Temporal Frame Filtering with Near-Pixel Compute for Autonomous Driving

Wantong Li[1], Qiucheng Wu[2], Janak Sharda[1], Shiyu Chang[2], Shimeng Yu[1]
[1]*Georgia Institute of Technology, Atlanta, USA;* [2]*University of California Santa Barbara, Santa Barbara, USA*
shimeng.yu@ece.gatech.edu

***Abstract*—As the computer vision techniques behind the advances in autonomous driving become more complex, the on-vehicle processors experience increased burden due to the need to compute on the received images in real-time. Near-pixel compute is an emerging paradigm that brings part of the critical compute tasks very close to the image sensors, and the transmitted data is reduced in dimensionality or partially processed to alleviate the backend compute stress. In this work, we present a near-pixel compute architecture targeting a temporal frame filtering network which drops redundant image frames for the autonomous driving BDD100K dataset. Several circuit optimizations to the compute units are introduced, including using NOR gate array as data buffers, front-end demosaicing, sparsity-aware adder tree, and indirect calculation of image frame difference. The proposed design is synthesized at 40 nm, a node compatible with CMOS image sensor (CIS) technology, which completes the filtering algorithm in 2.2 ms with an energy efficiency of 15.7 TOPS/W and compute density of 380.1 GOPS/mm^2.**

Keywords—near-pixel compute, autonomous driving, hardware accelerator, convolutional network networks

I. Introduction

Recent advances in deep learning techniques have allowed potential deployment of autonomous vehicles onto the road. While the majority of driving scenarios can be solved by classical perception, planning, and control techniques, progresses in tackling more sophisticated problems such as multi-object detection and tracking are largely attributed to state-of-the-art computer vision and deep neural networks (DNNs) [1]. One popular sensor choice that modern autonomous vehicles use to perform these tasks is the camera, thanks to its low cost and easy installation [2]. On the hardware front, prior works [3] have demonstrated dedicated processors that accelerate the heavy workloads of object detection and tracking as the image frames continuously arrive from the camera's image sensor. One emerging compute paradigm to improve the performance of systems using image sensors is in/near-pixel compute [4]. This paradigm involves performing critical compute tasks inside or very close to the camera pixel array. Therefore, instead of entirely transmitting every raw frame to downstream processing, frames of reduced dimensionality or partially processed frames are transmitted to reduce bandwidth and ease the backend compute burden [5].

In this work, we demonstrate a near-pixel temporal frame filtering (TFF) system enabled by software-hardware co-design. Fig. 1 illustrates that the main motivation behind this near-pixel design is to filter out redundant frames early in the pipeline, while maintaining the properties of each transmitted frame. This approach not only helps to lower the power consumption of backend processors due to essentially reduced frame rate, but also guarantees that a variety of tasks can effectively operate on the transmitted raw frames. A convolutional neural network (CNN) is designed to examine consecutive frames from the image sensor for similarity search. We also design a near-pixel accelerator with the novel features of 1) NOR gate array as both multipliers and buffers between network layers, 2) support for front-end demosaicing, 3) sparsity-aware adder tree, and 4) indirect calculation of image frame difference. The accelerator components are implemented and simulated in TSMC 40 nm LP process, where CIS is integrable. Both the activations and weights in the TFF network are quantized to 8-bit integers to be compatible with our near-pixel accelerator, and the effect of frame filtering on the multi-object detection and tracking performances is evaluated on the BDD100K dataset [6].

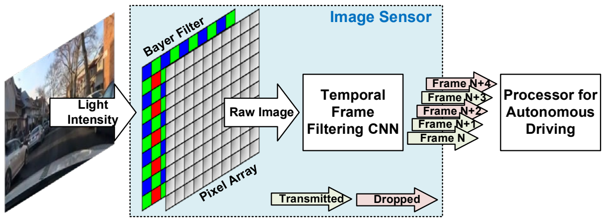


Fig. 1. Schematic of the imaging pipeline, where the near-pixel frame filtering CNN determines whether redundant frames are transmitted or dropped.

