Published in final edited form as:
IEEE Trans Biomed Circuits Syst. 2023 April ; 17(2): 169–179. doi:10.1109/TBCAS.2023.3268130.

FPGA-Based In-Vivo Calcium Image Decoding for Closed-Loop Feedback Applications

Zhe Chen [Member, IEEE],
Computer Science Department, UCLA, Los Angeles, CA 90095

Garrett J. Blair,
Psychology Department, UCLA, NYU Center for Neural Science, New York, NY 10003, USA.

Chengdi Cao,
Computer Science Department, UCLA NYU Center for Neural Science, New York, NY 10003, USA

Jim Zhou,
Computer Science Department, UCLA, Los Angeles, CA 90095

Daniel Aharoni,
Department of Neurology, UCLA.NYU Center for Neural Science, New York, NY 10003, USA

Peyman Golshani,
Department of Neurology, UCLA.NYU Center for Neural Science, New York, NY 10003, USA

Hugh T. Blair,
Psychology Department, UCLA NYU Center for Neural Science, New York, NY 10003, USA

Jason Cong [Fellow, IEEE]
Computer Science Department, UCLA NYU Center for Neural Science, New York, NY 10003, USA

Abstract

Miniaturized calcium imaging is an emerging neural recording technique that has been widely used for monitoring neural activity on a large scale at a specific brain region of rats or mice. Most existing calcium-image analysis pipelines operate offline. This results in long processing latency, making it difficult to realize closed-loop feedback stimulation for brain research. In recent work, we have proposed an FPGA-based real-time calcium image processing pipeline for closed-loop feedback applications. It can perform real-time calcium image motion correction, enhancement, fast trace extraction, and real-time decoding from extracted traces. Here, we extend this work by proposing a variety of neural network based methods for real-time decoding and evaluate the tradeoff among these decoding methods and accelerator designs. We introduced the implementation of the neural network based decoders on the FPGA, and showed their speedup against the implementation on the ARM processor. Our FPGA implementation enables the real-time calcium image decoding with sub-ms processing latency for closed-loop feedback applications.

marchzhe@gmail.com .

Keywords

Calcium Imaging; Closed-Loop Feedback; Decoding; FPGA; Neural Network; Real-Time

I. INTRODUCTION

Calcium imaging is an emerging method that has been widely used for observing neural activity at a large scale in neuroscientific research. The miniaturized calcium imaging microscope is a device that can be head mounted on a live mouse or a rat for monitoring firing activity from hundreds of neurons at single cell resolution *in vivo*, while allowing the rodent to perform behavioral tasks freely in the lab environment [1]. UCLA leads the efforts in developing a series of open-sourced miniaturized calcium imaging microscopes (”miniscopes” [2]). These miniscopes can record calcium activity from rats’ or mice’s brains and transfer them over meter-long flexible coaxial cable to a remote data acquisition (DAQ) board [3]. They have gained popularity quickly among neuroscientific research labs and have accelerated brain research for many subfields related to memory, behavior and disease, to name a few.

Several complete calcium image analysis pipelines have been proposed to extract neuron activity from the raw calcium image recordings. [4] proposed a Python-based pipeline that has been widely used for one-photon calcium image analysis. It consists of separate motion correction [5] and signal extraction [6] steps. The signal extraction relies on the constrained nonnegative matrix factorization (CNMF) approach to identify spatial cell footprints and temporal calcium traces simultaneously. [7] provided a Matlab-based calcium image analysis pipeline, which was also based on the CNMF approach. It employs a long short-term memory (LSTM) inference approach to refine neuron activity detected by a Gaussian mixture model. These pipelines have been widely used in offline calcium image analysis. However, it is hard to be deployed online, because the CNMF approach requires a full stack of images as input. Though some recent work [8] demonstrated that the CNMF-based pipeline achieved real-time throughput on general-purpose CPUs, it is still challenging to implement this on the embedded hardware with short processing latency for closed-loop feedback applications.

Some other research works [9]–[12] attempt to decode behavioral information from the calcium images. [9] used a k-Means method to predict forelimb reach directions based on averaged calcium images. [10] decoded mice’s behavioral states on a linear track by utilizing Laplacian Eigenmaps to extract internal representation of behaviors from high-dimensional neural activity patterns. Though these pioneering works show promising behavioral decoding results, they require batch processing, so they are difficult to be implemented for the online decoding with short latency. [13] combines principal component analysis (PCA) with the linear support vector machine (SVM) to perform decoding during a pressing movement based on offline extracted calcium traces [7]. Some recent works [11], [12] proposed SVM based methods for online calcium image decoding, however, they are evaluated on general purpose CPUs and there is still a lack of embedded hardware and electronic interface for demonstration of short processing latency for closed-loop feedback capability.

Fig. 1 illustrates the main target this paper aims to achieve — the closed-loop feedback capability enabled by real-time calcium image processing and decoding for miniscopes. Such closed-loop feedback capability brings three strict requirements: 1) The computation requires short and deterministic processing latency, in order to generate rapid and precise feedback based on event detection and decoding results from the calcium images. Recent advancement in voltage imaging [14] enables kilohertz frame rate, which escalates the requirements for short processing latency 2) The computation kernel must have low hardware cost and energy consumption, especially considering the integration of the computation kernel into the miniscope device. 3) The computation hardware has to be reconfigurable, as it may require adjustments of parameters involved in the calcium image processing across days during the experiments.

In prior work, we have proposed an FPGA-based real-time calcium image processing and decoding pipeline [15]–[19]. Here we report several advances and improvements to our real-time calcium image processing system, including improvements to the trace extraction accelerator and the introduction of a variety of neural network based calcium image decoding methods. In addition, we introduced the implementation of the neural network based decoders on the FPGA. We showed that the FPGA implementation can achieve significant speedup over the implementation on the embedded ARM processor and enable real-time calcium image decoding with sub-ms latency.

The paper is arranged as follows: In Section II, we provide a brief review of our previously proposed real-time calcium image processing pipeline. In Section III, we present design and optimization details of the trace extraction accelerator. In Section IV, we propose different neural network based methods and accelerator designs for the calcium image decoding, and evaluate the tradeoff among these decoding accelerators. Section V presents the implementation of the FPGA-based closed-loop calcium image processing pipeline. Section VI summarizes the performance of this work and discusses related works in the field. Section VII draws the conclusion.

II. REVIEW OF REAL-TIME PROCESSING PIPELINE

In the past years, we proposed and developed the DeCalciOn – an FPGA-based real-time calcium image processing pipeline for closed-loop feedback applications [19]. Fig. 2 shows the overall architecture of the processing pipeline. It includes customized accelerators for the motion correction, the image enhancement, the trace extraction, and the decoding. Fig. 3 shows the timing diagram of the DeCalciOn pipeline. The proposed accelerators operate in a streaming fashion and do not require off-chip memory access throughout the processing stages. As the complete pipeline and the accelerators for the motion correction and image enhancement have been introduced in [15], [16], [19], in this work we mainly focus on the implementation detail and the optimization for the trace extraction accelerator ACC-Trace and the decoding accelerator ACC-Decode.

III. ACC-TRACE: FAST TRACE EXTRACTION

In this session, we further elaborate the trace extraction accelerator: ACC-Trace, and discuss the optimization on fast trace extraction for both the cell-based and tile-based contours.

A. ACC-Trace Accelerator

Our proposed ACC-Trace accelerator has the 1-D systolic array architecture, as Fig. 4 shows. It consists of a chain of J tracing elements (TE). Each TE contains the trace extraction logic and a local contour buffer, and it takes in and shifts an 8-bit pixel vi with its corresponding 9-bit row and column indices (r_i, c_i) per clock cycle. The local buffer can store up to K pieces of cell contour information for the trace extraction. For a cell Cell (j, k) mapped to the kth piece of the jth TE, we store not only an $N_C \times N_C$ binary mask $Q_{j,k}$, but also the cell center location $(R_{j,k}, C_{j,k})$. Suppose the calcium image has N pixels, the 16-bit trace value f_{jk} for the Cell (j, k) can be derived by:

$$f_{j,k} = \sum_{i=0}^{N} \sum_{dr_{ijk}=0}^{N_C} \sum_{dc_{ijk}=0}^{N_C} v_i \cdot Q_{j,k}(dr_{ijk}, dc_{ijk})\,. \tag{1}$$

The (dr_{ijk}, dc_{ijk}) are derived indices from the scanned indices (r_i, c_i) and the cell center location $(R_{j,k}, C_{j,k})$:

$$\begin{cases} dr_{ijk} = r_i - R_{j,k} + N_C/2 \\ dc_{ijk} = c_i - C_{j,k} + N_C/2 \end{cases}. \tag{2}$$

The ACC-Trace accelerator operates under 3 modes. 1) Load: Cell contour and center location information are loaded to the local cell contour buffer and local register files in a bit-shift fashion. 2) Compute: Calcium image is scanned pixel by pixel through the chain of TEs, whereas the TEs work in parallel to monitor the scanned pixels and accumulate trace values based on the contour information fetched correspondingly for each pair of incoming row and column indices (r_i, c_i). 3) Store: The accumulated trace values f_{jk} are sent to an external buffer in the same bit-shift fashion. Though the ACC-Trace design is very scalable, oftentimes we are not able to build a sufficiently long chain of TEs to handle the trace extraction tasks for all target cells due to limited FPGA resources at hand. In this case, we build a relatively shorter ACC-Trace chain and reuse it for multiple iterations of operation, and reload the cell contour and center information between iterations.

