Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand, Saugata Ghose, Youngsok Kim,
Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim,
Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, Onur Mutlu.

ASPLOS 2018

Seminar in Computer Architecture
Presented by Xavier Servot
3.11.2022

ETH zürich

SAFARI
SAFARI Research Group

Outline

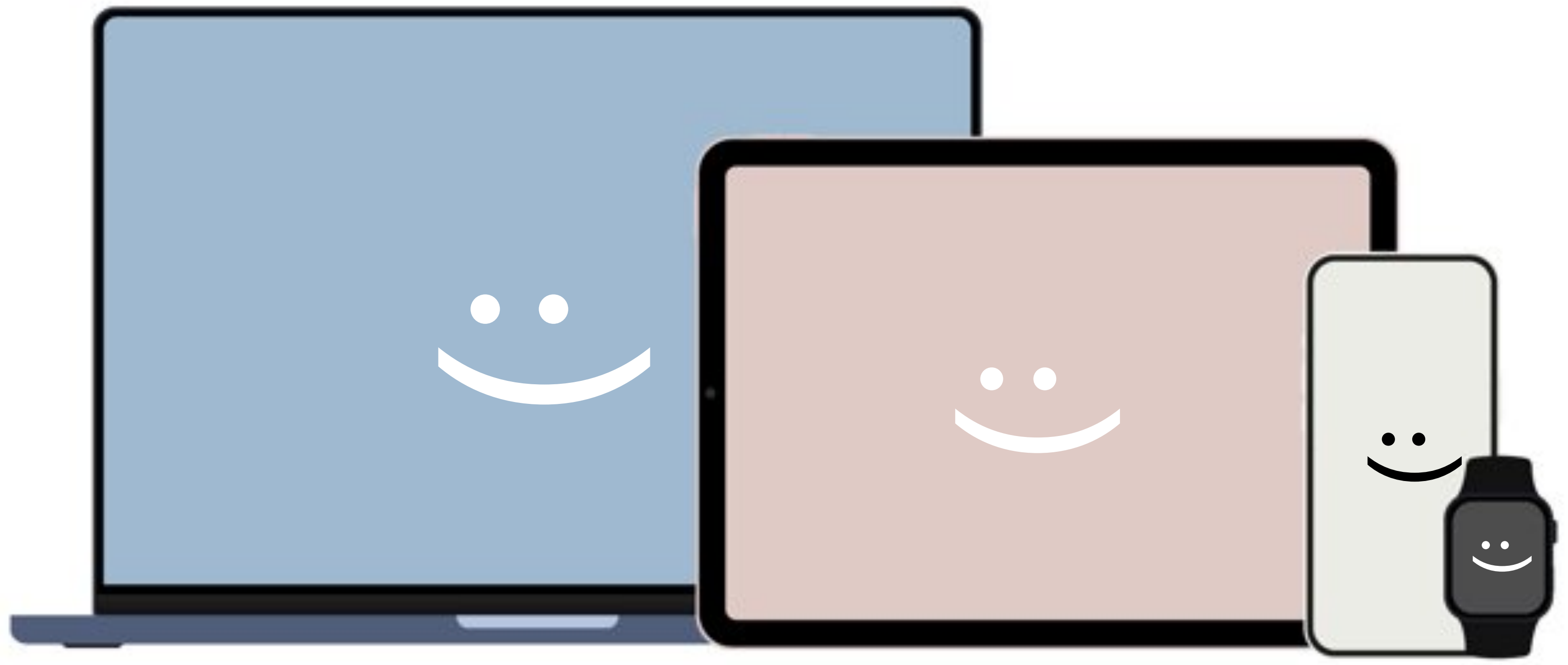
Paper presentation

1. Introduction and Background
2. Methodology
3. Workload Analysis
4. Results

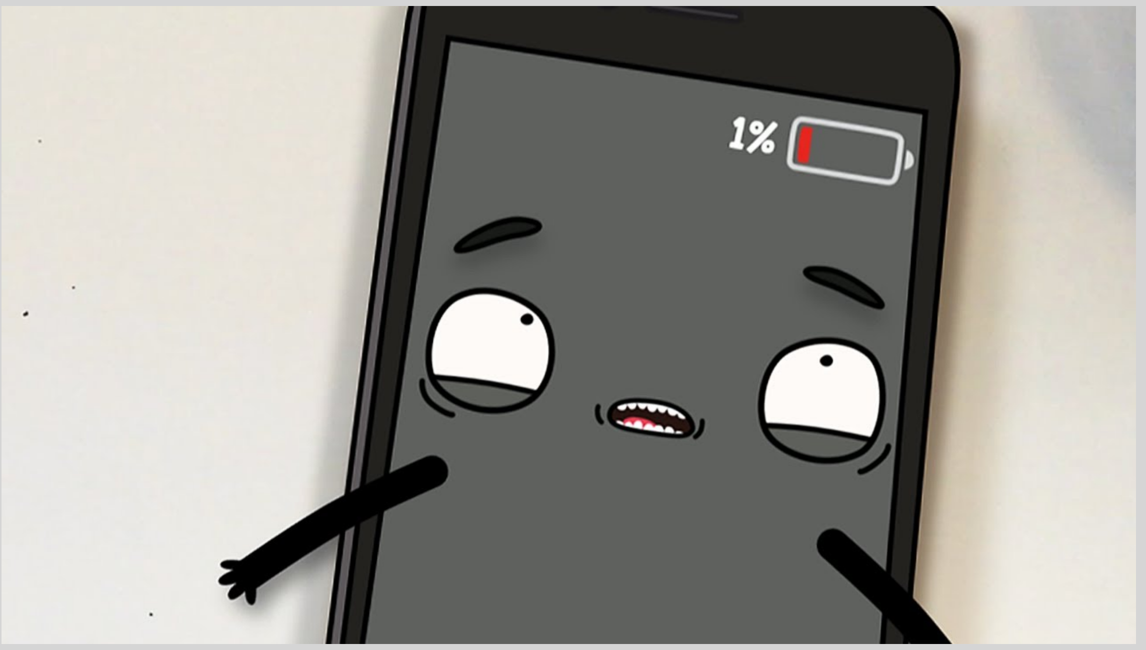
Analysis

5. Strengths
6. Weaknesses
7. Takeaways
8. Discussion

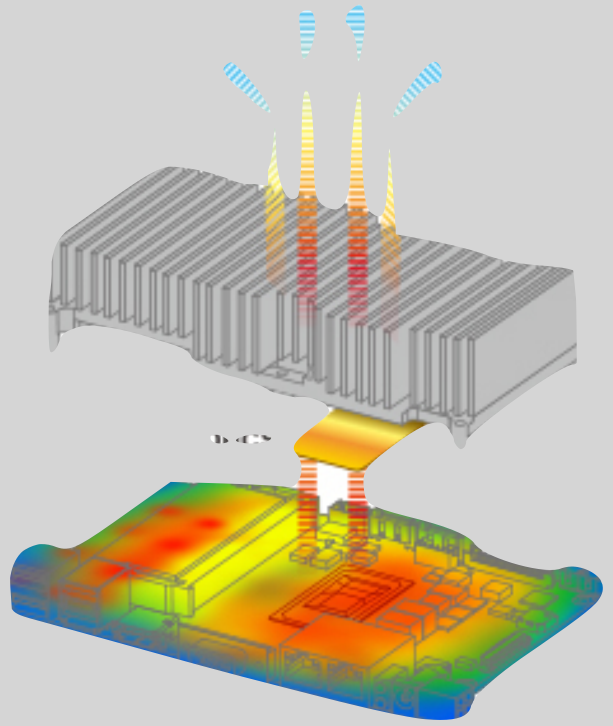
Problems and Motivation



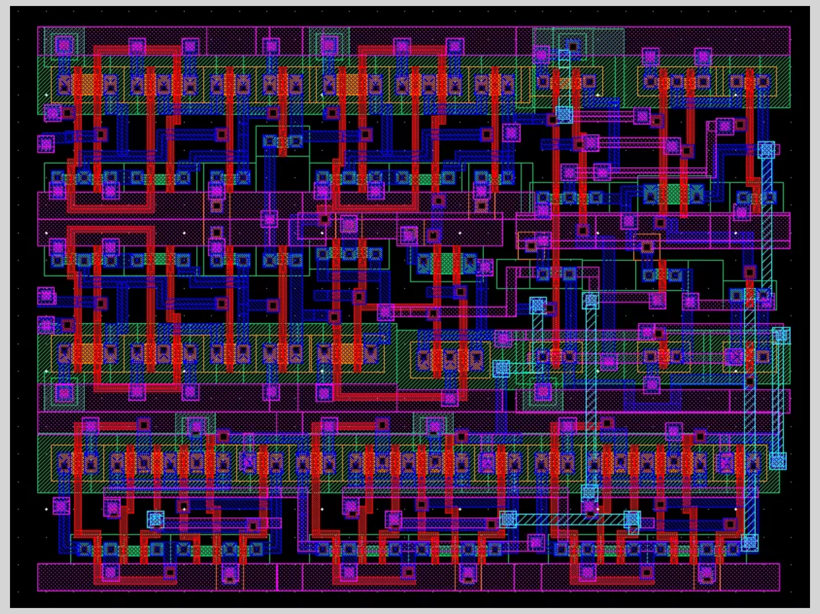
Tight energy budget



Tight thermal heat budget



Tight circuit area budget



➡ How to make Google Consumer Devices more energy-efficient?

Key Idea: Analyze Popular Workloads