II. Background

A. Target frame filtering network

Fig. 2 describes our target TFF network. The input to the TFF network has a dimension of height (H) × width (W) × 6. The channel depth of 6 comes from concatenating the current frame and the difference between current and previous frame, each of which contributes 3 colored channels in RGB format. The CNN consists of 3 layers, with the first 2 layers using filter size of 5×5 and the third layer using filter size of 1×1. The third convolutional (CONV) layer outputs a 2D feature map that goes through a global max pooling stage, and the scalar output from max pooling is compared against a predefined threshold. If the scalar is smaller than the threshold, the current frame will not be transmitted to downstream processing. The TFF network is trained on the assumption that the backend processor executes the Quasi-Dense Tracking (QDTrack) method [7] for autonomous driving. Given a frame drop rate, the TFF network is trained to minimize the ID switch/frame metric [8] that the QDTrack performs on the BDD100K dataset.

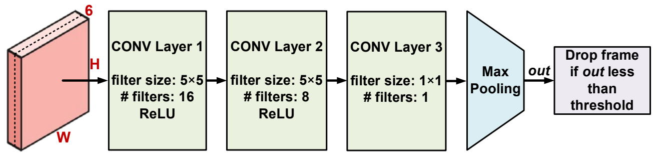


Fig. 2. Architecture of the temporal frame filtering CNN.

B. *Challenges of near-pixel compute for autonomous driving*

Implementing near-pixel compute for autonomous driving poses multiple unique challenges. A typical frame rate for autonomous driving, such as that in the BDD100K, is at 30 frames per second (FPS). Since additional heavier vision tasks remain for the backend processors, low latency for near-pixel compute is necessary. Given the high-resolution images of dimension 1296×720 used in the BDD100K, the TFF network involves 5.23 billion 8×8b multiply-and-accumulate (MAC) operations for the CONV layers. Examining consecutive image frames implies the need to store the frames, but keeping an entire frame for 33.3 ms causes large overheads if traditional storage elements are used. In addition, the TFF and many autonomous vision tasks operate better on colored images, as greyscale images conceal critical information such as traffic light and road warning signs [9]. In order to support color images, Bayer color filters are typically placed on top of the pixel photodiodes, and the accelerator needs to support front-end demosaicing on the raw frames before sending the colored frames to TFF. Our near-pixel accelerator design addresses all of the above challenges, while keeping a low power profile to make the trade-offs of frame filtering more attractive, as well as offering a certain degree of programmability.

III. Proposed Architecture

A. *Pixel array*

Fig. 3 (a) presents the block diagram of the proposed near-pixel compute architecture for TFF network targeting autonomous driving applications. In this design, the pixel array size is set to be 1296×720, matching the video resolution in the BDD100K dataset. We assume that the pixel exposure follows the rolling shutter method, and each pixel column has an analog-to-digital converter (ADC) to quantize the photodiode output.

Since the TFF network requires both the current frame and the difference frame as inputs, the pixel array should be capable of buffering the entire frame to support difference frame

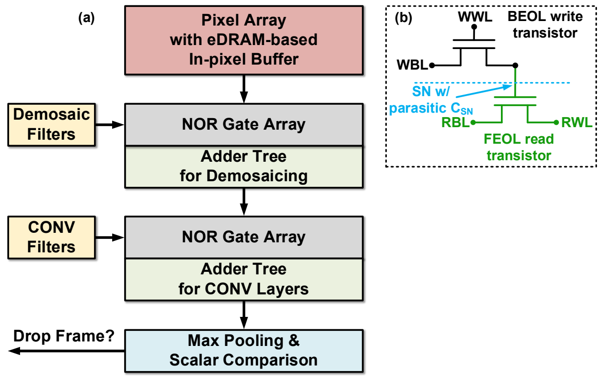


Fig. 3. (a) Schematic of the proposed near-pixel compute architecture; (b) 2T-gain cell based eDRAM with BEOL transistor for the in-pixel buffering.

calculation. Some prior works have demonstrated in-pixel circuit techniques to compute the difference between two consecutive frames using capacitors as storage elements [10] However, these works are demonstrated only in high FPS settings, which helps keep the capacitor sizes manageable. Due to our frame rate requirement of 30 FPS, every frame needs to be retained for 33.3 ms. As shown in Fig. 3 (b), we propose to utilize a 2T gain cell based eDRAM with back-end-of-line (BEOL) compatible oxide transistor [11] to achieve long retention time without need to refresh within 33.3 ms. Each pixel is equipped with an 8-bit eDRAM storage to buffer the converted pixel value for one frame period, and the total eDRAM capacity is 0.93 MB.

