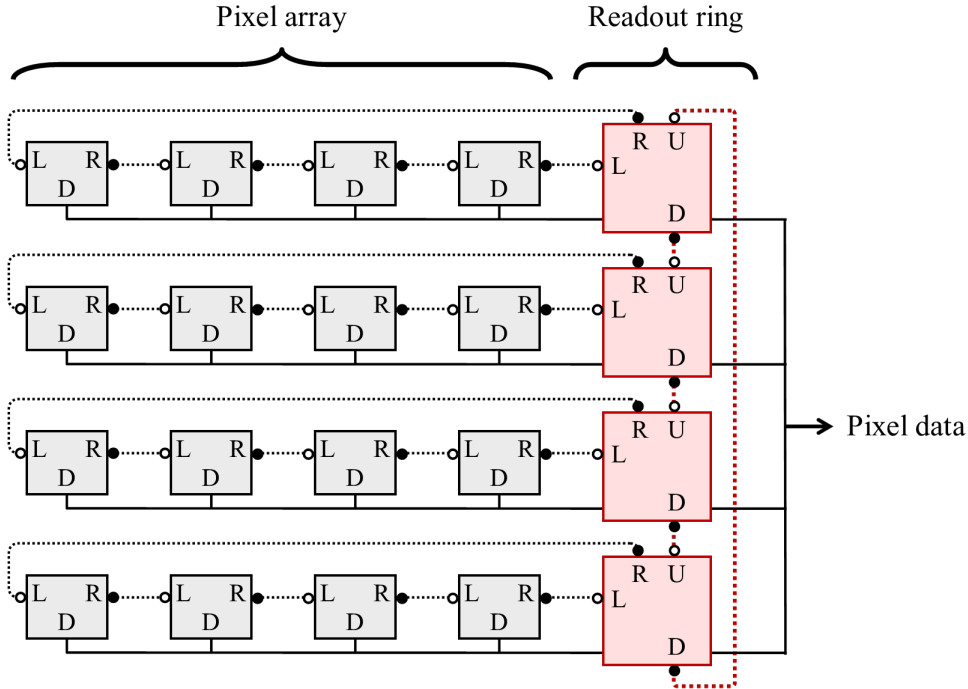


Fig. 6: Data AER based on hierarchical token rings.
 \dashrightarrow denotes the passive end of the channel and \rightarrow denotes the active end of the channel.

classification. Posch et. al. [6] made the first attempt towards this goal by designing an asynchronous time-based image sensor with event-based change detection and pulse-width-modulation (PWM) based exposure measurement circuit in each pixel, and their separate AER-based readout circuits. Brandli et al. [1] designed a vision sensor that combines an event-driven and frame-based readout circuit with a shared photodiode, allowing simultaneous output of asynchronous events and synchronous frames. A similar approach embedding high-frame event pixels with CIS imaging pixels was recently proposed by Sony Semiconductors. In [7] they proposed a shared approach where a single photodiode is shared between different types of pixel front-end and readout circuits while in [8] an approach with separate photodiode and pixel front-end was proposed to enable acquisition of both image and event data. A major drawback of such an approach is the separate circuit for both readout modes with increased penalty in chip area and energy consumption.

In summary, based on the recent work discussed above, we observe two trends:

- 1) Transition from a purely asynchronous, event-driven vision sensor to a frame-based vision sensor with sparse readout.
- 2) Ability to output both events and frames (hybrid vision sensors).

A. Readout scheme

In this work, we designed a new readout mechanism based on hierarchical token rings which allow both event-based and frame-based, scanning readout. A general representation of the readout scheme is shown in Fig. 6. A token ring connects pixels in a row and a second-level readout token ring connects all row-level rings such that a process from the readout ring also participates in a row-level token ring. In addition to the token rings, shared communication channels are used to perform additional tasks such as updating the address counter (for tracking token position) or communicating additional data from each pixel. When a pixel receives the token, it can send pixel address, brightness measurements, or a combination of both information using the shared channel. The shared channel can also be extended to communicate analog/mixed-signal information with the help of custom analog circuits. It is worth noting that because of the hierarchical arrangement of the token ring, the readout scheme can be divided into banks (groups of pixels) to increase readout speed by processing multiple events in parallel.

B. Hardware implementation

We start by describing the circuit functionality using Communication Hardware Processes (CHP) and synthesizing into a Production Rule Set (PRS) for CMOS implementation using Martin's synthesis method [9] and the open-source ACT EDA flow for digital asynchronous circuits [10].