Chrome

Google's default web browser

TensorFlow

Google's Deep Learning library

VP9 Decoder

Video Playback

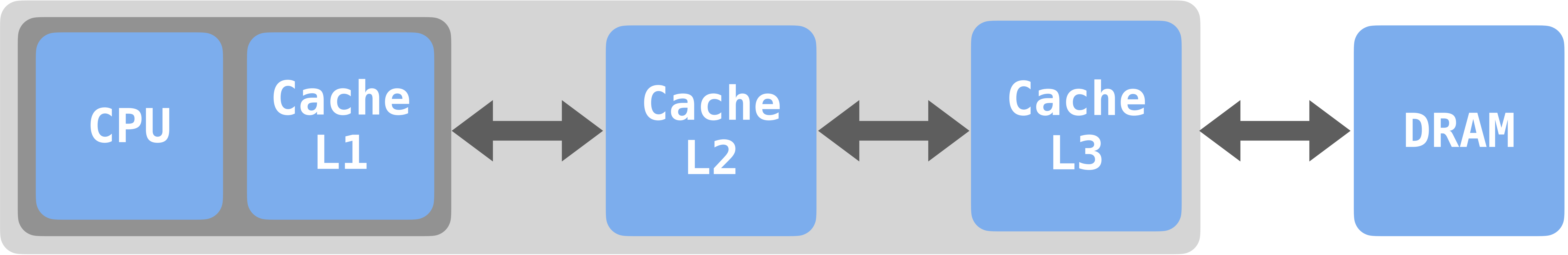
Google's video codec (Used in Youtube)

VP9 Encoder

Video Capture

Google's video codec (Used in Youtube)

Key Observations



On average, 62.7 % of system energy is spent on data movement

`add +` `multiply *` `shift <<` `memcopy`

A few simple primitives are responsible for a large chunk of total energy cost

Key Contributions

❶ **Analyze data movement in these workloads**

PiM Core

PiM Accelerator

❷ **Show opportunities for PiM to alleviate data movement costs**

❸ **Design PiM logic and evaluate efficiency gains**

➡ Reduces energy costs by an average of 55.4%

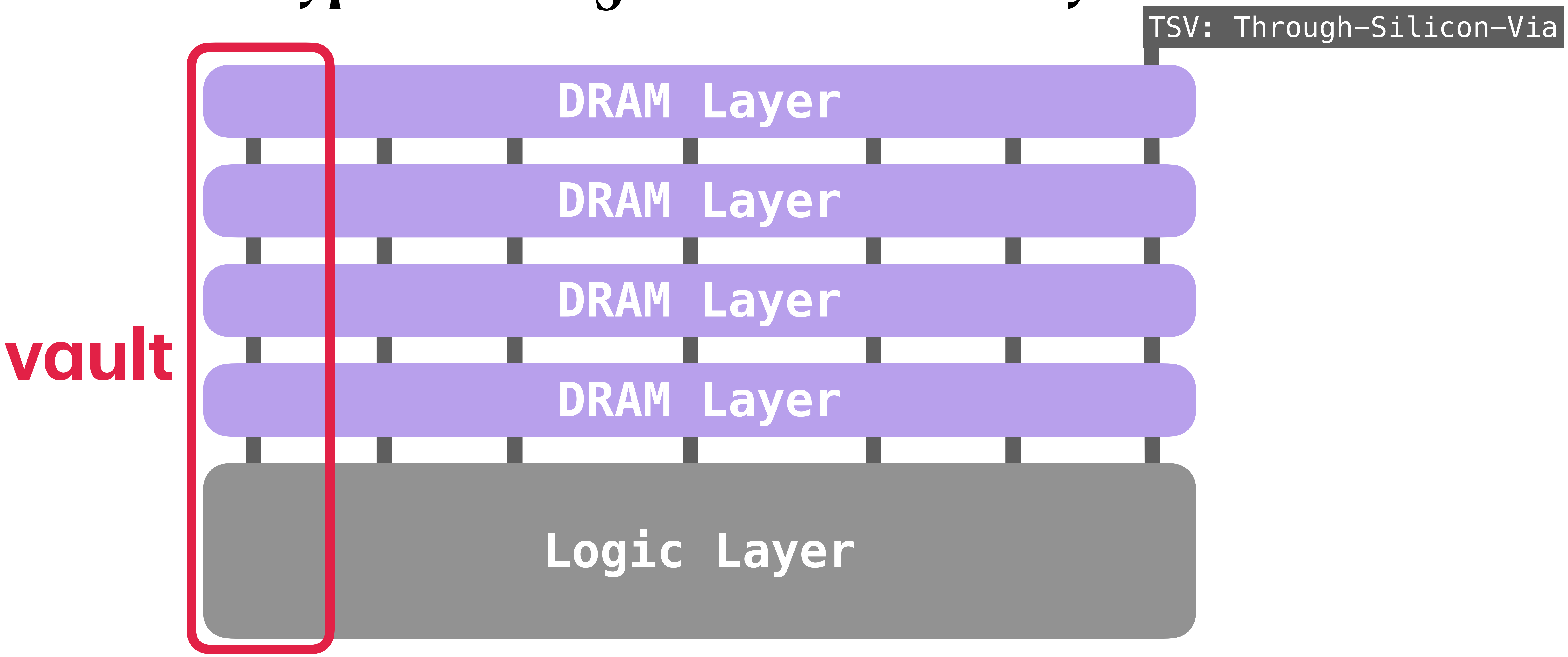
➡ Reduces execution time by an average of 54.2%

Background: Processing-in-Memory (PiM)