To avoid the power-hungry computation for the difference frame, we manipulate the convolution equation to achieve indirect calculation of the difference frame:

$$\text{output} = \text{activations} * \text{weights} \quad (1)$$

$$\text{output} = F[n] * W_{CF} + (F[n] - F[n\text{-}1]) * W_{DF} \quad (2)$$

$$\text{output} = F[n] * (W_{CF} + W_{DF}) + F[n\text{-}1] * (\text{-}W_{DF}) \quad (3)$$

where * is the convolution operator, and F[n] and F[n-1] denote the current frame and previous frame respectively. The network weights for the current frame (CF) and the difference frame (DF) are W_{CF} and W_{DF}. As shown in Eq. (3), the original convolution can be rearranged into a linear combination of the current frame convoluted with modified weights (W_F + W_{FD}) and previous frame convoluted with the negation of its original weights. Thus, the convolution maintains the correct output without involving direct calculation of the difference frame, and this scheme only requires different weights to be loaded for the convolution.

B. *Near-pixel compute architecture*

For each row of pixel exposure, the near-pixel compute units receive data of dimension 1296 columns × 2 channels × 8-bits. The data first enter the demosaicing MAC units that separates each of the 2 mosaiced channels into 3 RGB channels, thereby increasing the input channel depth from 2 to 6. The demosaiced row then enters the MAC units for convolution, which executes the 3-layer CNN layer-by-layer. We set the numbers of both demosaicing and convolution MAC units to be the number of pixel columns (1296), in order to allow parallel processing and eliminate overhead of column sharing. The convolution MAC units support 1×1 and 5×5 filter sizes to accommodate the TFF network, so each pixel column can be broadcast to at most 5 adjacent MAC units for parallelism. The demosaicing units only support 3×3 filter size, so each column is broadcast to 3 demosaicing MAC units. As elaborated in Section IV, the NOR gate array of a MAC unit serves both as a multiplier of activations and weights, and as the intermediate buffer between each layer. The design requires 1.05 MB of NOR gates as buffers to hold all partial sums of the TFF network. The entire weights in the TFF network only occupy 628 Bytes, and are stored on-chip by SRAM. The max pooling layer is executed by parallel comparators on the output of the third CONV layer. Once the entire frame is processed, the maximum is used to compare with a threshold and determine whether this frame is dropped or sent to downstream processor.