1) *Row server*: The row server (Rserver), shown in Fig. 7, is the process in every pixel and serves as the interface for processing an event. Each Rserver communicates with the Rservers in neighboring pixels through passive communication channel L and active communication channel R . Passive communication channel S with associated data bus (S.r, S.e, Din) is used to communicate with the pixel logic whereas active communication channel D with associated shared data bus (D.r, D.e, Dout) is used to send pixel information to the output channel. The bundled-data channel D can be used to communicate pixel address (in event-mode), brightness measurement (in frame-mode), or both information. The CHP of the Rserver, shown in Fig. 8 is described as:

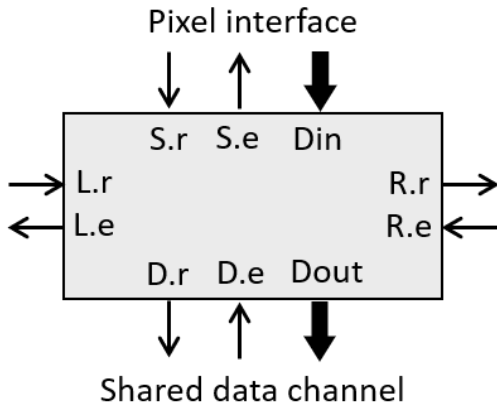


Fig. 7: Row server

$Rserver \equiv$

```
*[[  $\bar{S} \rightarrow [b \rightarrow skip \ \square \ -b \rightarrow R!; b\uparrow]; D!; S?$ 
|  $\bar{L} \rightarrow [b \rightarrow skip \ \square \ -b \rightarrow R!; b\uparrow]; b\downarrow; L?$ 
]]
```

Fig. 8: CHP description of the Row server

When the Rserver receives a request to communicate an event on channel S , it sends the information on channel D if it has the token. If Rserver doesn't have the token then it requests a token from the neighboring Rserver using channel R , updates the local token variable, and sends the information on channel D . Similarly, when a neighboring Rserver requests the token through the channel L , it sends the token and updates the local token variable. In situations where a dedicated counter is used to track token position, the Rserver can update an address counter on the periphery using the shared channel. In the case of simultaneous requests on channel S and L , Rserver arbitrates between the two requests and selects one of them.

2) *Edge server*: The edge server (Eserver), as shown in Fig. 9, connects Rservers and facilitates the token movement across the pixel array. It helps in moving a token within row of pixels with the help of passive communication channel L and active communication channel R . Communication channels U and D are used to move the token between different rows. Apart from the token movement, Eserver also helps in buffering the shared bundled-data output channel. The CHP description of Eserver is given in Fig. 10.

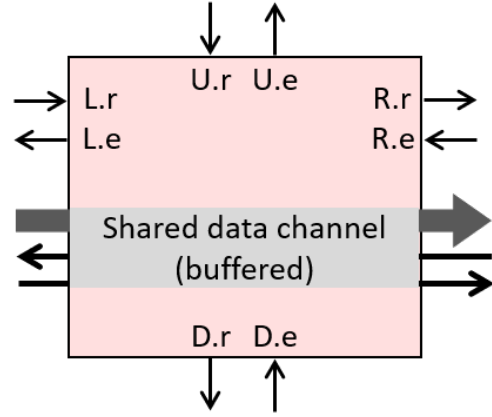


Fig. 9: Edge server

$Eserver \equiv$

```
*[[  $\bar{L} \rightarrow [ b = 0 \rightarrow skip$ 
|  $b = 1 \rightarrow R!$ 
|  $b = 2 \rightarrow D!$ 
];  $b := 1; L?$ 
|  $\bar{U} \rightarrow [ b = 0 \rightarrow skip$ 
|  $b = 1 \rightarrow R!$ 
|  $b = 2 \rightarrow D!$ 
];  $b := 2; U?$ 
]]
```

Fig. 10: CHP description of the Edge server

In implementations where a counter is used to track token position, an additional channel can be used to update the address counter which represents the selected row in the array and provides the most significant bits (MSB) of the token address. Similar to Rserver, a local variable (b) is used in each Eserver to represent token position i.e. which row is selected for communicating an event or brightness value. The variable is implemented using a one-of-three encoding and indicates the token position as:

- $b = 0$: Eserver has the token
- $b = 1$: One of the servers in the corresponding row has the token
- $b = 2$: Eserver and its corresponding row don't have the token

IV. RESULTS AND DISCUSSION

The data-aer based readout scheme discussed above has been implemented in a standard 65 nm bulk CMOS technology and simulated with a 1V supply at a nominal device temperature of 25°C to verify the correct functionality of the circuit implementation. A small capacitance was added on the output of every gate to account for parasitic loads and get a more accurate estimate of the circuit performance. The SPICE simulations were performed using Xyce, a high-quality open-source SPICE simulator [11].

