# EyeCoD: Eye Tracking System Acceleration via FlatCam-based Algorithm & Accelerator Co-Design

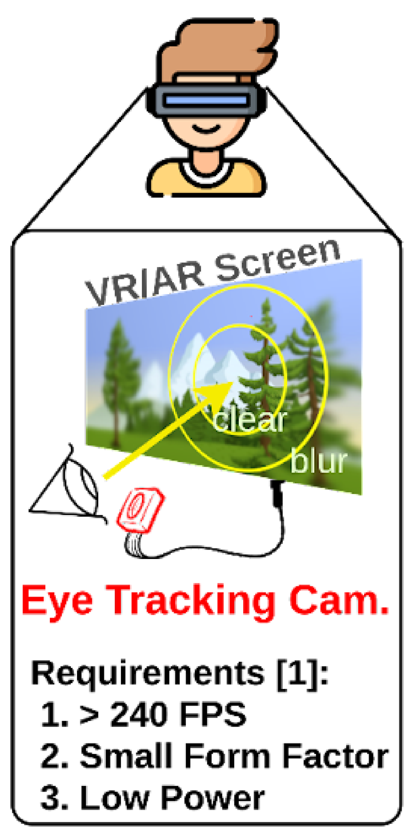
ISCA 2022

H. You[1], C. Wan[1], Y. Zhao[1], Z. Yu[1], Y. Fu[1], J. Yuan[1], S. Wu[1], S. Zhang[1], Y. Zhang[1], C. Li[1], V. Boominathan[1], A. Veeraraghavan[1], Z. Li[2], and Y. Lin[1]

[1] Rice University, Houston, Texas, USA   [2] Meta, Redmond, Washington, USA

## Background & Motivation

- **Eye tracking** is an essential human-machine interface modality in AR/VR

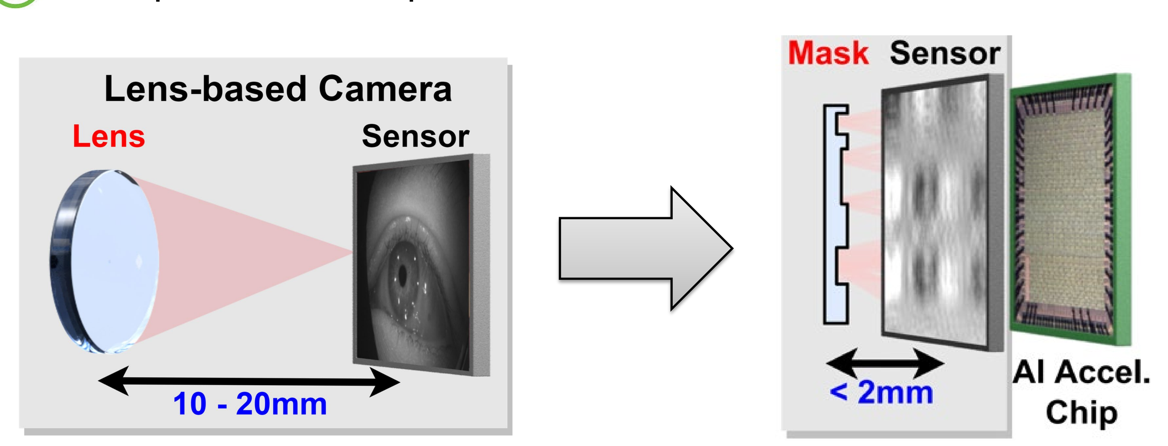



- **Challenges for eye tracking in AR/VR**
  - >240 FPS
  - Small form factor
  - Power consumption in mW
  - Visual privacy
- **Existing works**
  - ☹ An order of magnitude slower (i.e., 30 FPS)
  - ☹ Large form factor and low visual privacy due to the adopted lens-based cameras