B. Cell-Based Contour Allocation

In the ACC-Trace accelerator design, each TE stores K contours in its local buffer. We implement the local buffer by a single BRAM slice on the FPGA, and use only one read port for contour access during the trace extraction given the limited number of ports provided by the BRAM. This will raise a memory access conflict when the TE tries to fetch data from two different contours at the same clock cycle. In order to avoid the memory access conflict, we need to guarantee that there is no overlap among K stored contours in every TE on the algorithm level.

We propose a cell contour allocation algorithm to make sure the cells assigned to a same TE do not overlap, as shown in Fig. 5. Initially, we allocate $J \times K$ cell contours to J TEs with a default order. Then we loop through each contour and check if the current contour of Cell (j, k) in a TE has overlap with other contours allocated in the same TE. If it has an overlap with another contour, then it means that there will be a memory access conflict at this contour, and the algorithm will go to a solve_conflict routine to adjust the allocation for this contour. In the solve_conflict routine, the algorithm will iteratively pick up a random contour from another TE, swap the conflict contour of Cell (j_0, k_0) and the current contour of Cell (j, k), and check if there is any overlap between these two contours and the reset of contours in the same TE after swapping. If there is no overlap, the algorithm will accept the swap as a solution for the conflict and update the contour allocation correspondingly. Otherwise, the algorithm will go on to pick up another random contour and go through the same checking described by line 20–24 in Algorithm 1 shown in Fig. 5 until it finds an acceptable solution.

C. Tile-Based Contour Allocation

Fig. 6(a) and (b) illustrate the concept of the tile-based contours. As the image size is 512×512, we can divide them evenly into $N \times N (N = 32)$ non-overlapping $N_T \times N_T (N_T = 16)$ tiles as shown in Fig. 6(a), and each tile can be viewed as a $N_C \times N_C (N_C = 25)$ tile-based contour. Inside each tile-based contour, we set the elements at the central 16 × 16 region to be "1", while keeping the rest of elements to be "0". By taking advantage of the tile-based contours for the calcium trace extraction, we can eliminate the need of running offline analysis pipeline, such as CaImAn [4], for cell contour detection. The offline analysis pipeline usually takes tens of minutes or even longer to finish on a pre-recorded session of calcium images depending on the computation platform the pipeline runs on. With tile-based contours, we can bypass the cell contour detection step and directly extract calcium traces, and it contributes to a short turnaround time for decoding model training and a quick deployment of decoder onto the hardware for closed-loop feedback applications [19]. In Section IV, we will provide more evaluation results regarding the comparison between the decoding based on the tile-based and cell-based contours.

Based on the ACC-Trace accelerator microarchitecture introduced in Section 3, a direct mapping of the 32 × 32 tile-based contours to the $J = 32$ accelerator will cause memory access conflict, as the neighboring tile-based contours mapped to the same TE have overlaps with each other. As Fig. 6(c) shows, we come up with a way of mapping of tile-based contours in order to avoid memory access conflicts. Here, 8 tile-based contours highlighted by 25 × 25 size squares are shown with the same shaded color and mapped to the same TE, and following tile-based contours in these rows are mapped to following TEs, so on and so forth. In this way, the tile-based contours mapped to every single TE do not have any overlap, so the memory access conflicts are avoided during the scanning of pixels in the trace extraction computation.

D. Latency Optimizations

As Fig. 3 shows, the runtime of the trace extraction step largely contributes to the overall processing and decoding latency. In this subsection, we introduce three optimizations that can speed up the trace extraction and reduce the overall latency, as shown in Fig. 7.

Fig. 7(a) presents the first latency optimization: region segmentation. This optimization takes effect when the ACC-Trace accelerator needs to be reused for multiple times during the trace extraction of a single frame. If we do not consider the region segmentation, the contours that need to be processed at each round are distributed evenly across the image. In that case, the whole frame of the image needs to be scanned in order to complete the trace extraction task. On the contrary, if we constrain the contours by their locations when allocating them to different rounds of trace extraction, we only need to scan a part of the image at each round. In this way, we can largely save runtime without increasing the overhead on the hardware implementation.

Within one iteration, we find that there is more opportunity for the latency optimization, as Fig. 7(b) shows. While a portion of pixels in the image are in the background and do not have a cover of the cell footprint, scanning these pixels simply adds up runtime and but does not make an effect on the trace extraction results. The main idea of our second latency optimization is to skip over those pixels in the scanning. Through an offline analysis of the locations of the cell contours, we can figure out starting and ending position indices of each consecutive segment of background pixels. We store these position indices in a local BRAM buffer aside the ACC-Trace accelerator and design digital circuits that keep monitoring the indices of scanned pixels and take action of the fast forward when a background pixel is detected.

The third latency optimization is the double buffering as shown in Fig. 7(c). Suppose we design an ACC-Trace accelerator with J TEs. By default, the load, compute and store operate in sequential order as the same data and indices transmission paths are shared among operations. If we consider splitting the accelerator chain into two parts, we can double buffer the operation of these two parts, and hide the load and store time behind the compute operation without incurring overhead on the hardware implementation. In this way, additional speed-up can be achieved for further reduced latency.

Taking a combination of the proposed latency optimizations, the ACC-Trace accelerator can reduce the runtime from 3.5 ms to 589 μs on an evaluation set with 760 cell contours under a 300 MHz clock frequency, whereas the increased LUT, FF and BRAM resource usage is within 1.5%.

IV. ACC-DECODE: EFFICIENT DECODING

In this section, we first review the position decoding task [19], and then discuss the proposed neural network based decoding methods and accelerator designs, which take input from the extracted traces by the ACC-Trace accelerator.

A. Position Decoding Task

The task is to decode rat's positions on a 2.5-m linear track as it runs back and forth [18], [19]. The linear track is evenly divided into 24 bins, labeled with discrete numbers 0, 1, 2, ..., 23. Since the real-time decoding aims at closed-loop feedback applications, it is required to be causal, which means that the decoding does not rely on future calcium images as input.

B. Accuracy Evaluation Metrics

We came up with two accuracy evaluation metrics for the position decoding task: 1) *Hit-N* accuracy measurement, and 2) mean error measurement, as elaborated below.

1. *1) Hit-N Accuracy:* The *Hit-N* accuracy counts the percentage of frames for which the decoded position falls into the *N*-sample neighborhood around the true position. For example, the *Hit-1* accuracy measures the percentage of frames for which the decoded position is exactly right, whereas the *Hit-3* accuracy measures the percentage of frames for which the decoded position falls into the ±1-bin neighborhood around the true position.

2. *2) Mean Error:* The mean error σ measures the absolute difference between the decoded position and the true position on average for each frame. It reflects the decoding accuracy in general and can be used as a fair comparison metric across different decoding methods on the same dataset.

C. Decoding Methods

We first propose the CNN and the SNN based decoding methods for the position decoding. Fig. 8 shows the proposed CNN and SNN models, and they share the same model architecture that we decide to use based on the algorithm exploration on the decoding task [18]. For the calcium image decoding, we first extract all of the cells from the calcium images in the training set offline using the CaImAn method [4]. Then we identify the first set of $N_P \times N_P$ cells with a largest possible $N_P \times N_P$ from the extracted cells based on a downwards peak trace value sorting [15], normalize the trace values collected from these cells to [0,1], and form an image as the CNN and SNN inputs. For the CNN model, the input image is first convolved with 6 filter kernels in 3×3 size, then the feature maps spread out to an $(N_P - 2) \times (N_P - 2) \times 6$ element vector. A hidden layer establishes all-to-all connections from these elements to 24 output nodes, and each output node corresponds to a specific position bin on the linear track. Finally, the accumulated values at all output nodes are compared, and the node that has the maximum value is identified and the corresponding position label is generated as the decoding result.