The overall circuit performance depends on many design choices such as the number of processes (Rserver and Eserver) included in a token ring i.e. number of pixels in the array, and how many banks are implemented in the array. To allow designers to estimate circuit performance in different use cases, we divided the circuit operation into two major parts:

- **Token movement:** During this operation, we measured the latency and power consumption for moving a token from one process (Rserver/Eserver) to its neighbor process.
- **Data transmission:** In this operation, we measured the latency and power consumption for processing a request from a pixel and transmitting the input data. This operation assumes that the process already has the token.

The simulation results of both operations are presented in Table I.

TABLE I: Data-AER simulation results

Operation	Delay	Power	Energy
Token movement	0.7 ns	173 μ W	121.1 fJ
Data transmission	2.9 ns	26 μ W	75.4 fJ

TABLE II: Area estimates for Data-AER

# Pixels (row x column)	Area estimates (# gates)
16 pixels (4x4)	1,714
64 pixels (8x8)	6,139
256 pixels (16x16)	23,000

To evaluate circuit complexity, we also compare the estimated design area for different sizes of the array. The area is usually measured in μm^2 but this value depends on the fabrication technology, the amount of layout optimizations, and the cell library. We calculated design area as *gate equivalents* (GE) where one GE is equal to the area of a 2-input NAND gate designed using gridded cell layout style [12]. Based on the GE estimate, given in Table II, we observe that the circuit area scales linearly with the number of pixels. The readout speed for a given array can be increased by dividing the array into smaller banks/groups

of pixels and reading them in parallel while keeping the area or the number of Rserver/Eserver processes roughly the same. A quantitative comparison of different arbitration mechanism for AER communication under variety of event-rate is provided in [13], [14].

The proposed design offers many benefits compared to the existing readout mechanisms for event-driven vision sensors. Some of the benefits are discussed below:

- **Timestamp**

In arbitration-based, event-driven readout when multiple requests arrive in a short time window, one of the requests wins the arbitration while all other requests are queued until the winning input releases the output bus. As a result, the event-handling order becomes unpredictable and causes errors in processing time even between the neighboring pixels. An external timing reference in the form of a global timer or clock is used typically to timestamp individual events. The ability to send additional data from each pixel is an interesting property of the proposed readout circuit which could be used for capturing in-pixel timestamps.

- **Pixel addressing**

The advances in CMOS technology have allowed designers to make larger vision sensors with smaller pixels. This improvement results in an increased number of pixels and therefore increase in the number of events in the sensor. In such situations, the classical approach of designing vision sensors using AER circuits based on the arbiter tree was efficient only up to some limit. The proposed design can support readout similar to the Group-AER by connecting multiple neighboring pixels to a single row server *RServer* such that each process captures event status from the pixels and combines it with hard-wired group address before sending it out.

- **Throughput**

One major drawback of the word-parallel readout is a reduction in throughput with an increase in the number of inputs because every request has to propagate through $\log_2(N)$ stages. The word-serial readout provides an improved solution but the improvement depends on the location and timing of the event. A ring-based readout [13] with a shared counter for tracking the event address also suffers from slower token movement and hence slower throughput. This is because each process in the ring has to complete a full handshake on the counter increment channel before it can release the token. An increase in the number of processes in the ring further reduces the throughput due to the increase in capacitive load on the wires. The hard-wired address in each pixel removes the delay associated with incrementing a shared counter

or computing the address like word-serial or word-parallel readouts. This results in improved throughput for readout. In the proposed design, a token does not return to the root after handling pixel request and any new request from neighboring pixels in the row can be serviced quickly, which results in higher throughput.

- **Data payload**

As discussed in Section III, existing event-driven readout circuits have very limited capability to send data payload from each pixel. Each pixel in such sensors communicates an optional 1-bit *polarity* information. However, scientific imaging devices used in physical sciences often have some data (payload) associated with each pixel. In such a situation, existing AER-based readout circuits with polarity output are not useful. The proposed design offers an additional shared channel to send data payload from each active pixel, allowing the use of traditional computer vision algorithms.