PiM: Process data closer to memory

➡ More bandwidth
➡ Lower latency
➡ Higher energy efficiency

Type of PiM: 3d-stacked memory



Outline

Paper presentation

1. Introduction and Background
2. Methodology
3. Workload Analysis
4. Results

Analysis

5. Strengths
6. Weaknesses
7. Takeaways
8. Discussion

Methodology: Workload Analysis

Machine 💻

1. Chromebook with Intel Celeron N3060 dual core SoC
2. 2 GB of DRAM

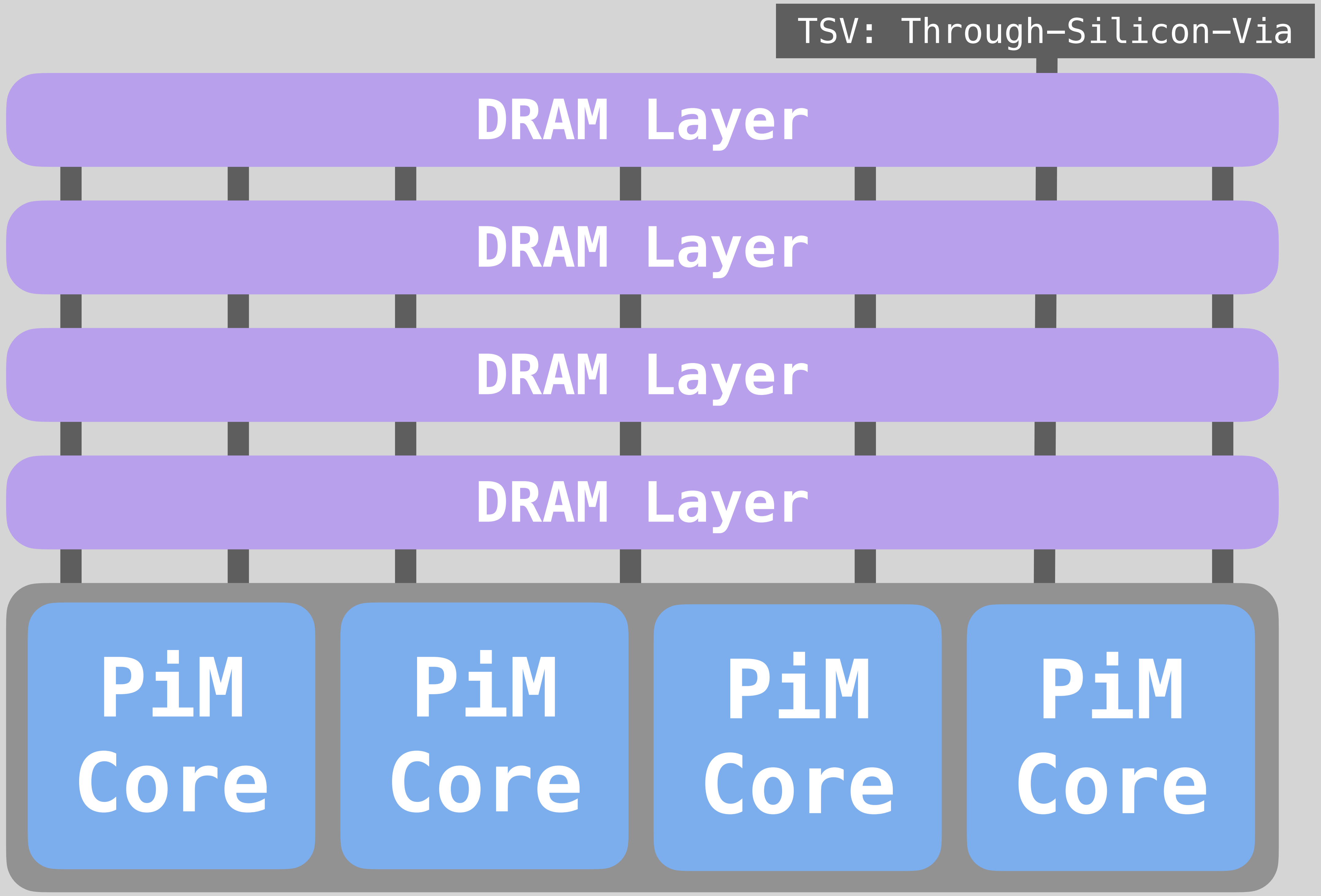
Performance and traffic analysis 🚦

Hardware performance counters on the SoC

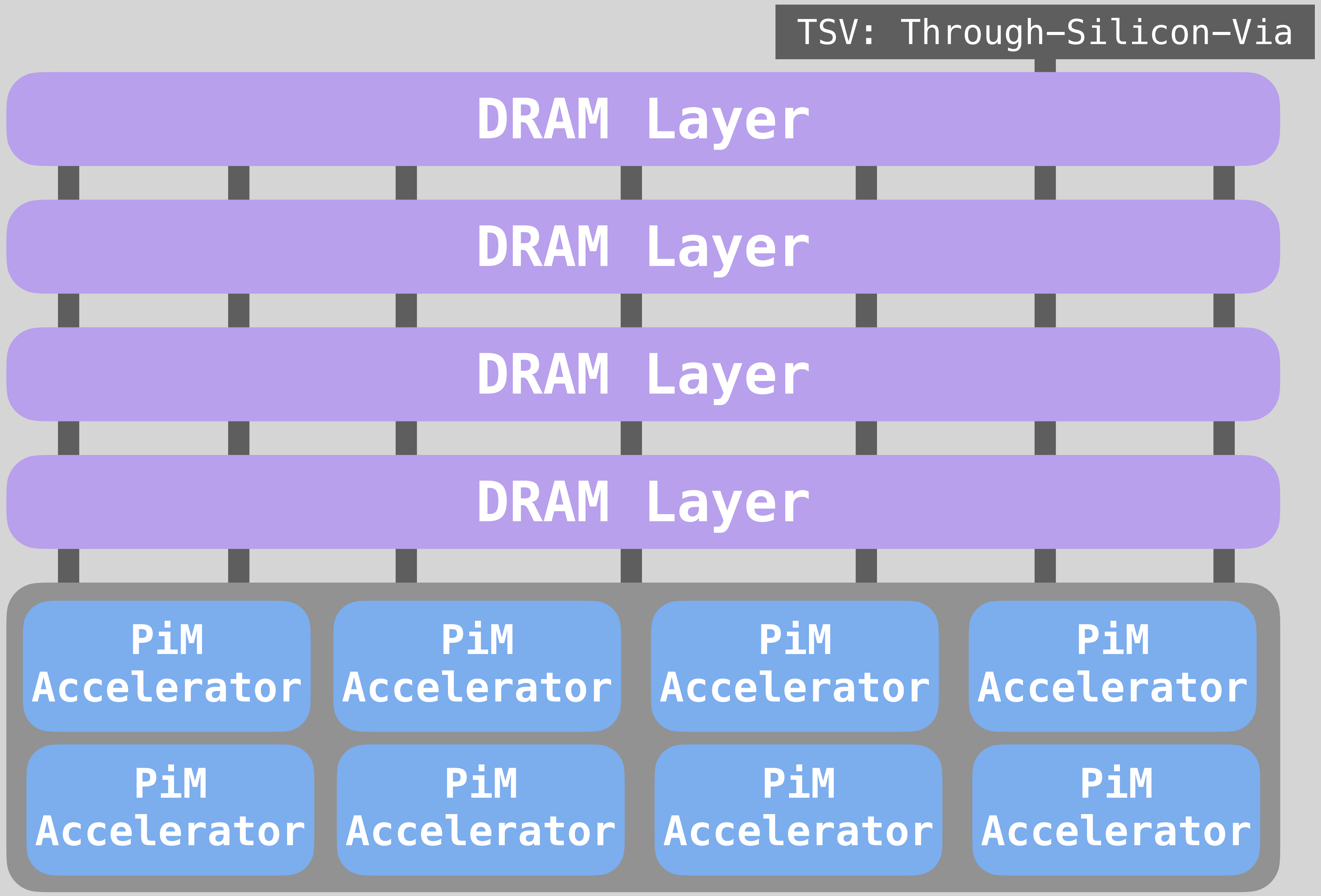
Energy model ⚡

CPU Energy + Cache Energy (L1/2) + DRAM Energy + Off-chip Interconnects Energy

Methodology: PiM Implementation



1. General purpose
2. Low-power: no fancy ILP
3. Data-parallelism ➡ SIMD



1. Custom logic (for each workload)
2. Data-parallelism ➡ Multiple copies

Outline

Paper presentation

1. Introduction and Background
2. Methodology
3. Workload Analysis
4. Results

Analysis

5. Strengths
6. Weaknesses
7. Takeaways
8. Discussion

Chrome

Google's web browser

TensorFlow

Google's Deep Learning library

Video Playback

VP9 Decoder

Google's video codec (Used in Youtube)

Video Capture

VP9 Encoder

Google's video codec (Used in Youtube)

Motivations

Why analyze Google Chrome?