Our proposed SNN decoder adopts a rate-based encoding, and it is converted from the CNN by the SNN Toolbox [20]. The conversion relies on an integrate-and-fire (IAF) neuron model. As Fig. 8 illustrates, the input value is accumulated to a potential at each hidden node at every time step. Whenever the potential passes a threshold V_t, the hidden node (as a neuron) produces an output of "1" (as a spike) at a specific time step, and a value of V_t is subtracted from the potential. The generated spikes at the hidden layer propagate to the output layer. The output nodes are also modeled as IAF neurons. Each output node takes input spikes from $(N_P - 2) \times (N_P - 2) \times 6$ IAF neurons in the hidden layer, and produces output spikes under the same threshold V_t. Whenever an output node receives an input spike from a specific cell, a constant weight bound with the connection is accumulated in the same fashion as the hidden node. Finally, the number of spikes generated by each output node is counted over the number of time steps TS for each SNN inference. The output node that produces the maximum number of spikes is identified and its index serves as the decoding outcome.

We then propose a simple two-hidden-layer ANN model for the same position decoding task. Each hidden layer contains 32 nodes. We explored two different encodings for the output layer: 1) Categorical encoding, which contains 24 nodes with each node representing a specific position on the linear track; 2) Ordinal encoding, which contains 12 nodes. The inference algorithm converts the 12 output node values into a 12-bit binary code by setting a threshold of 0.5 at each output node. The number of consecutive 1s or 0s from the left of the 12-bit binary code indicates the decoded position bin index. We evaluated effects of 1) the decoding option: tile-based/cell-based decoding and 2) the encoding method: categorical/ordinal encoding on the decoding accuracy across datasets collected from 6 different rats.

D. ACC-Decode: Efficient Decoder

We evaluated the *Hit-1*/*Hit-3* accuracy achieved by a baseline floating-point CNN and a converted 8-bit SNN with $T_S = 8$ time steps on calcium image decoding test sets across 6 different rats. Table I presents an overview of these test sets. Among 6 rats $R_1 - R_8$, there were 153–760 cells detected from their corresponding training sets. According to Section IV-B, we assigned N_P with the value of 13–27 for these rats. We collected 8000 frames of calcium images with corresponding tracking positions for each rat, and divided each recording session into two parts for training and testing separately. *Hit-1*/*Hit-3* accuracies on the test sets for the CNN and the SNN models are evaluated and summarized in Fig. 9. On average, the CNN and SNN models achieve 56.3%/83.1% and 56.0%/82.8% *Hit-1*/*Hit-3* accuracy, correspondingly. It can be observed that the 8-bit SNN with 32 time steps has almost no accuracy loss compared to the floating-point CNN in performing the decoding task across rats.

We further apply fixed-point quantization and time-step reduction for the SNN model. Fig. 10 shows the accuracy evaluation results under various weight quantization and time steps. We first keep the time step constant and gradually apply more aggressive bit-width quantization. From Fig. 10(a), we see that the SNN model under 6-bit quantization on average has 5.58%/3.62% loss on the *Hit-1*/*Hit-3* accuracy. Then we apply the time-step reduction for the 6-bit SNN model, and according to Fig. 10(b), we can reduce time steps of the SNN from 32 to 16, while incurring less than 1% *Hit-1* and *Hit-3* accuracy loss. Table II shows the comparison on FPGA resource usage among different decoder models. We set the input size to be the same at 16 × 16 for all these models. The 6-bit SNN consumes less LUT, FF, and BRAM, and completely avoids the usage of DSP compared to the 8-bit CNN model. The ANN model consumes higher LUT and FF usage, but saves the BRAM resource against the CNN model. Under 300 MHz clock frequency, the cycle counts for the CNN, SNN (considering 8 time steps), and ANN models are 65.9k, 278.9k, and 19.9k, respectively.

Table III shows a more comprehensive decoding performance evaluation by the Hit-3 accuracy and the mean error σ on the decoded position for the proposed CNN/ANN based decoding methods under various input (cell-based vs. tile-based) and output (categorical vs ordinal) settings. The evaluation results are derived from the average of 5 independent experimental trials. We make several observations from the evaluation results: 1) the tile-based decoding generally outperforms the cell-based decoding, which is consistent with the finding in [19]. This can be explained by the fact that more tile-based contours (900 vs. 169–729) are involved in the decoding. 2) The ANN based method slightly outperforms the CNN counterpart. That's because the extracted trace data does not contain visual features that the CNN can leverage. On the contrary, the ANN can take the advantage of its global connections for more efficient and accurate decoding. 3) The ordinal encoding in general outperforms the categorical encoding. That's because the decoded positions are continuous, and the ordinal encoding inherently suits this characteristic.

We designed CNN/ANN based decoding accelerator kernels with the Vitis HLS for the cell-based scenario. Table IV shows hardware resource utilization from the post implementation report. Compared to the CNN based design, the ANN based design consumes 30–35% more LUT and FF, and much less BRAM resources. Table V reports the runtime evaluated on the CNN/ANN based decoding kernels. As we targeted 300 MHz for the decoding kernels, the runtime for the CNN-based decoding kernel ranges from 155–800 μs, while the ANN-based decoding kernel takes much shorter runtime by a factor of 6.5–9.5x.

V. IMPLEMENTATION

We implemented the proposed calcium image processing pipeline introduced in Session II on the Ultra96 SoC platform. The complete processing pipeline operates at 300 MHz. Table VII reports the overall FPGA resource utilization, and Fig. 11 shows the resource usage breakdown for the implemented accelerators. The motion correction consumes the largest part of computation and memory resources, as it speeds up the most time-consuming and critical pre-processing step, which removes motion artifacts from the raw calcium images and helps increase the *Hit-3* decoding accuracy by 1.42% on average based on our analysis with the CNN-based decoder. The ACC-Trace accelerator also costs considerable LUT, FF and BRAM resources given its fine-grain pipeline architecture. It largely reduces the input dimension and inference time of the decoder whereas it does not increase accuracy loss. The ACC-Decode accelerator consumes less resources than the ACC-Trace accelerator, and the remaining resource is reserved for implementing the feedback control logic on the same FPGA.

We measured the runtime of the proposed neural network based decoders on the embedded ARM processor of the Ultra96 platform under 1 GHz operating frequency. The runtime of the CNN and ANN based decoders with the 27×27 input size are 7.45 ms and 1.42 ms per inference, respectively. The runtime of the SNN based decoder with the 16×16 input size and 8 time steps is 7.65ms per inference. Compared to the ARM based implementation, the FPGA implementation of the CNN, ANN, and SNN based decoders achieve 9.6x, 17.1x, and 8.2x speedup, respectively.

We built a customized hardware interface board that can be plugged on top of the Ultra96 platform, as Fig. 12 shows. The interface board provides the input port for connecting the miniscope sensor, the output port for applying feedback stimulation with TTL pulses, and a standard Ethernet port that can communicate with the host computer for data transfer and user interaction. The input port is connected to the DAQ board with 13 bundled fly wires, which correspond to one 66.67 MHz sensor readout clock, 8 sensor data pins, H/V synchronization signals, and VDD/GND signals. We also connect the DAQ board to the host computer with a USB 3.0 cable to control the miniscope sensor, and connect the miniscope sensor to the DAQ board with a 1.5-m flexible coax cable. The overall system except the host computer is powered by a 12V power supply. The peak power consumption is 5.3W, and the standby power consumption is 2.2W.

We also developed graphical user interface software that runs on the host computer and interacts with the FPGA hardware. The user interface supports two application scenarios: 1) real-time calcium-image trace extraction; 2) real-time calcium image decoding. In the first scenario, the user interface is able to display received calcium images and a maximum of 63 extracted traces in real time. The contours of the selected cells can be superimposed on top of the displayed calcium images, as Fig. 13(a) shows. In the second scenario shown in Fig. 13(b), the user interface displays the tile-based trace values extracted at every frame with the real-time position decoding result. It also shows superimposed hollow contours and a flow of activity heat map for selected place tiles, which are identified offline [19].