V. CONCLUSION

Event-driven readout architecture offers an energy-efficient solution for reading out information from pixels when events are sparse. A frame-based readout, on the other hand, can quickly process a group of events without requiring multiple communication handshakes on the output bus. However, neither of the readout schemes performs well for vision sensors when both event-based and frame-based imaging need to be supported. In this work, we proposed a better readout mechanism, based on the hierarchical token rings, which allows both event-based and frame-based, scanning readout. To take further advantage of the shared resources, we introduced another shared channel to transmit additional data from the pixel. Depending on the target application, this data can be an integrated brightness intensity or pixel address. We synthesized the design, verified the correct functionality using SPICE simulation, and demonstrated the use of a shared channel to communicate meaningful data from every pixel.

APPENDIX

The circuit functionality is described using Communicating Hardware Processes (CHP) language and the key notations of the CHP syntax are summarized below:

- Skip: No operation
- Send: $X!v$ means send the value of v over channel X .
- Receive: $X?v$ means receive a value on channel X and store it in variable v .
- Probe: \bar{X} determines if there is a pending communication on a channel X
- Assignment: $a := b$ means assign the value of b to a .
- Sequential Composition: $S1;S2$ means execute statements $S1$ and $S2$ sequentially

- Parallel Composition: $S1, S2$ means execute statements $S1$ and $S2$ in parallel
- Deterministic Selection: $[G1 \rightarrow S1 \square \dots \square Gn \rightarrow Sn]$ waits until one of the guards ($G1, G2, \dots, Gn$) is *true* and then execute corresponding statement. Requires that the guards must be mutually exclusive
- Non-Deterministic Selection: $[G1 \rightarrow S1 \mid \dots \mid Gn \rightarrow Sn]$ is same as the Deterministic Selection except guards don't have to be mutually exclusive
- Repetition: $*[S]$ infinitely repeats statement S
- Do-while loop: $*[S \leftarrow G]$ executes statement S , and then evaluates the guard G . If G is true, then the loop repeats; otherwise, the loop terminates.

REFERENCES

- [1] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240×180 130dB $3\mu s$ latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.
- [2] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120dB $15\mu s$ latency asynchronous temporal contrast vision sensor," *IEEE journal of solid-state circuits*, vol. 43, no. 2, pp. 566–576, 2008.
- [3] K. A. Boahen, "A burst-mode word-serial address-event link-I: Transmitter design," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 7, pp. 1269–1280, 2004.
- [4] B. Son, Y. Suh, S. Kim, *et al.*, "A 640×480 dynamic vision sensor with a $9\mu m$ pixel and 300meps address-event representation," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, 2017, pp. 66–67.
- [5] Y. Suh, S. Choi, M. Ito, *et al.*, "A 1280×960 dynamic vision sensor with a $4.95\text{-}\mu m$ pixel pitch and motion artifact minimization," in *2020 IEEE international symposium on circuits and systems (ISCAS)*, IEEE, 2020, pp. 1–5.
- [6] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, 2010.
- [7] A. Niwa, F. Mochizuki, R. Berner, *et al.*, "A $2.97\ \mu m$ -pitch event-based vision sensor with shared pixel front-end circuitry and low-noise intensity readout mode," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, 2023, pp. 4–6.
- [8] K. Kodama, Y. Sato, Y. Yorikado, *et al.*, "1.22 μm 35.6 mpixel rgb hybrid event-based vision sensor with 4.88 μm -pitch event pixels and up to 10k event frame rate by adaptive control on event sparsity," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, 2023, pp. 92–94.
- [9] A. J. Martin, "Compiling communicating processes into delay-insensitive VLSI circuits," California Institute of Technology, Tech. Rep., 1986.
- [10] S. Ataei, W. Hua, Y. Yang, *et al.*, "An open-source eda flow for asynchronous logic," *IEEE Design & Test*, vol. 38, no. 2, pp. 27–37, 2021.
- [11] Sandia National Laboratories, *Xyce parallel electronic simulator*, version 7.2, Nov. 2, 2020. [Online]. Available: <https://xyce.sandia.gov>.

- [12] Y. Yang, J. He, and R. Manohar, "Dali: A gridded cell placement flow," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020, pp. 1–9.
- [13] P. Purohit and R. Manohar, "Hierarchical token rings for address-event encoding," in *2021 27th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, IEEE, 2021, pp. 9–16.
- [14] P. Purohit and R. Manohar, "Field-programmable encoding for address-event representation," *Frontiers in Neuroscience*, vol. 16, p. 1 018 166, 2022.