≥ 1 billion monthly active users

What makes Chrome feel fast? (🏎️)

1. Page load time
2. Smooth web page scrolling
3. Quick tab switching

What's next?

1. Take care of scrolling (2) and tab switching (3)
2. Page load time (1) reduces by increasing scrolling and tab switching performance !

Chrome

Page Scrolling

Tab Switching

TensorFlow

Google's Deep Learning library

VP9

Decoder

Video Playback

Google's video codec (Used in Youtube)

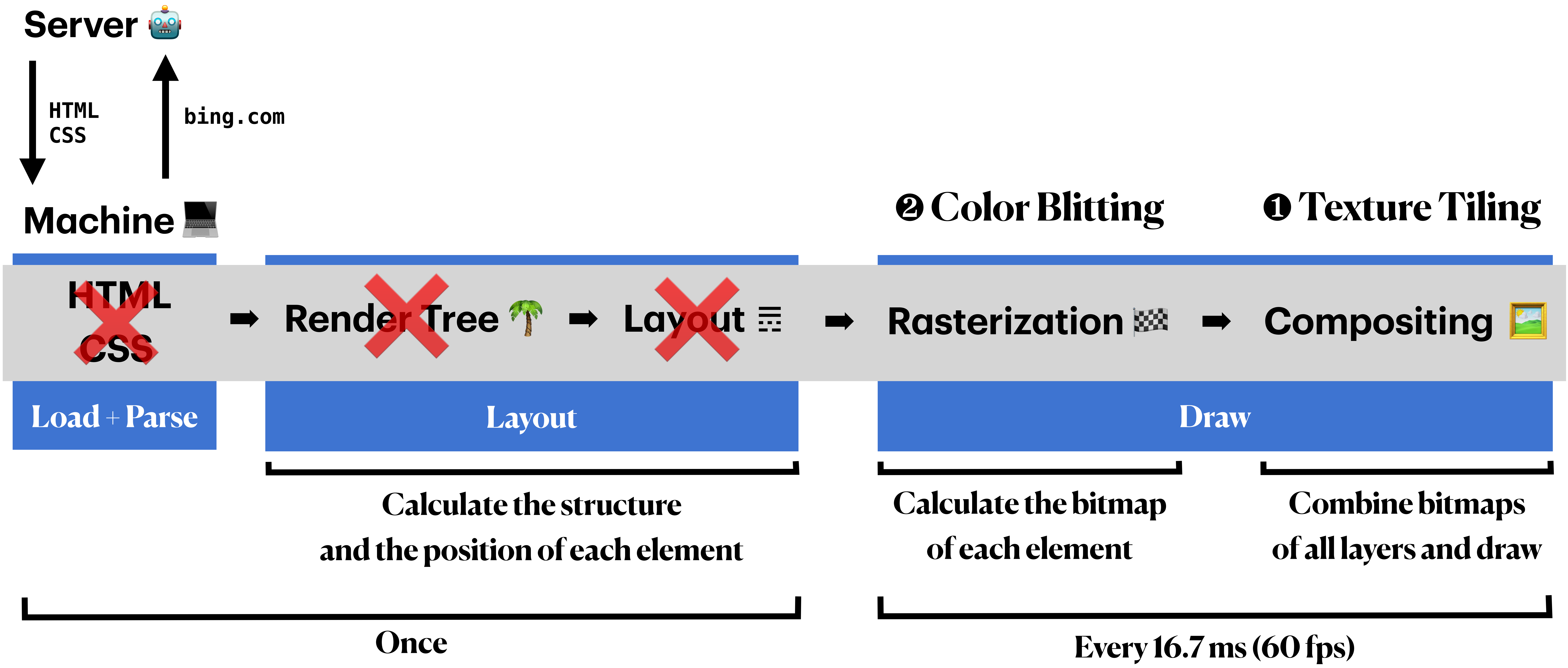
VP9

Encoder

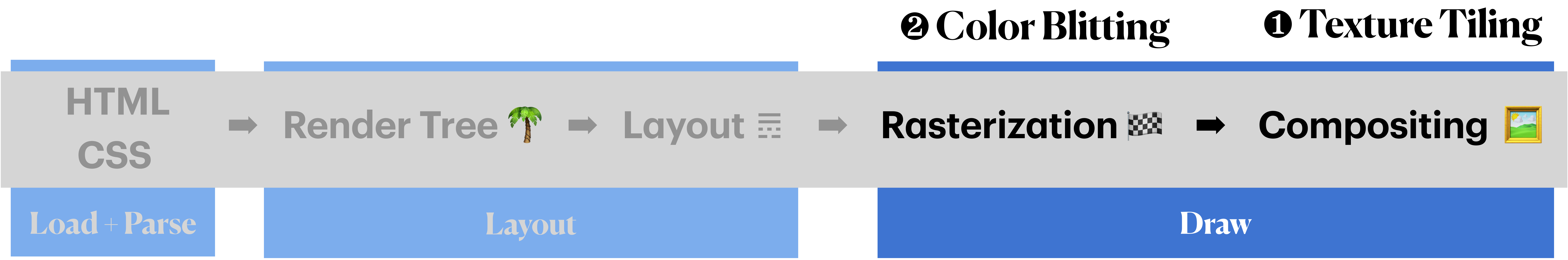
Video Capture

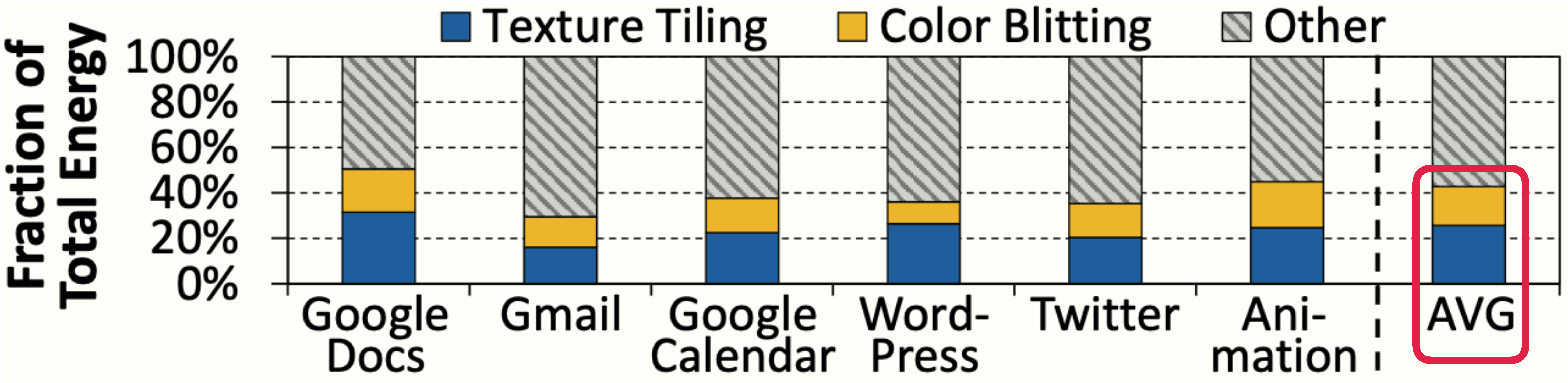
Google's video codec (Used in Youtube)