VI. EVALUATION

A. Performance Summary

Our work achieves comparable performance against other state-of-the-arts targeting real-time calcium image processing for closed-loop applications. Table VII summarizes a comparison among these works. [19] and [11] focus on real-time behavior decoding, whereas [23] and [24] attempt to address the real-time cell activity pattern detection problem. Compared with others, our work highlights the usage of hardware platform of the FPGA, and for the first time reduces the calcium image decoding latency to less than 1 ms, which is regarded as the biological spike-timing precision and critical timing factor for kilo-frame-rate voltage imaging [14]. Besides, our work first introduces the use of the SNN for the calcium image decoding, which introduces a new application area for the neuromorphic algorithms. Though the proposed rate-based SNN still cannot match the CNN model on the latency and thus the power consumption, it has shown the potential to become a low hardware cost counterpart aside from the CNN. Finally, our implementation not only supports closed-loop feedback, but it also provides an end-to-end tool flow with dedicated hardware and software interfaces for neuroscientists who rely on closed-loop experiments to advance brain research.

B. Comparison to State-of-the-Arts

As miniaturized calcium imaging devices are gaining momentum as critical intervention tools for brain research, we have seen increased interests and related works focusing on calcium image analysis and processing. Based on targeted applications, these works can be mainly classified into 1) calcium imaging based neural signal extraction and 2) calcium image decoding.

Calcium imaging based neural signal extraction combines stabilization, enhancement, and spike inference to extract useful neural signal information from recorded calcium images. [5] proposed a non-rigid motion correction method that is able to effectively compensate for non-ideal motion artifacts at subregions in the calcium image. [25] and [26] introduced the method for identifying cells from calcium images based on the CNMF approach. [5] came up with an extension of the CNMF approach to address the background contamination for one-photon calcium images. [27], [28] and [29] concentrated on spike inference from the calcium imaging data based on fast deconvolution and Bayesian methods. [30] with its follow-up work [4] unified previous efforts and established a complete calcium image analysis flow, which is open sourced with Python [31] and Matlab [32] releases. MIN1PIPE [7] is another popular calcium image analysis pipeline which combines the LSTM inference for the cell identification and trace extraction. BSSE [33] offers a generic calcium image analysis tool for not only brain imaging but also other tissue studies. [34] brought Z-dimension motion registration and visualization to attention for improvement on calcium image analysis. [35] proposed an independent component analysis based method to identify cells and extract neural signals from calcium images. Although these methods have been successful in extracting neural signal information from noisy calcium images, they are usually hard to be realized in real time. [30] and [4] achieve real-time calcium image analysis speed with optimized computation processes, but it’s still hard to realize short processing latency for closed-loop applications. [24] proposed a closed-loop all-optical feedback stimulation framework based on threshold crossing detection of neural fluorescence signals, and it’s dedicated for two-photon calcium imaging on head-fixing mice.

Some works look beyond neural activity and pursue the idea of calcium image decoding. [9] combined a deep residual neural network with the k-Means method to decode the forelimb reach activity from averaged calcium image from mice. [10] applied the Laplacian Eigenmaps to reduce dimension of neural activity extracted from calcium images and decode behavior states of mice on a linear track. [37] studied the relationship between behavioral states and neuronal activity based on the Bayesian classifier. [13] leveraged the MIN1PIPE [7] for preprocessing, the principal component analysis for dimension reduction, and a linear SVM to decode the lever pressing movement of mice. [36] benchmarked 5 different classifiers on a high/low velocity decoding task. Some other works look into tackling the real-time calcium image decoding challenge. [39] showed that calcium traces without spike inference can be directly used for real-time position decoding by supervised and unsupervised methods. [38] introduced incremental linear discriminant analysis method for real-time neural decoding with online adaptation capability. [11] and [12] developed real-time calcium image decoding pipelines based on the SVM method.

Compared to previous works, this work provides an end-to-end real-time calcium image processing and decoding solution for closed-loop feedback applications. Combining the customized algorithm pipeline, the accelerator design and optimization, and the hardware implementation, it achieves deterministic and sub-ms processing latency for closed-loop calcium image processing on the low-cost FPGA platform. It also evaluates and analyzes the tradeoff on accuracy, performance and cost among different neural network based methods for the calcium image decoding task.

VII. CONCLUSION

This paper introduces an end-to-end FPGA-based prototype for real-time calcium image processing and decoding. With the support of dedicated hardware interface, it can perform a series of calcium image processing including the motion correction, enhancement, trace extraction and decoding in a frame-based fashion with sub-ms and deterministic processing latency. It empowers neuroscientists working with the in-vivo calcium imaging to perform efficient and precise closed-loop feedback stimulation for a wide range of neuroscientific experiments, and has potential to contribute to future brain research.

Acknowledgments

This work was supported by NSF NeuroNex DBI-1707408, and NIH MH122800. The authors would like to thank Dr. Changliang Guo and Jeffrey Johnson for their support on the miniscope device and the Ethernet interface development.

REFERENCES

[1]. Ghosh KK, Burns LD, Cocker ED, Nimmerjahn A, Ziv Y, El Gamal A, and et al. , "Miniaturized integration of a fluorescence microscope," Nature Methods, vol. 8, no. 10, pp. 871–878, 2011. [PubMed: 21909102]

[2]. Aharoni D, Khakh BS, Silva AJ, and Golshani P, "All the light that we can see: a new era in miniaturized microscopy," Nature Methods, vol. 16, no. 1, pp. 11–13, 2019. [PubMed: 30573833]

[3]. Guo C, Blair GJ, Sehgal M, Sangiuliano Jimka FN, Bellafard A, Silva AJ, and et al. , "Miniscope-LFOV: A large field of view, single cell resolution, miniature microscope for wired and wire-free imaging of neural dynamics in freely behaving animals," bioRXiv, 2021.11.21.469394, 2021.

[4]. Giovannucci A, Friedrich J, Gunn P, Kalfon J, Brown BL, Koay SA, and et al. , "CaImAn an open source tool for scalable calcium imaging data analysis," eLife, vol. 8, e38173, 2019.

[5]. Pnevmatikakis EA and Giovannucci A, "NoRMCorre: An online algorithm for piecewise rigid motion correction of calcium imaging data," J. Neurosci. Methods, vol. 291, pp. 83–94, 2017. [PubMed: 28782629]

[6]. Zhou P, Resendez SL, Rodriguez-Romaguera J, Jimenez JC, Neufeld SQ, Giovannucci A, and et al. , "Efficient and accurate extraction of in vivo calcium signals from microendoscopic video data," eLife, vol. 7, e28728, 2018.

[7]. Lu J, Li C, Singh-Alvarado J, Zhou ZC, Fröhlich F, Mooney R, and¨ et al. , "MIN1PIPE: a miniscope 1-photon-based calcium imaging signal extraction pipeline," Cell Reports, vol. 23, no. 12, pp. 3673–3684, 2018. [PubMed: 29925007]

[8]. Friedrich J, Giovannucci A, and Pnevmatikakis EA, "Online analysis of microendoscopic 1-photon calcium imaging data streams," PLoS Computational Biology, vol. 17, no.1, e1008565, 2021.

[9]. Li C, Chan D, Yang X, Ke Y, and Yung WH, "Prediction of forelimb reach results from motor cortex activities based on calcium imaging and deep learning," Front. Cell. Neurosci, vol. 13, 2019.

[10]. Rubin A, Sheintuch L, Brande-Eilat N, Pinchasof O, Rechavi Y, Geva N and et al. , "Revealing neural correlates of behavior without behavioral measurements," Nature Commun, vol. 10, no. 1, 4745, 2019. [PubMed: 31628322]

[11]. Liu C, Li M, Wang R, Cui X, Jung H, Halin K and et al. , “Online decoding system with calcium image from mice primary motor cortex,” Int. Conf. IEEE Eng. Medicine Biology Soc. (EMBC), pp. 6402–6405, 2021.

[12]. Lee K, Lee Y, Lin DT, Bhattacharyya SS, and Chen R, “Real-time calcium imaging based neural decoding with a support vector machine,” IEEE Biomed. Circuits Syst. Conf. (BioCAS), 2019.

[13]. Wang R, Han J, Chen J, Li M, Feng L, and Zhang S, “Decoding with calcium signals from layer 2/3 motor cortex during a pressing movement,” Int. Conf. IEEE Eng. Medicine Biology Soc. (EMBC), pp. 3054–3057, 2019.

[14]. Kazemipour A, Novak O, Flickinger D, Marvin JS, Abdelfattah AS, King J, and et al. , “Kilohertz frame-rate two-photon tomography,” Nature Methods, vol. 16, no. 8, pp. 778–786, 2019. [PubMed: 31363222]

[15]. Chen Z, Blair HT, and Cong J, “LANMC: LSTM-assisted non-rigid motion correction on FPGA for calcium image stabilization,” Proc. of the ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays (FPGA), pp. 104–109, 2019.

[16]. Chen Z, Blair GJ, Blair HT, and Cong J, “BLINK: Bit-sparse LSTM inference kernel enabling efficient calcium trace extraction for neurofeedback devices,” Proc. Int. Symp. Low Power Electron. Des. (ISLPED), pp. 217–222, 2020.