→ Fail to meet the requirements

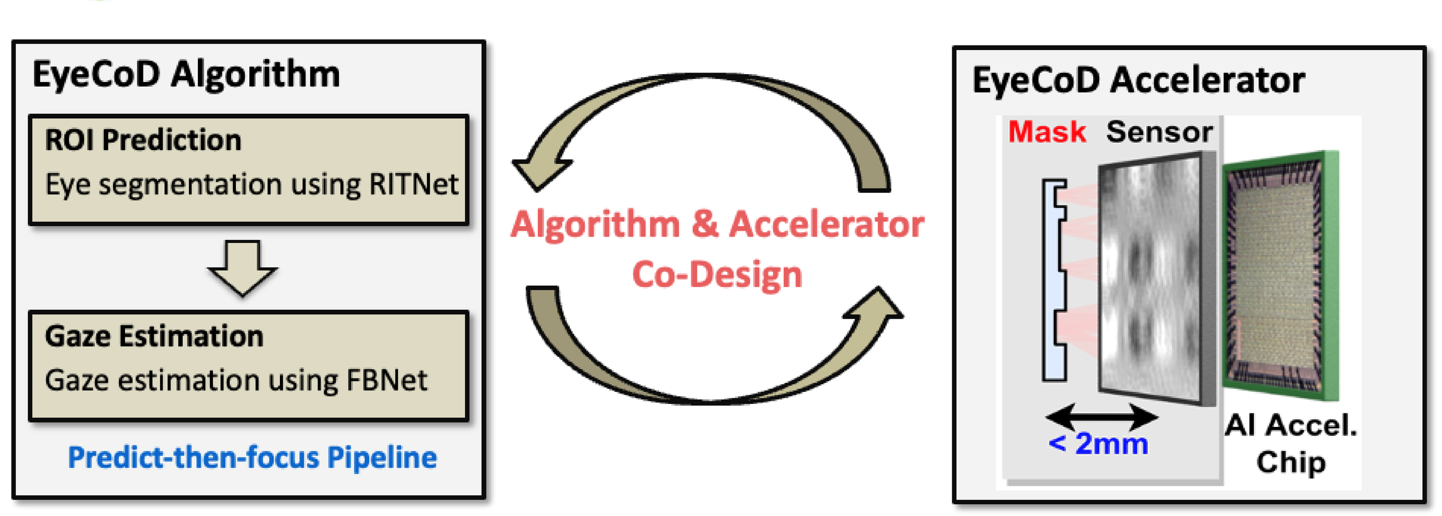
## Unexplored Opportunities for Eye Tracking?

- **Can we build a lensless eye tracking system?**
  - A lensless camera, i.e., FlatCam
    - ☺ Small form factor, i.e., 5-10x thinner
  - An AI acceleration chip featuring algorithm and accelerator co-design
    - ☺ >240 FPS
    - ☺ mW power consumption