Page Scrolling



Page Scrolling: Energy Analysis





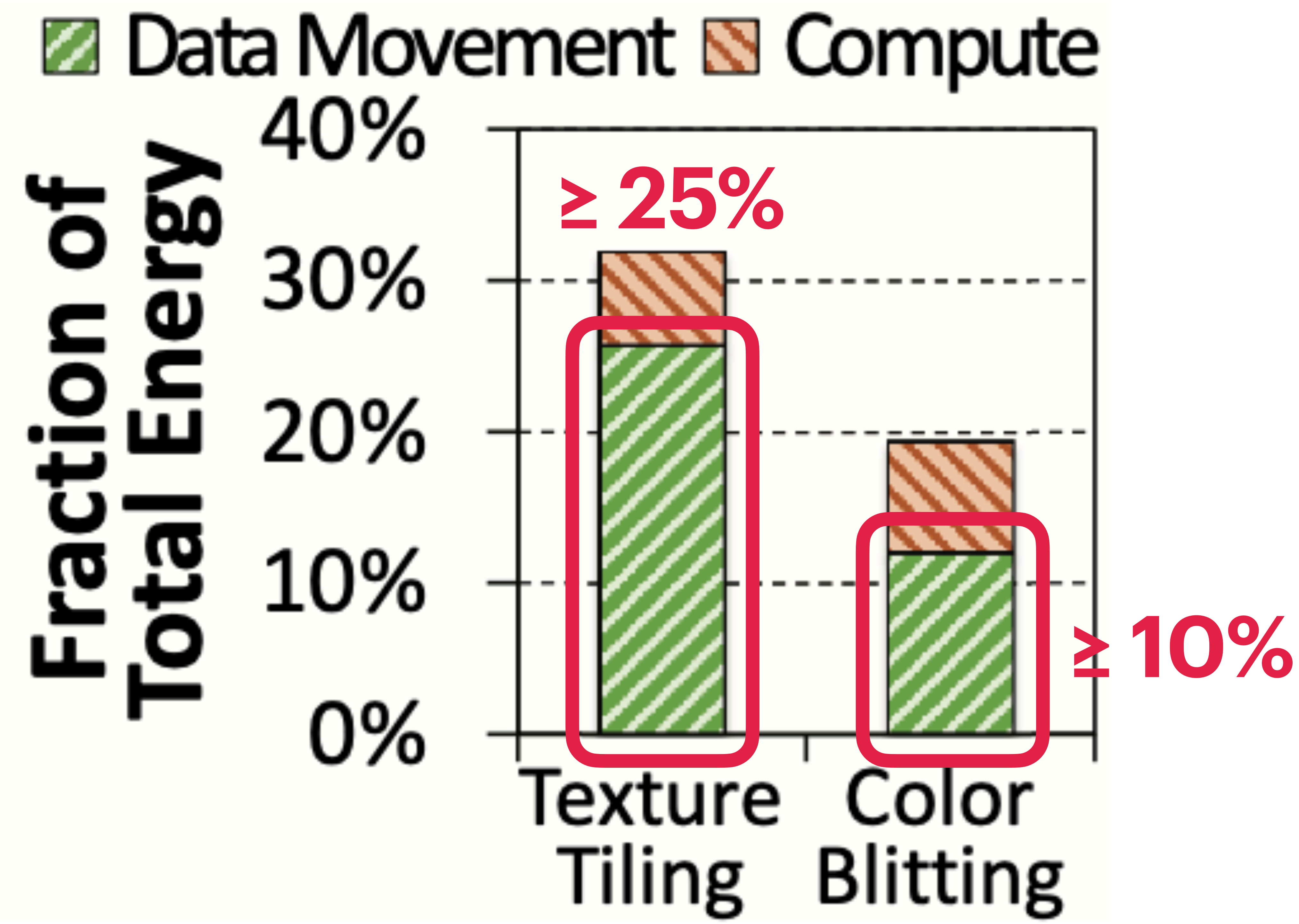
≥ 40%

Page Scrolling: Energy Analysis

❷ Color Blitting ❶ Texture Tiling

HTML CSS ➡ Render Tree 🌴 ➡ Layout ☰ ➡ Rasterization 🏁 ➡ Compositing 🖼️

Load + Parse | Layout | Draw



Recall

Data Movement = 😡

Chrome

Page Scrolling

1. Texture Tiling
2. Color Blitting

Tab Switching

TensorFlow

Google's Deep Learning library

VP9 Decoder

Video Playback

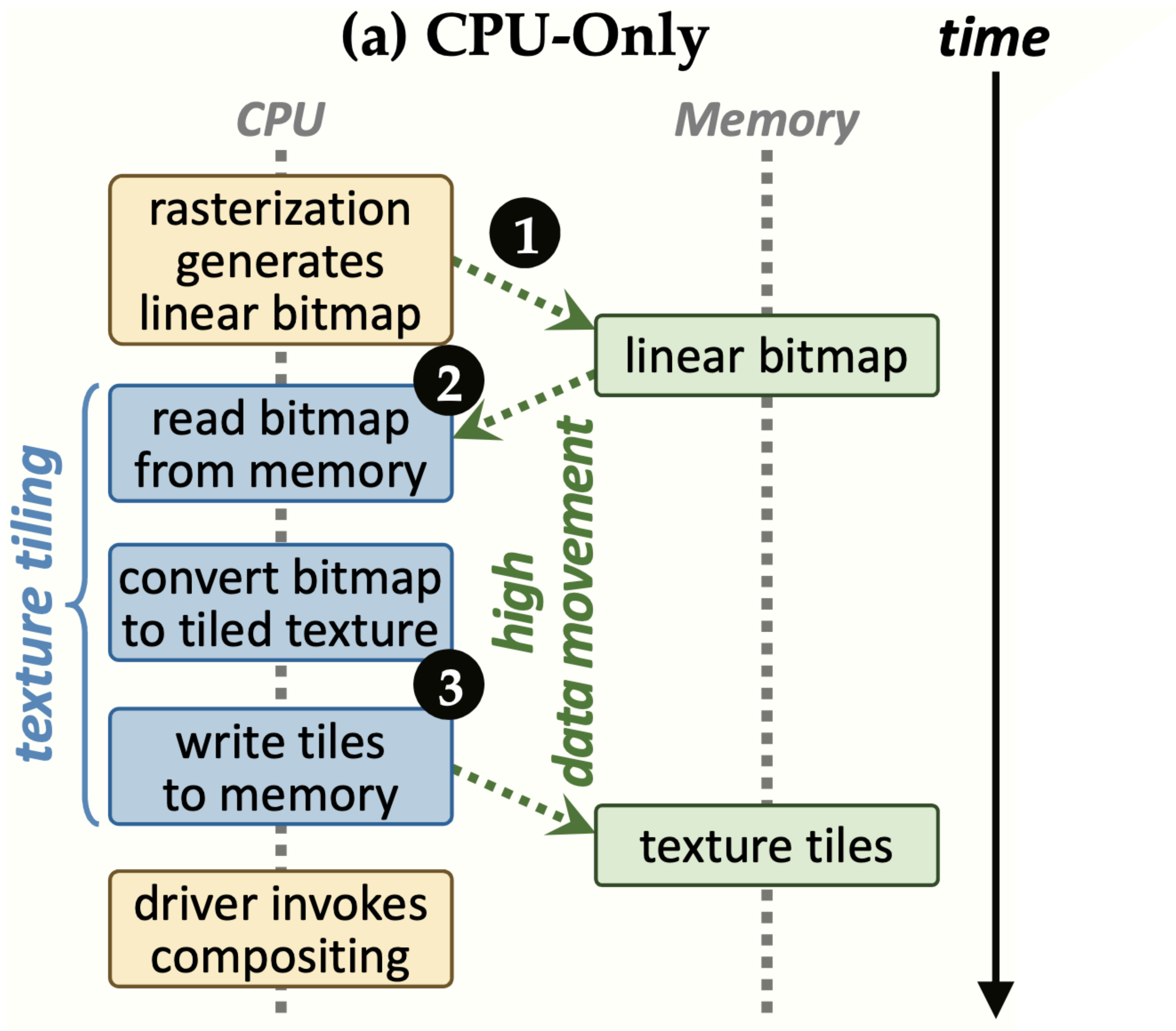
Google's video codec (Used in Youtube)

VP9 Encoder

Video Capture

Google's video codec (Used in Youtube)

PiM Feasibility: Texture Tiling



Is Texture Tiling a good fit for PiM?