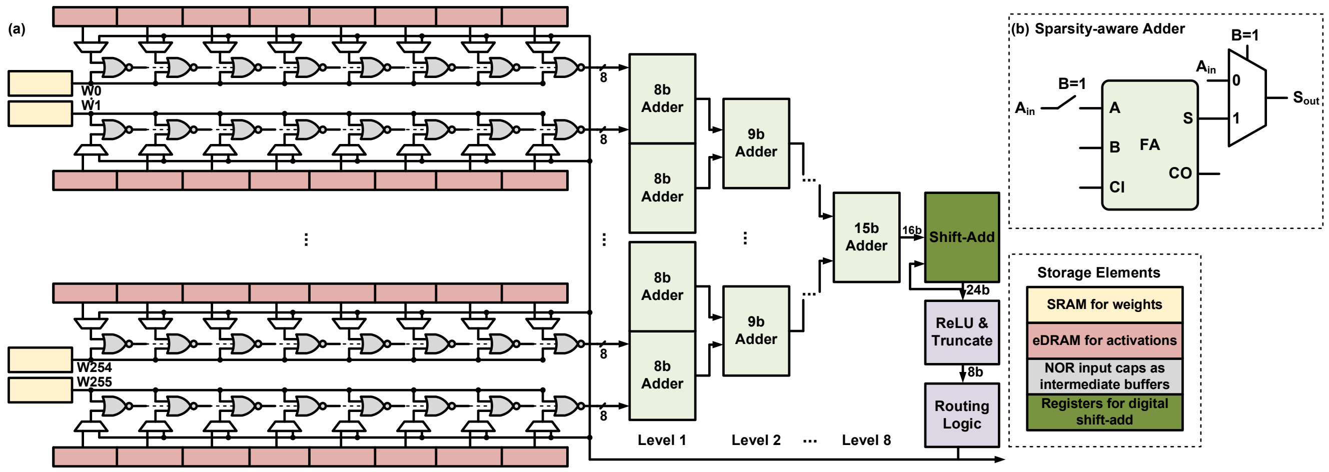


Fig. 4. (a) MAC compute unit with NOR gate array and adder tree; (b) schematic of sparsity-aware full adder.

IV. Circuit Implementations

A. Digital MAC units with NOR gates

Fig. 4 (a) shows the schematic of a digital MAC unit. Each NOR gate in the NOR gate array receives one bit of activation (from eDRAM) and one bit of weight (from SRAM) as inputs. Since NOR only outputs logic 1 when both inputs are logic 0, it can be used as a 1-bit multiplier for two inverted inputs. To accomplish 8×8b multiplication in a MAC operation, all 8 bits of activations are loaded to 8 NOR gates simultaneously, whereas the 8 bits of weights from SRAM are loaded in a bit-serial manner from LSB to MSB. Different weights for each layer are also processed in series. Hence, 9 cycles are required to execute one set of 8×8b MAC operations in this design. For the demosaicing MAC units, the activations come from the 2T gain cell based eDRAM inside the pixel array; and for the convolution MAC units, the activations are routed to the NOR gates from the outputs from the previous layer. We propose that besides as a logic device, a NOR gate can function as a volatile buffer, similar to an 1T1C eDRAM with the NOR input capacitance as storage. This buffering scheme requires that the stored data being accessed sooner than the retention time, which is verified in simulation and discussed in Section V. Buffering using NOR gates offers the advantages of non-destructive reads and absence of memory peripheral circuits.

For the accumulation part of MAC, each convolution MAC unit contains an adder tree with 256 8-bit inputs, followed by a digital shift-add. This allows the adder tree to accumulate 256 8b multiplication results from the NOR gate array in one clock cycle. For the layers with unrolled input feature maps larger than 256, for example TFF's layer 2 (400 inputs), additional cycles are needed to process the layers. For the adder tree, we adopt the hybrid topology as reported in [12], in which standard 28T full-adders (FAs) are interleaved with 14T transmission-gate FAs for improved energy efficiency and area. With 256 inputs, the adder tree contains 8 levels, with 128 FAs on the first level and halved number of FAs for each deeper level. The 16-bit output of the adder tree is shift-added to the previous results for the weight significance, and the final partial sum goes through ReLU and lastly truncated to 8-bit. Routing logic determines whether the 8-bit results are sent to the NOR gate array of next layer as activations or to max pooling for final comparison.

B. Sparsity-aware adder tree

In addition to interleaving 28T and 14T FAs, we introduce another optimization to the adder tree design. As shown in Fig. 4 (b), each FA has one of its inputs gated to become sparsity-aware, which helps leverage the high sparsity level in the adder inputs. For addition between two numbers A and B, it can be observed that the sum S equals A if B = 0. However in this case, the signal A still needs to propagate through the FA circuits to reach the output S, causing transistors to switch and dissipate dynamic power. By gating input A with input B using a switch and adding a 2-to-1 multiplexer to the output S, the FA ensures that if B is 0 then A does not propagate through the circuits, but is instead directly bypassed to the final sum output S_{out}. This sparsity-aware scheme works for both the 28T and 14T adders, and requires 6 additional transistors (2 for input switch and 4 for multiplexer) for each FA. Although the input gating can applied to adders of any levels in the adder tree, the amount of sparsity will likely decrease in deeper levels, and latency will increase due to additional switches in the critical path. Therefore, we only apply input gating to the first level 8b-adders, which account for half of the FAs in the adder tree.