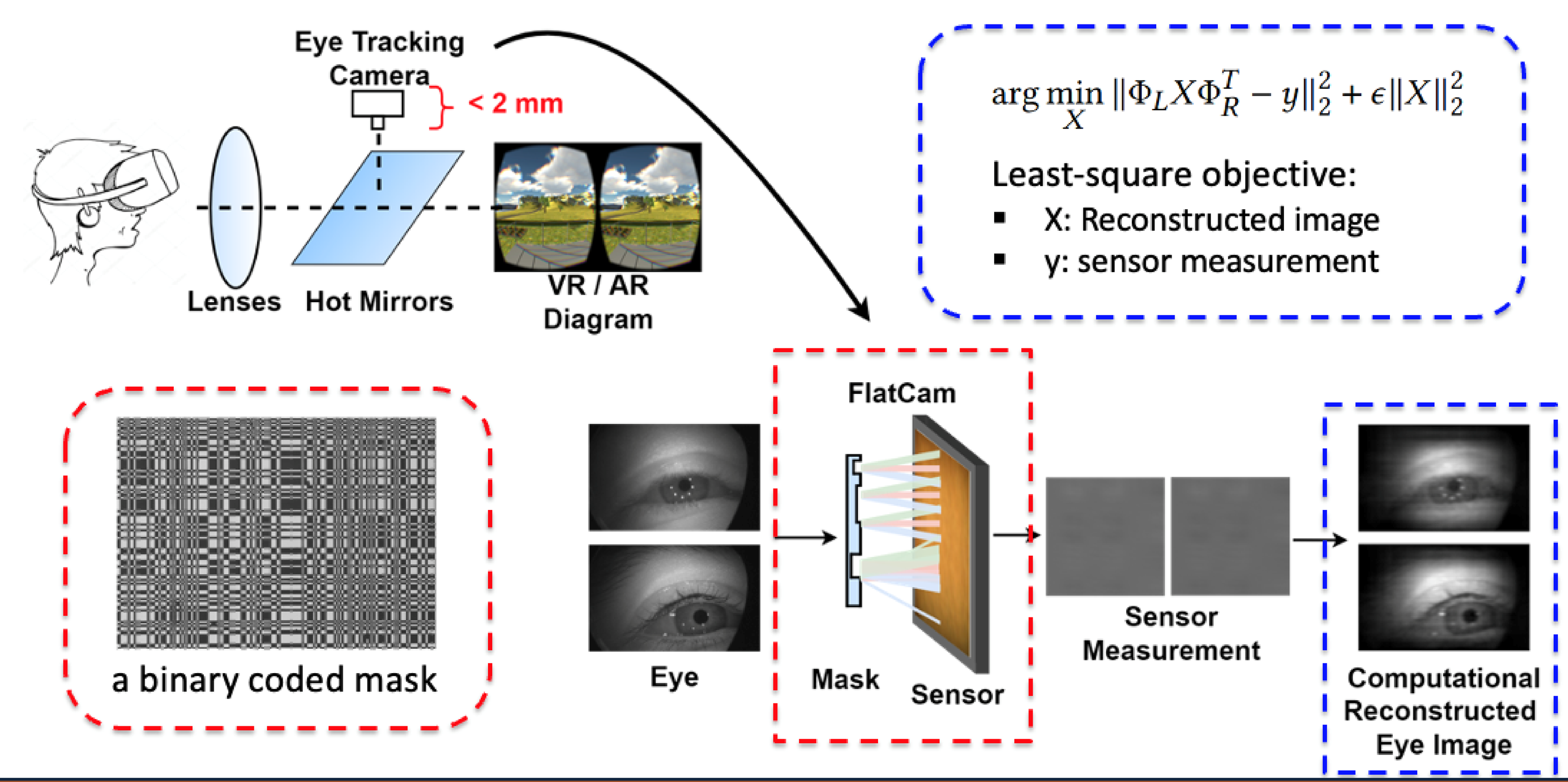

## Proposed EyeCoD

- **FlatCam-based algorithm & accelerator co-design (EyeCoD)**
  - Leverage FlatCam's much reduced form-factor to design a real-time eye tracking system (i.e., > 240 FPS), incorporating
    - ☺ Sensing-processing interface
    - ☺ Predict-then-focus algorithm pipeline
    - ☺ Dedicated accelerator attached to FlatCam



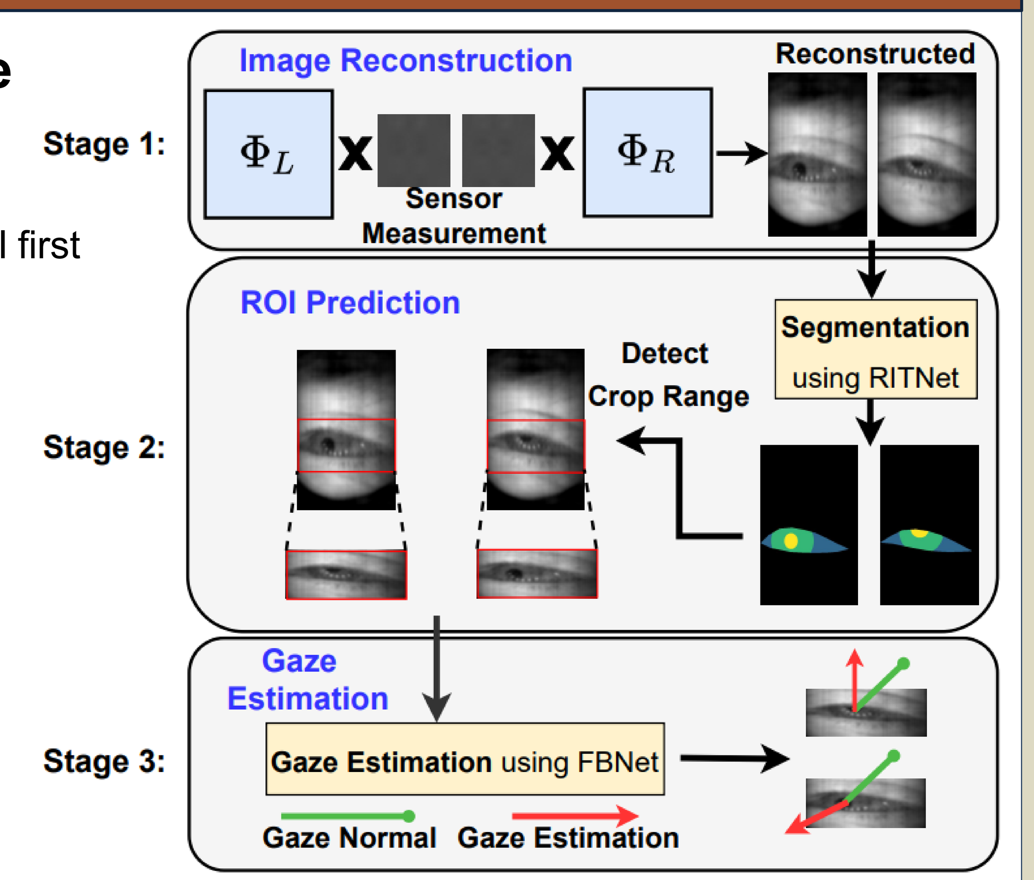

## EyeCoD System Overview

- **EyeCoD system**
  - The core idea in the system side is to replace lens-based cameras to lensless FlatCams → smaller form factor