1. During texture tiling, **85% of energy consumed by data movements**
2. Poor data locality during texture tiling
3. The rasterized bitmap is big 1024 by 1024 (4 MB)

Is PiM Cost effective?

1. **Simple primitives:** *memcopy*, bitwise logic and addition
2. PiM Accelerator takes 0.25 mm² per vault
3. **7.1% of total per vault area**

Chrome

Page Scrolling

1. Texture Tiling
2. Color Blitting

Tab Switching

TensorFlow

Google's Deep Learning library

VP9 Decoder

Video Playback

Google's video codec (Used in Youtube)

VP9 Encoder

Video Capture

Google's video codec (Used in Youtube)

PiM Feasibility: Color Blitting

Is Color Blitting a good fit for PiM?

1. During Color Blitting, **64% of energy consumed by data movements**
2. Poor data locality due to streaming patterns
3. The rasterized bitmap is big 1024 by 1024 (4 MB)

Is PiM Cost effective?

1. **Simple primitives:**
   memset, add and multiply for alpha-blending, bit shifts
2. PiM Accelerator takes a **small per vault area**

Chrome

Page Scrolling

Tab Switching

TensorFlow

Google's Deep Learning library

VP9 **Decoder**

Video Playback

Google's video codec (Used in Youtube)

VP9 **Encoder**

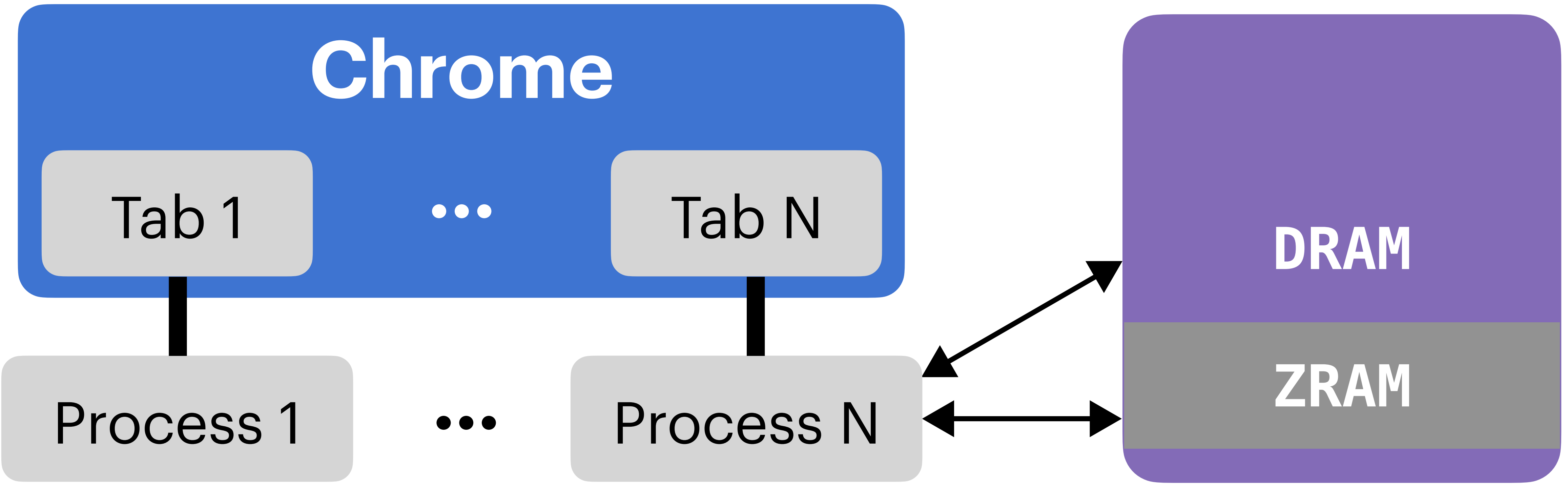
Video Capture

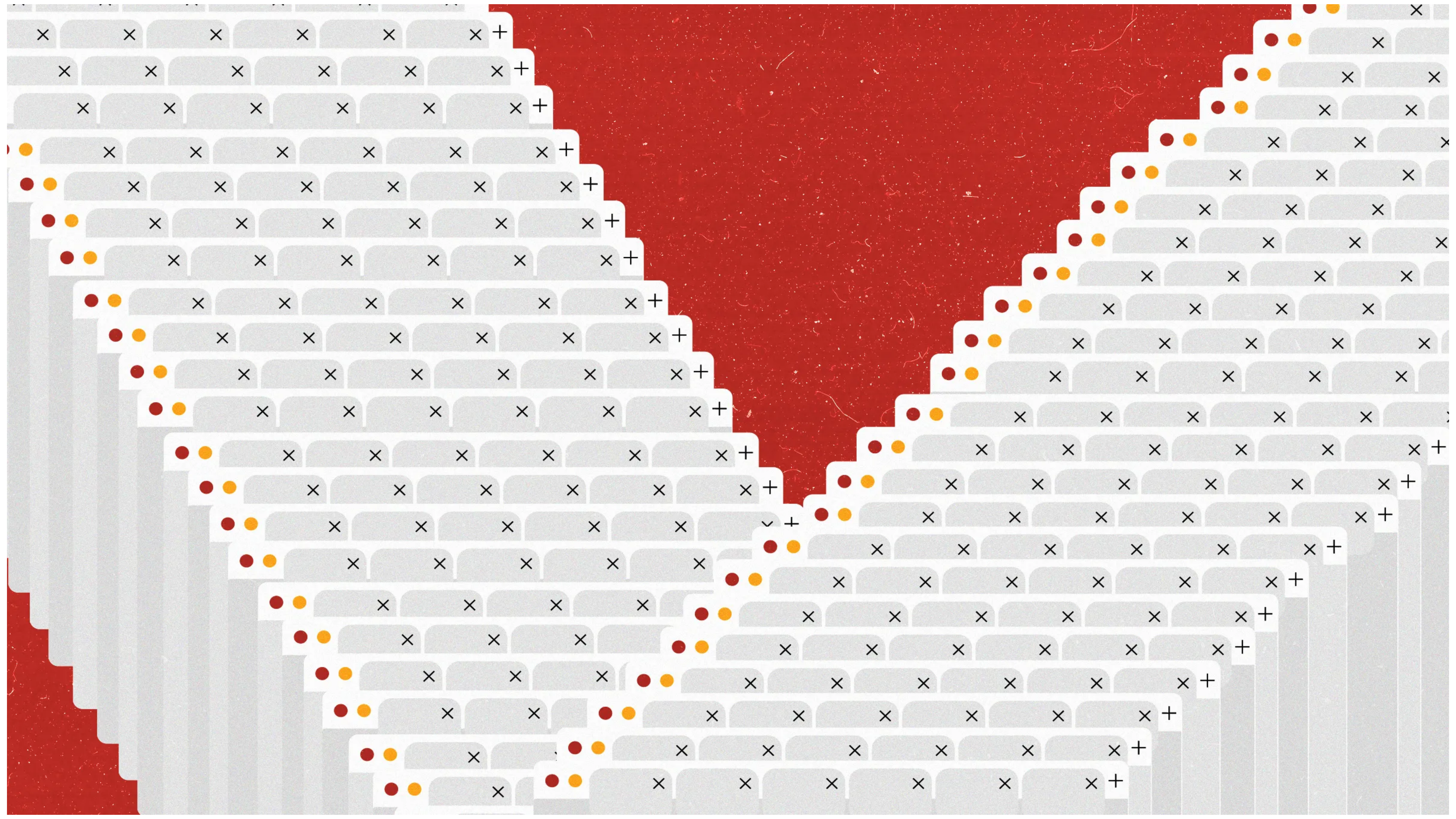
Google's video codec (Used in Youtube)

Google Chrome: Tab Switching

Tab switching: what is?