[17]. Chen Z, Blair GJ, Blair HT, and Cong J, “Fast calcium trace extraction for large-field-of-view miniscope,” IEEE Biomed. Circuits Syst. Conf. (BioCAS), 2021.

[18]. Chen Z, Zhou J, Blair GJ, Blair HT, and Cong J, “Efficient kernels for real-time position decoding from in vivo calcium images,” Int. Symp. Circuits Syst. (ISCAS), 2022.

[19]. Chen Z, Blair GJ, Guo C, Zhou J, Izquierdo A, Golshani P, and et al. , “DeCalciOn: A hardware system for real-time decoding of in-vivo calcium imaging data,” eLife, in press.

[20]. Rueckauer B, Lungu IA, Hu Y, Pfeiffer M and Liu SC, “Conversion of continuous-valued deep networks to efficient event-driven networks for image classification,” Front. Neurosci, vol. 11, 682, 2017. [PubMed: 29375284]

[21]. Diehl PU, Neil D, Binas J, Cook M, Liu SC and Pfeiffer M, “Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing,” Int. Joint Conf. on Neural Networks (IJCNN), 2015.

[22]. Chen Z, Blair GJ, Guo C, Aharoni D, Blair HT, and Cong J, “Live demonstration: Real-time calcium trace extraction from large-field-of-view miniscope,” IEEE Biomed. Circuits Syst. Conf. (BioCAS), 2021.

[23]. Zhang Z, Russell LE, Packer AM, Gauld OM, and Hausser M,¨ “Closed-loop all-optical interrogation of neural circuits in vivo,” Nature Methods, vol. 15, no. 12, pp. 1037–1040, 2018. [PubMed: 30420686]

[24]. Taniguchi M, Tezuka T, Vergara P, Srinivasan S, Hosokawa T, Cherasse Y and et al. , “Open-source software for real-time calcium´ imaging and synchronized neuron firing detection,” Int. Conf. IEEE Eng. Medicine Biology Soc. (EMBC), pp. 2997–3003, 2021.

[25]. Pnevmatikakis EA, Gao Y, Soudry D, Pfau D, Lacefield C, Poskanzer K, and et al. , “A structured matrix factorization framework for large scale calcium imaging data analysis,” arXiv, 1409.2903, 2014.

[26]. Pnevmatikakis EA, Soudry D, Gao Y, Machado TA, Merel J, Pfau D and et al. , “Simultaneous denoising, deconvolution, and demixing of calcium imaging data,” Neuron, vol. 89, no. 2, pp. 285–299, 2016. [PubMed: 26774160]

[27]. Friedrich J and Paninski L, “Fast active set methods for online spike inference from calcium imaging,” Proc. Advances in Neural Inform. Process. Syst. (NIPS), pp. 1992–2000, 2016.

[28]. Friedrich J, Zhou P, and Paninski L, “Fast online deconvolution of calcium imaging data,” PLoS Computational Biology, vol. 13, no. 3, 2017.

[29]. Pnevmatikakis EA, Merel J, Pakman A, and Paninski L, “Bayesian spike inference from calcium imaging data,” Asilomar Conf. on Signals, Syst. and Comput., pp. 349–353, 2013.

[30]. Giovannucci A, Friedrich J, Kaufman M, Churchland A, Chklovskii D, Paninski L, and et al. , “OnACID: Online analysis of calcium imaging data in real time,” Proc. Advances in Neural Inform. Process. Syst. (NIPS), pp. 2378–2388, 2017.

[31]. https://github.com/flatironinstitute/CaImAn

[32]. https://github.com/flatironinstitute/CaImAn-MATLAB

[33]. Son J, Mandracchia B, Caponegro MD, Tsirka SE, and Jia S, "BSSE: An open-source image processing tool for miniaturized microscopy," Optics Express, vol. 27, no. 13, 17620, 2019.

[34]. Stringer C and Pachitariu M, "Computational processing of neural recordings from calcium imaging data," Current Opinion in Neurobiology, vol. 55, pp. 22–31, 2019. [PubMed: 30530255]

[35]. Levin-Schwartz Y, Sparta DR, Cheer JF, and Adali A, "Parameter-free automated extraction of neuronal signals from calcium imaging data," Proc. IEEE Int. Conf. Acoust., Speech and Signal Process. (ICASSP), pp. 1033–1037, 2017.

[36]. Chen R and Lin D, "Decoding brain states based on microcircuits," Proc. IEEE Int. Conf. Cyborg and Bionic Syst. (CBS), pp. 397–400, 2018.

[37]. Etter G, Manseau F, and Williams S, "A probabilistic framework for decoding behavior from in vivo calcium imaging data," Front. Neural Circuits, vol. 14, 19, 2020.

[38]. Lee Y, Madayambath SC, Liu Y, Lin D, Chen R, and Bhattacharyya SS, "Online learning in neural decoding using incremental linear discriminant analysis," Proc. IEEE Int. Conf. Cyborg and Bionic Syst. (CBS), pp. 173–177, 2017.

[39]. Tu M, Zhao R, Adler A, Gan W, and Chen ZS, "Efficient position decoding methods based on fluorescence calcium imaging in the mouse hippocampus," Neural Computat, vol. 32, no. 6, pp. 1144–1167, 2020.

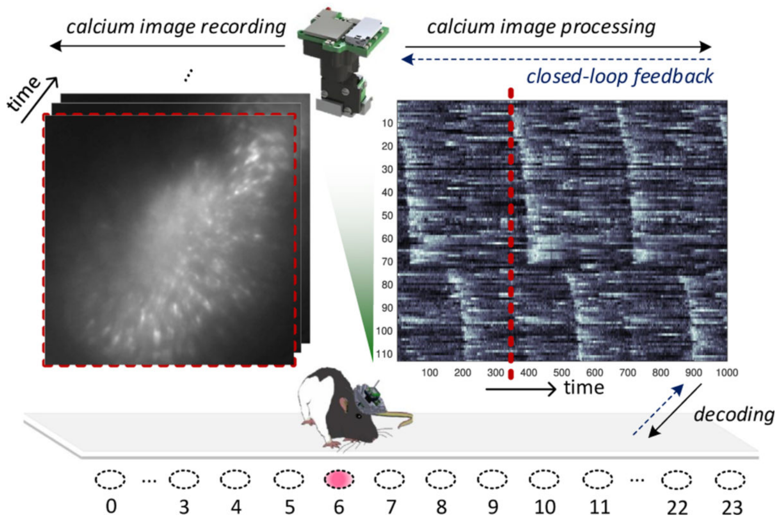

Fig. 1.

Closed-loop feedback capability enabled by real-time calcium image processing and decoding for miniscope devices: As the rat runs back and forth on a linear track, the computation kernel performs calcium image processing and decoding at each frame.

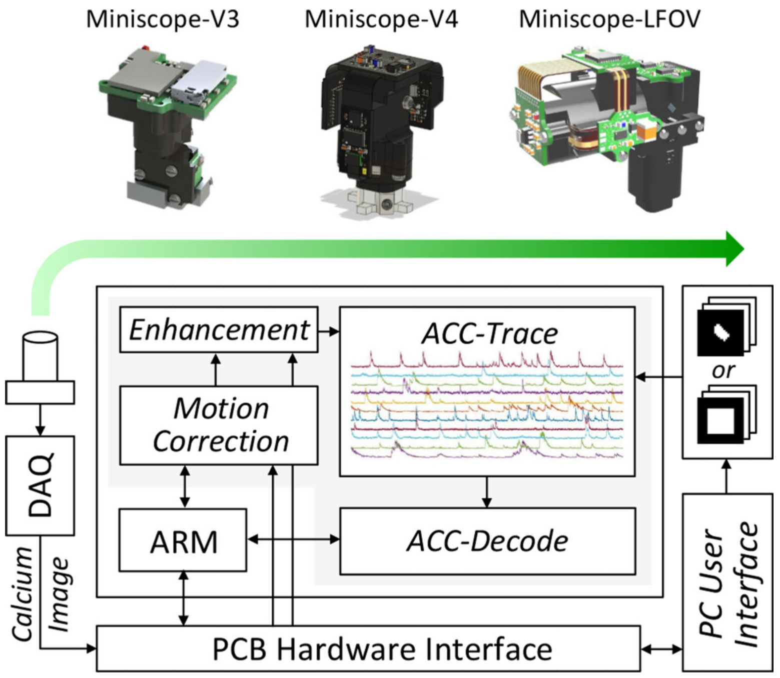

Fig. 2.
Review of DeCalciOn: The first FPGA-based real-time calcium image processing pipeline [19].