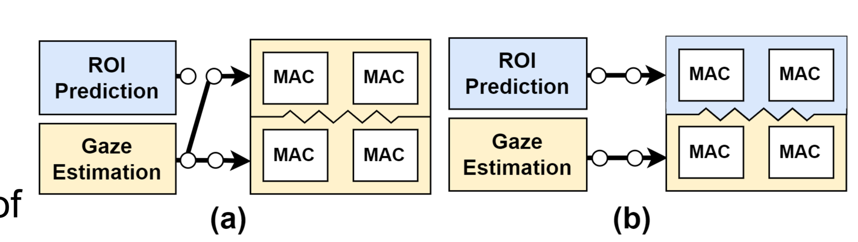

## EyeCoD Algorithm

- **Predict-then-focus pipeline**
  - Stage 1: Image reconstruction
    - Sensing-processing interface replaces FlatCam sensing & model first layer with coded masks
  - Stage 2: ROI prediction
    - Predict and crop the most informative core eye area
    - Once per 50 frames
  - Stage 3: Gaze estimation
    - Estimate the gaze direction based on extracted ROIs
    - Processed for each frame



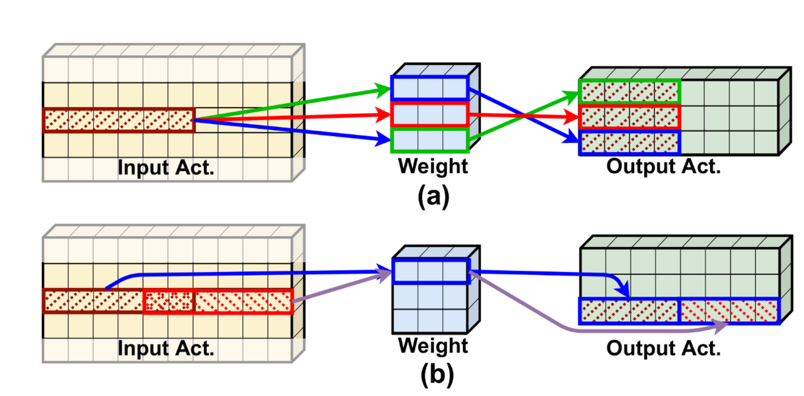

## EyeCoD Accelerator

- **EyeCoD accelerator features**
  - Partial time-multiplexing mode for workload orchestration
    - ☺ Balance the diff. execution frequencies of ROI prediction and gaze estimation
    - ☺ 2.31× speed up over time-multiplexing mode; 1.6× higher energy efficiency over concurrent mode




(a) Gaze estimation only
(b) ROI and gaze estimation when gaze uses < 80% computational resources

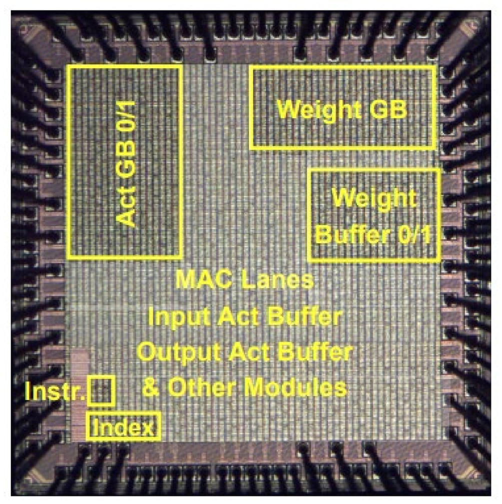
  - Intra-channel reuse for depth-wise convolutional layers (DW)
    - ☺ Reduce 71% of DW's latency
  - Activation partition and memory access parallelism
    - ☺ Save 36% activation memory
    - ☺ Save 50~60% activation BW




(a) Column-wise intra-channel reuse
(b) Deeper row-wise intra-channel reuse

## Evaluation Results

- **Evaluation setups**
  - AI models
    - RITNet for eye segmentation
    - FBNet-C100 for gaze estimation
  - Eye tracking datasets
    - OpenEDS 2019 for eye segmentation
    - OpenEDS 2020 for gaze estimation
  - Metrics
    - Gaze estimation accuracy, Model FLOPs
    - Throughput, Energy efficiency
  - Benchmark baselines
    - EdgeCPU (Raspberry Pi), CPU (AMD EPYC 7742)
    - EdgeGPU (Nvidia Jetson TX2), GPU (Nvidia 2080Ti)
    - Prior eye tracking accelerator, CIS-GEP
  - EyeCoD AI chip
    - Silicon prototype and configurations