1. Each tab is its own process
   - ➡ Context-switching
   - ➡ Load page from memory
2. What's the problem?

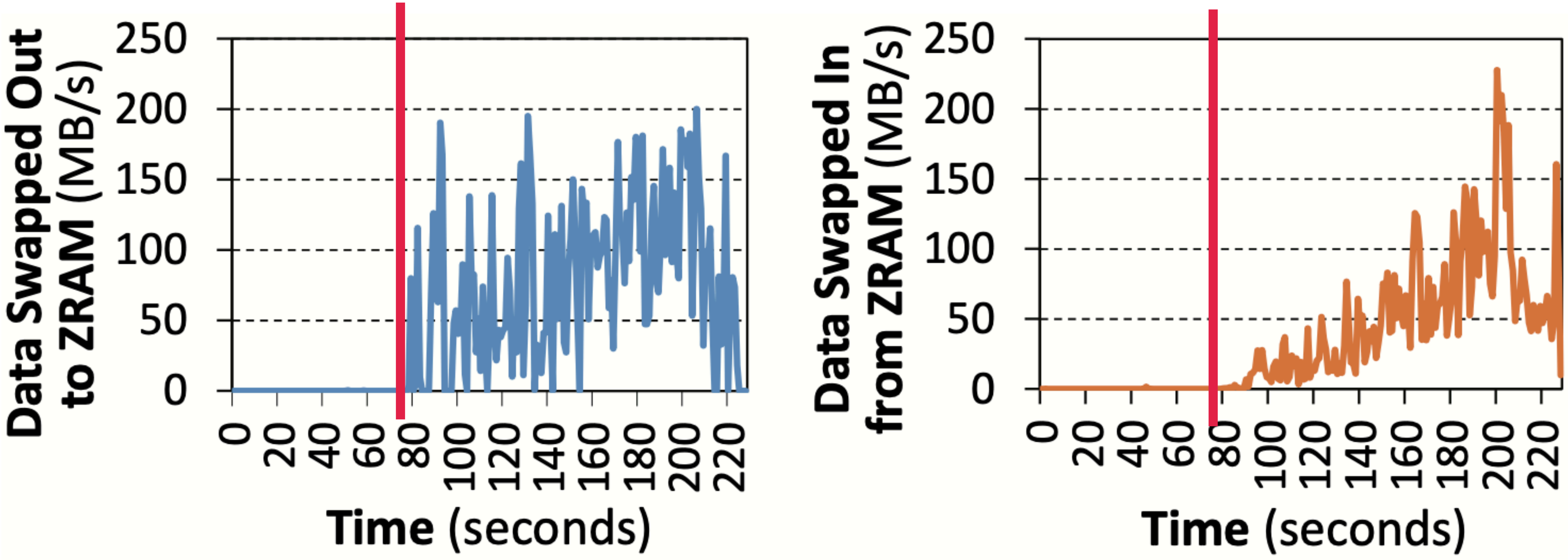




➡ **Memory is a big problem!**

1. Increasingly rich web pages
2. Need responsive tabs ➡ Use DRAM
3. Too many tabs ➡ Compress inactive (ZRAM)
4. Decompress from ZRAM when needed

Tab Switching: Energy Analysis



Methodology

1. Open 50 tabs
2. Scroll for 3s then switch to the next

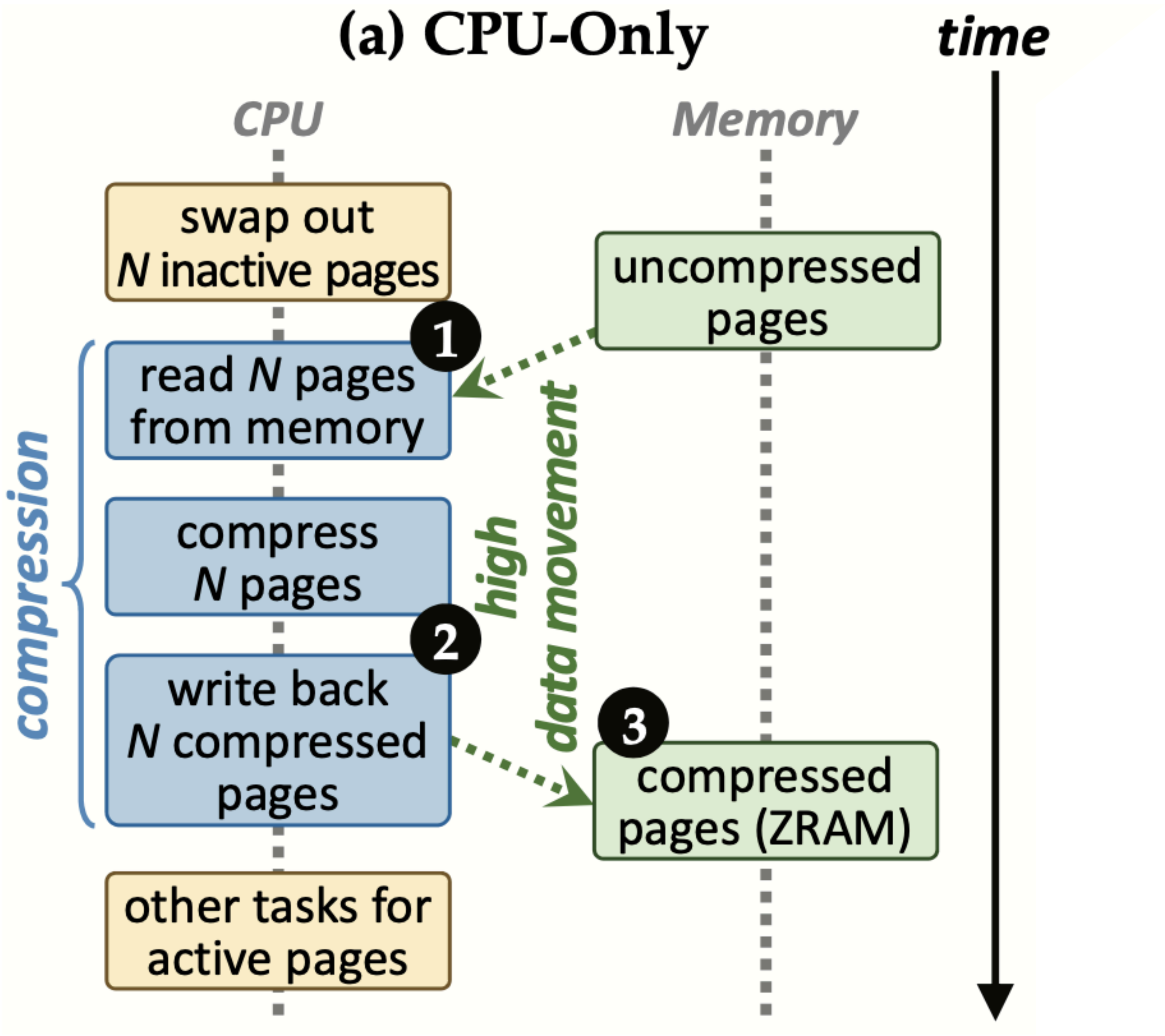
Results

- 11.7 GB of data swapped out to ZRAM
- 7.8 GB of data swapped in from ZRAM

➡ **Total of 19.6 GB of data movement**

➡ 18.1% of system energy spent on compression / decompression

PiM Feasibility: Tab Switching



Is Tab Switching a good fit for PiM?