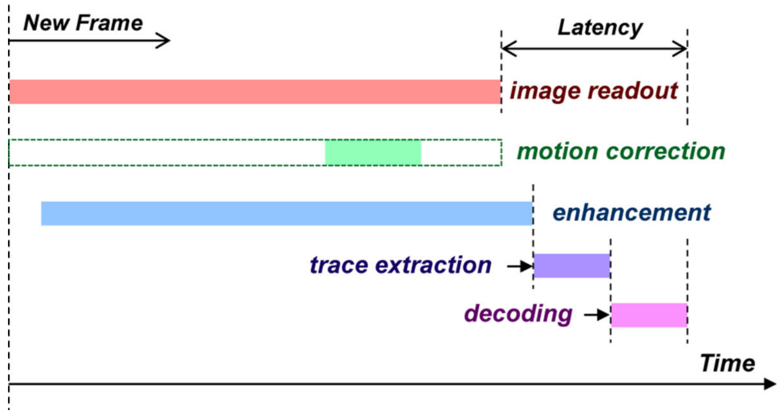

Fig. 3.
Timing diagram for the calcium image processing pipeline.

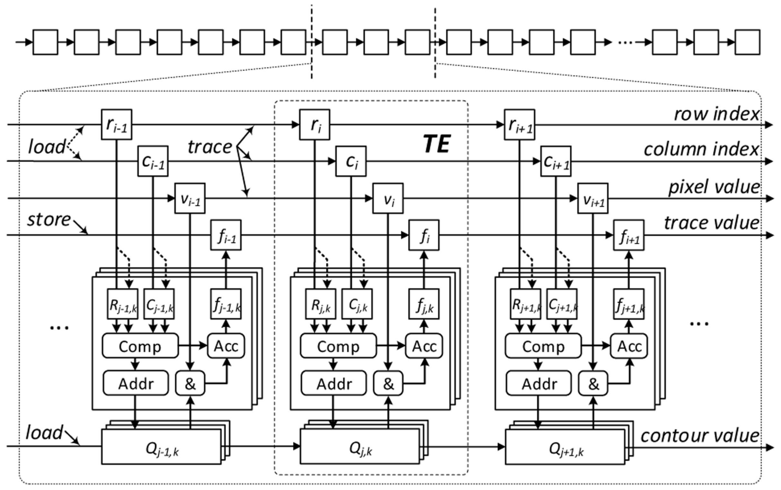

Fig. 4.
ACC-Trace: Efficient calcium trace extraction with 1-D systolic array architecture [17].

Algorithm 1 Contour Allocation

```
1:  procedure
2:      $cell \leftarrow default\ cell\ contours$
3:      $alloc \leftarrow default\ allocation$
4:  $loop$:
5:      for $j \leftarrow 1$ to $J$ do
6:          for $k \leftarrow 1$ to $K$ do
7:              if $is\_conflict(cell(j, k), alloc)$ then
8:                  goto $solve\_conflict(cell(j, k), alloc)$
9:      goto $end$
10: $is\_conflict(cell(j_0, k_0), alloc)$:
11:     $conflict \leftarrow 0$
12:     for $k \leftarrow 1$ to $K$ $(k \neq k_0)$ do
13:         if $is\_overlap(cell(j_0, k_0), cell(j_0, k))$ then
14:             $conlict \leftarrow 1$
15:     return $conflict$
16: $solve\_conflict(cell(j_0, k_0), alloc)$:
17:     $conflict \leftarrow 1$
18:     while $conflict$ do
19:         $j, k \leftarrow random\ select$
20:         $swap(cell(j_0, k_0), cell(j, k))$
21:         $new\_alloc \leftarrow update\ allocation$
22:         if $is\_conflict(cell(j_0, k_0), new\_alloc) = 0$ &&
              $is\_conflict(cell(j, k), new\_alloc) = 0$ then
23:             $conflict \leftarrow 0$
24:             $alloc \leftarrow new\_alloc$
25: $end$: return $alloc$;
```

Fig. 5.

Algorithm for allocating contours to the ACC-Trace accelerator.

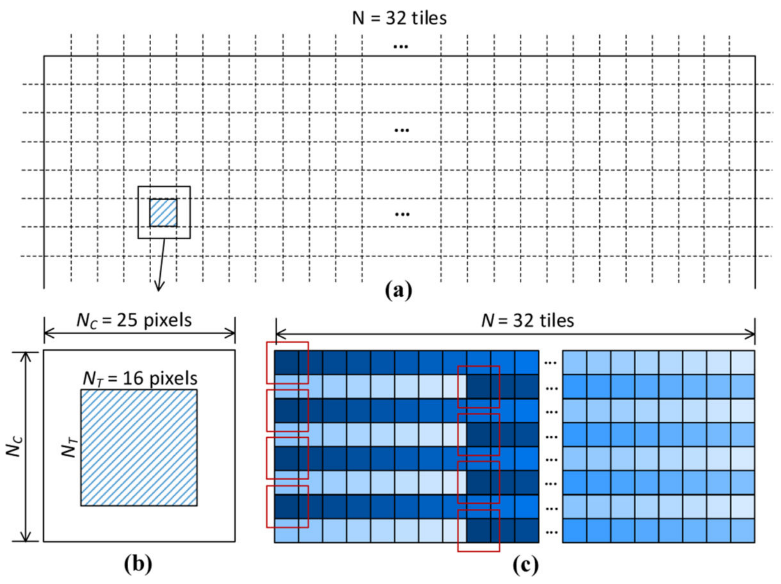

Fig. 6.

(a) Tile based contour and (b) allocation of tile-based contours to the ACC-Trace accelerator.

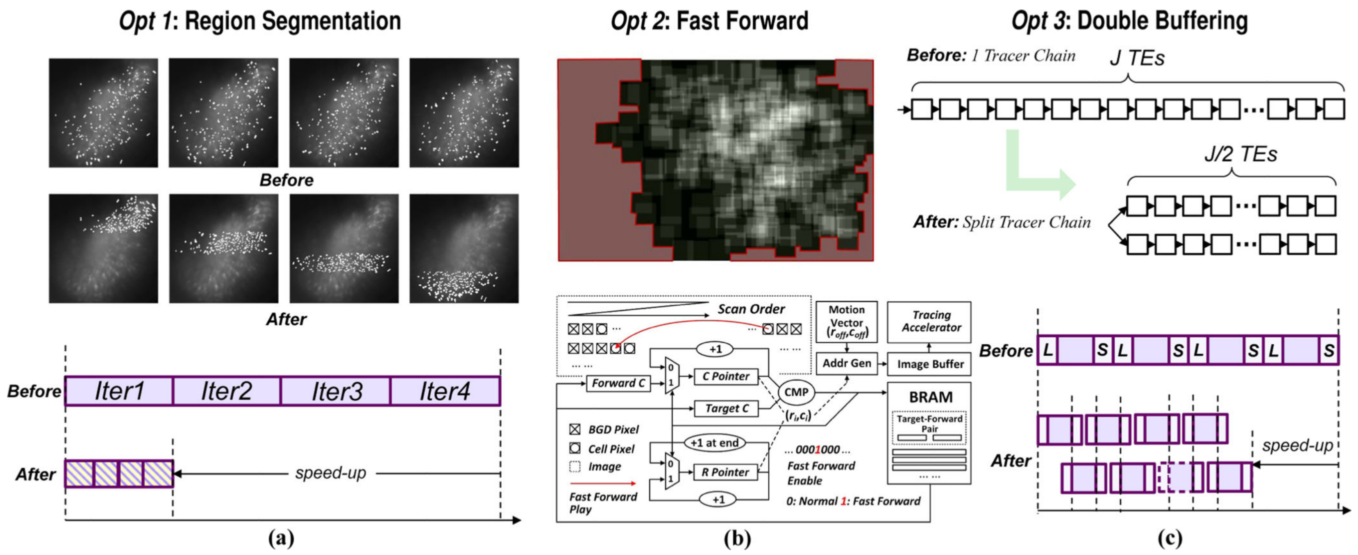

Fig. 7.

Summary of proposed latency optimizations for the ACC-Trace accelerator: (a) Region segmentation; (b) Fast forward; (c) Double buffering.