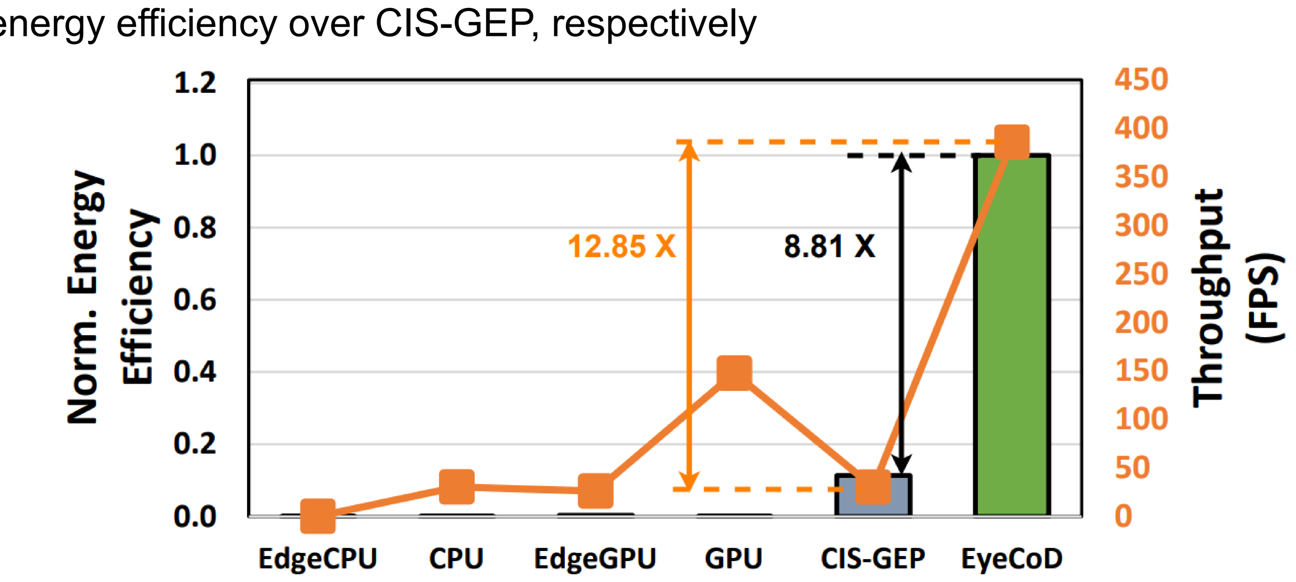



Silicon prototype

| Act GB0/GB1 | Weight Buffer0/1 | Weight GB | Index SRAM | Instr. SRAM |
|---|---|---|---|---|
| 512KB * 2 | 64KB * 2 | 512KB | 20KB | 4KB |

| MAC Lanes | MACs/MAC Lane | Area | Clock frequency | Power |
|---|---|---|---|---|
| 128 | 8 | 8 $mm^2$ | 370MHz | 335mW |

- **EyeCoD over SOTA accelerators**
  - EyeCoD over CPU/GPU platforms
    - EyeCoD achieves up to 2966x, 12.7x, 14.8x, and 2.61x throughput improvements over EdgeGPU, CPU, EdgeGPU, and GPU
  - GCoD over SOTA eye tracking accelerators
    - EyeCoD achieves on average 12.8x throughput improvement and 8.1x higher energy efficiency over CIS-GEP, respectively




  - Breakdown analysis
    - EyeCoD algorithm (P.F.) leads to 1.99x improvements
    - EyeCoD accelerator designs, i.e., Input., Partial., and Depth. further offers 1.22x, 1.28x, and 1.29x improvements, respectively.

| System | Throughput (FPS) | Norm. Energy Eff. |
|---|---|---|
| Lens-based System | 96.34 | 1.00 |
| EyeCoD w/ P.F. | 191.94 | 1.99 |
| EyeCoD w/ P.F. & Input. | 233.64 | 2.43 |
| EyeCoD w/ P.F. & Input. & Partial. | 299.04 | 3.10 |
| EyeCoD w/ P.F. & Input. & Partial. & Depth | 385.66 | 4.00 |

- **P.F.** : Predict-then-focus pipeline
- **Input.** : Sequential-write-parallel-read input activation buffer design
- **Partial.** : Partial time-multiplexing workload orchestration
- **Depth.** : Intra-channel reuse for depth-wise layers

***Acknowledge:***

*NSF RTML & EPCN programs*