The need to perform demosaicing at the front-end is illustrated in Fig. 5, with the bilinear interpolation as the filter example. The mosaiced input feature map enters 3 smaller adder trees, each of which interpolates one color channel through convolution. This occurs for the current frame and difference frame in parallel. Since the filter weights are stored in SRAM, other filter-based demosaicing methods are supported. The interpolated RGB channels are combined after the adder trees to enter the CONV layer 1 of TFF network.

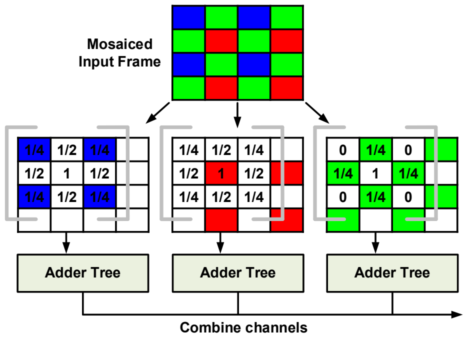


Fig. 5. Front-end demosaicing to interpolate missing color channels.

V. Simulation Results

Fig. 6 compares the ID switch/frame at various drop rates between TFF software baseline and the quantized network whose weights are pruned to 40% sparsity. Lower ID switch/frame is more preferable and indicates that fewer objects are mistakenly swapped during detection and tracking in each frame. We observe that ID switch/frame worsens as the drop rate increases and that the degradation becomes more rapid after 40% drop rate. This means 40% frames will not be transmitted to downstream processors. Performance due to quantization and pruning does not noticeably deviate from the software baseline.

Components of the near-pixel compute units are implemented in TSMC 40 nm LP process. The custom sparsity-aware adder trees and NOR gate arrays are designed and simulated in Cadence Virtuoso, whereas metrics of other digital blocks are obtained through post-synthesis analysis. At a supply voltage of 0.9 V, the design can operate at 100 MHz frequency. Fig. 7 (a) compares the dynamic power between a regular interleaved 8b-adder and a gated interleaved 8b-adder. We observe that with successful gating of input A with input B, FA dynamic power is suppressed by 230×. Fig. 7 (b) shows the simulated power consumption of a sparsity-aware adder tree with various input sparsity levels. At an input sparsity level of 40%, each adder tree consumes 202.9 µW of power.

Each of the demosaicing stage, 1st CONV layer, and 3rd CONV layer can process one row of data in 9 cycles, whereas the 2nd CONV layer requires twice as many cycles due to its larger input feature map. Thus at 100 MHz, one row of pixel can be processed 3 µs after exposure. Transient simulation suggests that the intrinsic input capacitance of a minimum-sized NOR gate at 40 nm node can achieve a retention time of around 10 µs. This does not suffice for our design as one row of pixel data needs to be retained until 5 rows of pixels are processed, which is approximately 15 µs. To mitigate, we add a 0.2 fF capacitor to each NOR gate input to extend the retention time to 20 µs. The area of the near-pixel compute accelerator is 12.5 mm^2. At 100 MHz, the accelerator consumes total power of 303.4 mW.

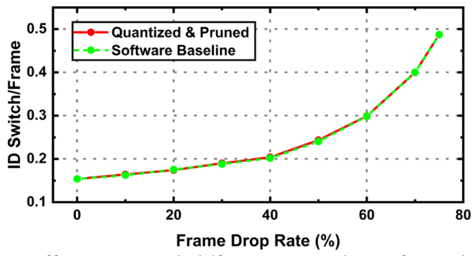
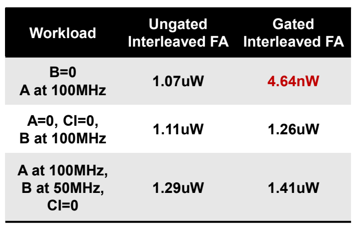
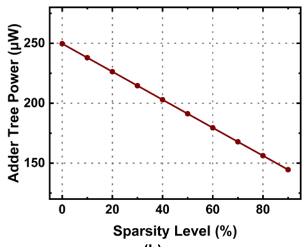


Fig. 6. TFF's effect on ID switch/frame, comparing software baseline and hardware implementation quantized to 8×8b and pruned to 40% sparsity.