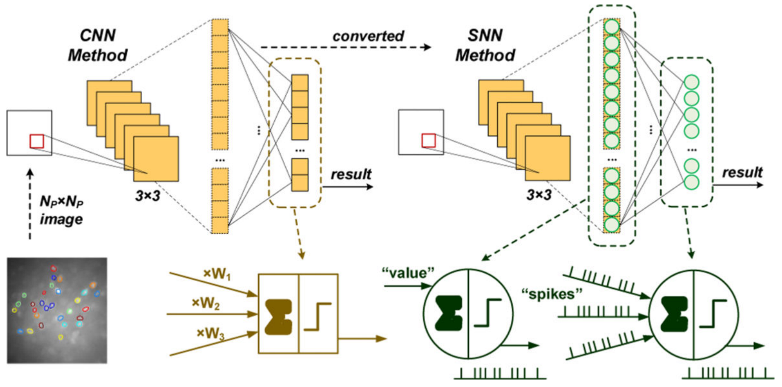

Fig. 8.

Proposed CNN and SNN based methods for position decoding from calcium images.

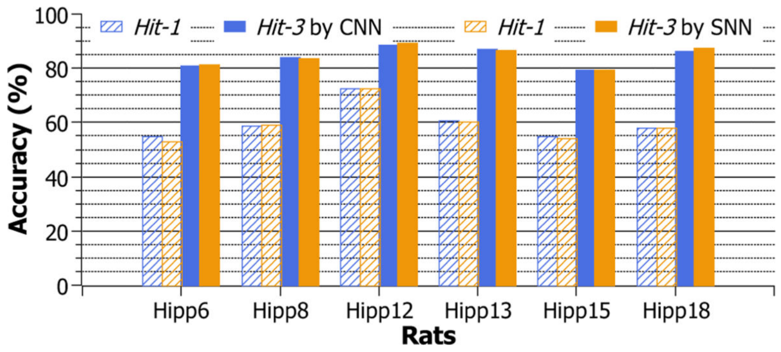

Fig. 9.

Hit-1/*Hit-3* accuracy evaluated for CNN and converted SNN decoders on 6 different rats.

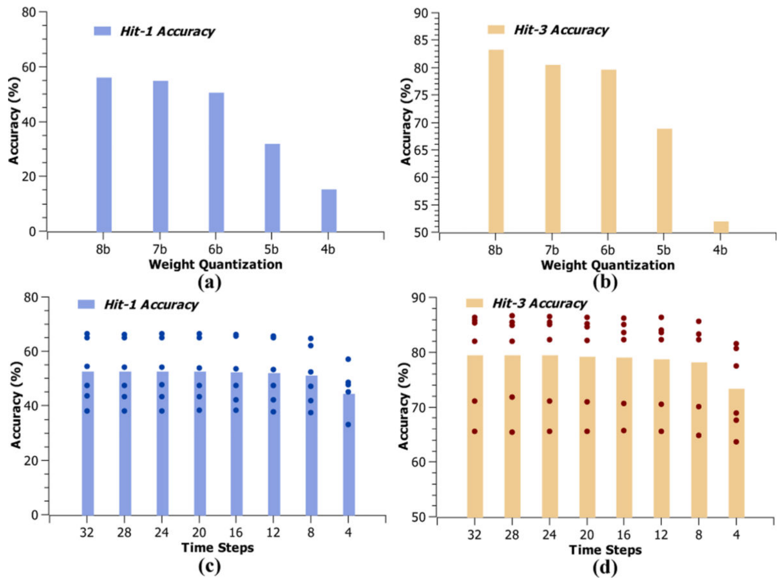

Fig. 10.

Accuracy evaluation for the SNN based method under various quantization of weights and inference time steps: (a) and (b) show the *Hit-1*/*Hit-3* accuracy for a 32 time-step SNN under different bit-widths of weights. (c) and (d) show the *Hit-1*/*Hit-3* accuracy for a 6-bit SNN with different inference time steps.

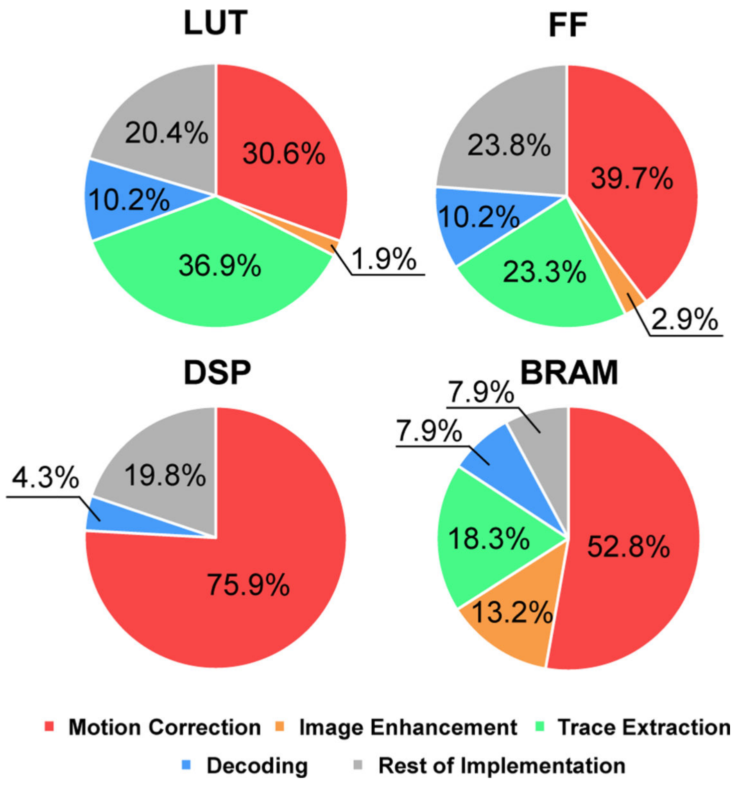

Fig. 11.

Breakdown of the FPGA resource usage of the implementation for the real-time calcium image processing and decoding.

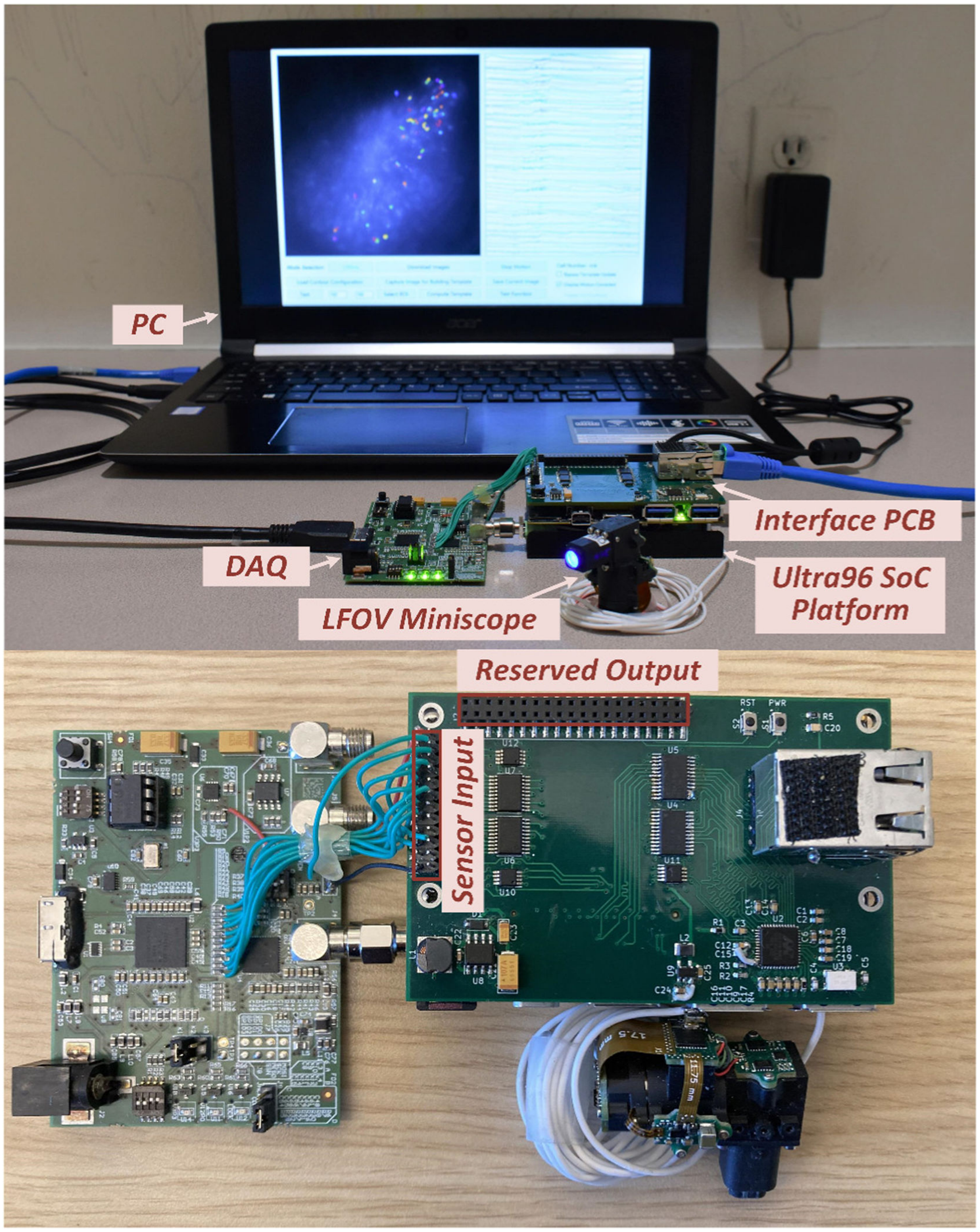

Fig. 12.
The hardware interface and the demonstration setup for the closed-loop feedback system.

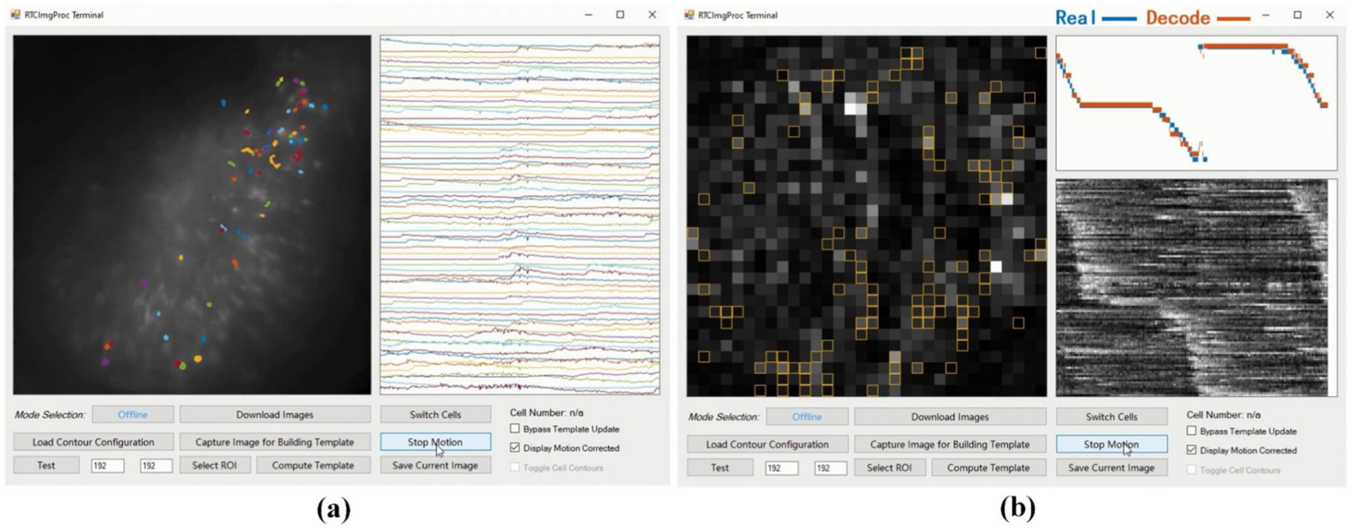

Fig. 13.
Software interface for a demonstration on real-time (a) calcium image trace extraction and (b) calcium image position decoding.

TABLE I

THE NUMBERS OF CELLS IDENTIFIED AND SELECTED FOR DECODING

Rat's ID	Hipp6	Hipp8	Hipp12	Hipp13	Hipp15	Hipp18
# Cells	296	309	317	194	760	643
$N_P \times N_P$	16×16	17×17	17×17	13×13	27×27	25×25

TABLE II

FPGA Resource Usage by Decoding Accelerator Kernels

	CNN-Based	**SNN-Based**	**ANN-Based**
LUT	2713	1484	3356
FF	2454	1440	3133
DSP	7	0	7
BRAM	18	14	6

TABLE III

Accuracy Evaluation for CNN/ANN Based Decoding Methods

Metric	***Hit-3* Accuracy (%)**						**Mean Error σ**					
Input	**Cell-Based**			**Tile-Based**			**Cell-Based**			**Tile-Based**		
Output	**Categorical**		**Ordinal**	**Categorical**		**Ordinal**	**Categorical**		**Ordinal**	**Categorical**		**Ordinal**
Model	**CNN**	**ANN**		**CNN**	**ANN**		**CNN**	**ANN**		**CNN**	**ANN**	
Hipp6	82.69	87.93	89.88	86.72	90.66	**92.39**	1.800	0.736	0.618	0.888	0.563	**0.508**
Hipp8	82.87	86.89	88.99	88.49	90.40	**91.39**	1.091	0.723	0.621	0.597	0.579	**0.553**
Hipp12	88.97	95.37	95.29	94.07	96.73	**97.30**	0.667	0.291	0.285	0.386	0.236	**0.226**
Hipp13	86.28	96.30	94.97	92.07	**98.55**	96.64	0.877	0.338	0.423	0.460	**0.252**	0.593
Hipp15	78.69	83.47	82.84	78.10	**87.43**	87.41	1.710	0.924	0.890	2.220	**0.782**	0.846
Hipp18	86.67	80.89	83.71	**86.83**	79.68	86.26	1.181	1.167	**0.779**	1.081	1.352	0.911

TABLE IV

Hardware Resource Usage for Cell-Based CNN/ANN Based Decoding Accelerator Kernels

Resource	**LUT**			**FF**			**DSP**			**BRAM**		
Output	**Categorical**		**Ordinal**	**Categorical**		**Ordinal**	**Categorical**		**Ordinal**	**Categorical**		**Ordinal**
Model	**CNN**	**ANN**		**CNN**	**ANN**		**CNN**	**ANN**		**CNN**	**ANN**	
Hipp6	2,713	3,394	3,472	2,454	3,236	3,297	7	7	7	18	6	6
Hipp8/12	2,525	3,411	3,529	2,460	3,250	3,311	7	7	7	19	8	8
Hipp13	2,589	3,390	3,475	2,427	3,239	3,298	16	11	11	12	6	6
Hipp15	2,526	3,445	3,523	2,492	3,270	3,333	16	11	11	54	14	14
Hipp18	2,536	3,415	3,522	2,492	3,276	3,341	16	11	11	54	12	12

TABLE V

Runtime for CNN/ANN Based Decoding Accelerator Kernels

Metric	**Cycle Count**			**Runtime (μs)**		
Output	**Categorical**		**Ordinal**	**Categorical**		**Ordinal**
Model	**CNN**	**ANN**		**CNN**	**ANN**	
Hipp6	65,936	10,196	9,840	219.8	34.0	32.8
Hipp8/12	77,030	11,253	10,897	256.8	37.5	36.3
Hipp13	46,553	7,417	7,061	155.2	24.7	23.5
Hipp15	240,089	25,337	24,981	800.3	84.5	83.3
Hipp18	203,225	22,009	21,653	677.4	73.4	72.2

TABLE VI

The Overall FPGA Resource Utilization.

Resource	Utilization	Availability	Utilization %
LUT	52301	70560	74.12%
FF	60490	141120	42.86%
DSP	116	360	32.22%
BRAM	216	216	100%

TABLE VII

COMPARISON WITH STATE-OF-THE-ART CALCIUM IMAGE PROCESSING AND DECODING IMPLEMENTATIONS.

	This Work	**OBCIs [11]**	**[23]**	**Carignan [24]**
Calcium Imaging	One-photon	One-photon	Two-photon	One-photon
Spatial Resolution	512×512	752×480	512×512	480×480
Frame Rate	22.8–60 fps	10–30 fps	30 fps	10 fps
Decode/Detect Task	Decode Position	Lever-pressing	Event Detection	Cell Detection
Decode/Detect Metdod	CNN/SNN/ANN	SVM	Tdreshold	CNN+LSTM
Contour Based Metdod	Yes	Yes	No	Yes
Num. of Cells/Tiles	Up to 1024	10	3	345–621
Decode/Detect Accuracy *	91.9% (*Hit-3*)	81.04%	90%	89.4%
Hardware Platform	Ultra96 FPGA	CPU	CPU, GPU	CPU
Processing Latency	<1 ms	2.417 ms	7.01 ms	60.6 ms
End-to-End Solution	Yes	No	Yes	No
Closed-Loop Support	Yes	No	Yes	Yes

* The accuracy evaluation metric is not the same across different works.