Workload	Ungated Interleaved FA	Gated Interleaved FA
B=0 A at 100MHz	1.07uW	4.64nW
A=0, CI=0, B at 100MHz	1.11uW	1.26uW
A at 100MHz, B at 50MHz, CI=0	1.29uW	1.41uW

(a)



(b)

Fig. 7. (a) Comparison of dynamic power between gated and ungated 8b adders; (b) power of sparsity-aware adder tree under various sparsity levels.

Since the pixel exposure time can be hidden by the compute pipeline, the latency to process an entire frame is 2.2 ms. This leaves more than 90% of the frame period for downstream processing. With the near-pixel architecture and proposed techniques in this work, the accelerator design targeting the TFF network achieves an energy efficiency of 15.7 TOPS/W and a compute density of 380.1 GOPS/mm^2 for 8×8b MAC. The design summary is presented in Table I.

TABLE I. Design Summary

Technology	40 nm LP
Activation/weight precisions	8b/8b INT
Area (mm^2)	12.5
Supply voltage (V)	0.9
Operating frequency (MHz)	100
Power consumption (mW)	303.4
Energy efficiency (TOPS/W)	15.7 (8×8b)
Compute density (GOPS/mm^2)	380.1 (8×8b)

VI. Conclusion

In this work, we design and evaluate a near-pixel compute architecture targeting the TFF network that drops redundant image frames. The network helps maintain competitive performance for multi-object detection and tracking when mapped to hardware. The design can process the entire TFF network in 2.2 ms, achieving energy efficiency of 15.7 TOPS/W and compute density of 380.1 GOPS/mm^2 for 8×8b MAC.

Acknowledgment

This work is supported by DARPA IP2 program.

References

[1] S. Grigorescu *et al.*, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics 37(3)*, 2020.

[2] R. Hussain and S. Zeadally, "Autonomous cars: Research results, issues, and future challenges," *IEEE Comm. Survey & Tutorials*, 2018.

[3] K. Matsubara *et al.*, "A 12nm autonomous-driving processor with 60.4TOPS, 13.8TOPS/W CNN executed by task-separated ASIL D control," in *IEEE ISSCC,* 2021.

[4] F. Zhou and Y. Chai, "Near-sensor and in-sensor computing," *Nature Electronics 3 (11)*, 2020.

[5] Z. Liu *et al.*, "NS-CIM: A current-mode computation-in-memory architecture enabling near-sensor processing for intelligent IoT vision nodes," *IEEE TCAS-I*, 2020.

[6] F. Yu *et al.*, "BDD100K: A diverse driving dataset for heterogeneous multitask learning," in *IEEE/CVF CVPR*, 2020.

[7] J. Pang *et al.*, "Quasi-dense similarity learning for multiple object tracking," in *IEEE/CVF CVPR*, 2021.

[8] Y. Li *et al.*, "Learning to associate: HybridBoosted multi-target tracker for crowded scene," in *IEEE/CVF CVPR*, 2009.

[9] F. Secci and A. Ceccarelli, "On failures of RGB cameras and their effects in autonomous driving applications," in *IEEE ISSRE*, 2020.

[10] T.-H. Hsu *et al.*, "A 0.8 V multimode vision sensor for motion and saliency detection with ping-pong PWM pixel," *IEEE JSSC*, 2021.

[11] H. Ye *et al.*, "Double-gate W-doped amorphous indium oxide transistors for monolithic 3D capacitorless gain cell eDRAM," in *IEEE IEDM*, 2020.

[12] Y.-D. Chih *et al.*, "An 89TOPS/W and 16.3TOPS/mm^2 all-digital SRAM-based full-precision compute-in memory macro in 22nm for machine-learning edge applications," in *IEEE ISSCC*, 2021.