1. **34.3% of system energy** spent on (de)compression
2. Can be handled in the **background**

Is PiM Cost effective?

1. Simple compression (LZO) has **simple primitives**
2. PiM Accelerator takes 0.25 mm^2 per vault
3. **7.1% of total per vault area**

Chrome

Google's
web browser

TensorFlow

Google's Deep
Learning library

VP9
Decoder

Video Playback

Google's video codec
(Used in Youtube)

VP9
Encoder

Video Capture

Google's video codec
(Used in Youtube)

TensorFlow Mobile

Why analyze TensorFlow Mobile?

1. It's what the cool kids are doing
2. Deep Learning is becoming increasingly used in mobile application (e.g. Google Photos)

What does TensorFlow do?

1. We analyze CNNs: `Conv2D` and `MatMul`
2. Key operations: ❶ **Packing** ❷ **Quantization**

Chrome

Google's
web browser

TensorFlow

Packing

Quantization

VP9
Decoder

Video Playback

Google's video codec
(Used in Youtube)

VP9
Encoder

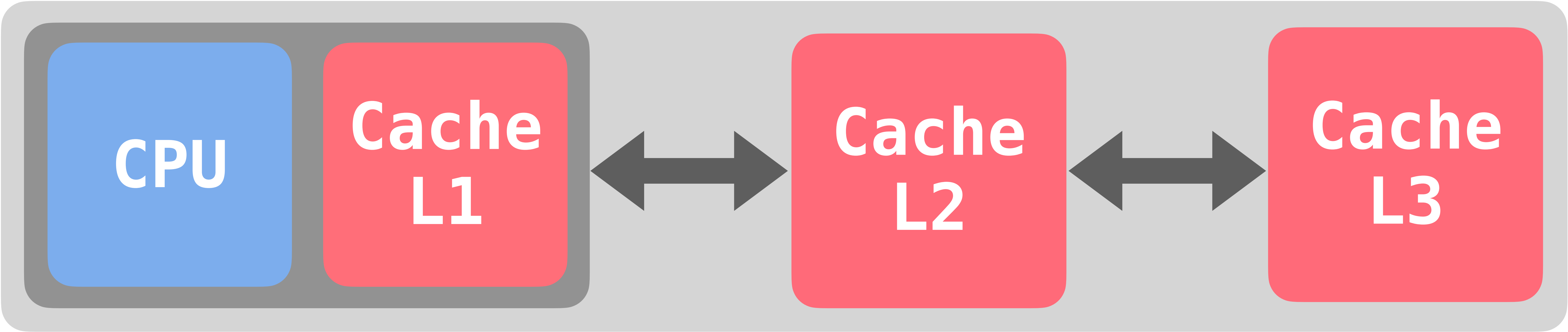
Video Capture

Google's video codec
(Used in Youtube)

Packing

The packing problem

During `MatMul`,
How to load matrix elements into caches to minimize cache miss rate?



Is Packing a good fit for PiM?

1. 33% of total system energy
2. During packing, **82% of energy consumed by data movement**

Is PiM cost effective?

1. **Simple memory reordering**
2. We can reuse the same logic as in texture tiling
   ➡ Cost-effective

Chrome

Google's
web browser

TensorFlow

Packing

Quantization

VP9
Decoder

Video Playback

Google's video codec
(Used in Youtube)

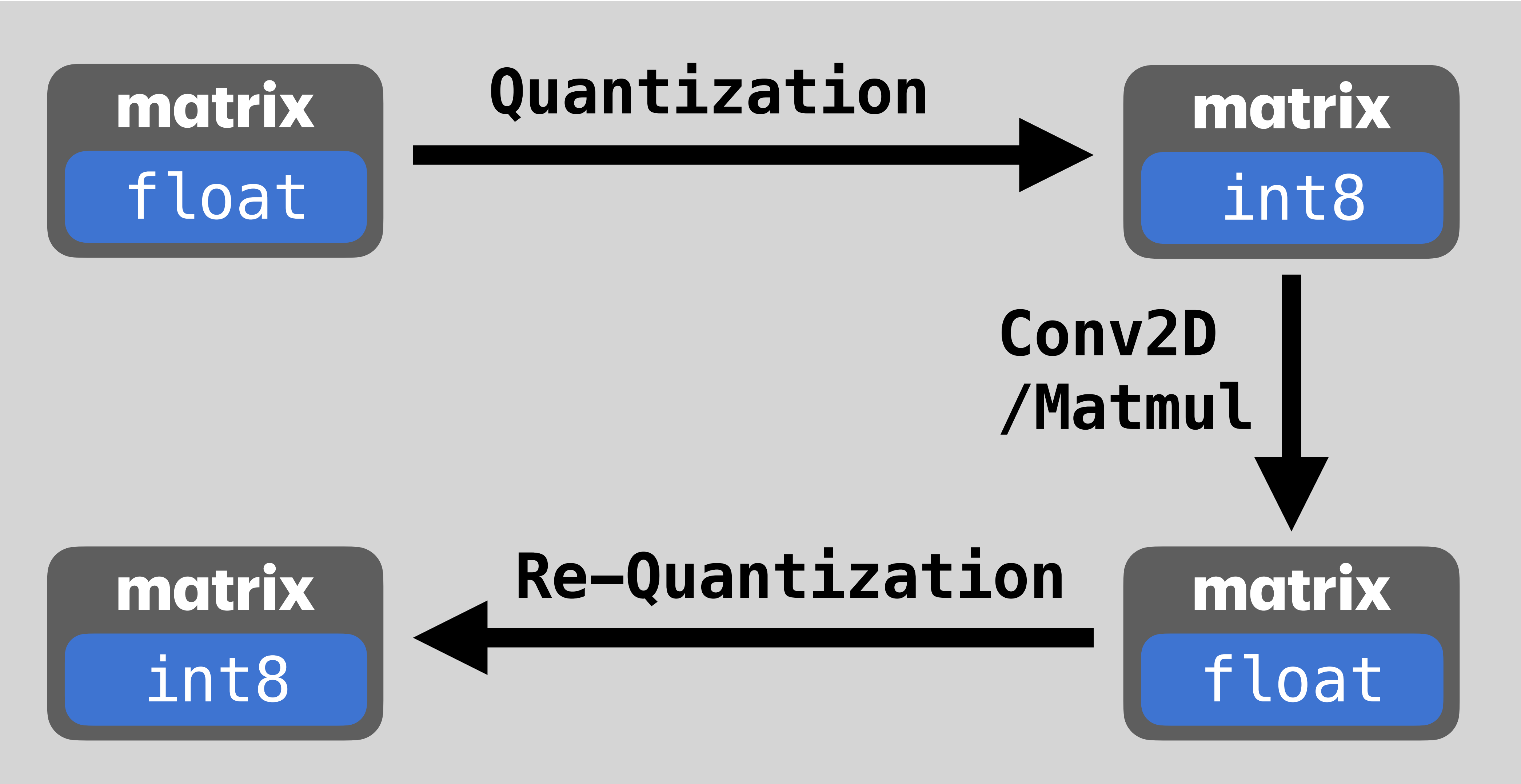
VP9
Encoder

Video Capture

Google's video codec
(Used in Youtube)

Quantization

The quantization process



Is Quantization a good fit for PiM?

1. Up to 16.1% of total system energy
2. During quantization, up to **73% of energy consumed by data movement**

Is PiM cost effective?

1. **Simple primitives:** shift, add, multiply
2. We can reuse the same logic as in texture tiling

➡ **Cost-effective**

Chrome

Google's
web browser

TensorFlow

Google's Deep
Learning library

VP9
Decoder

Video Playback

Google's video
codec
(Used in Youtube)

VP9
Encoder

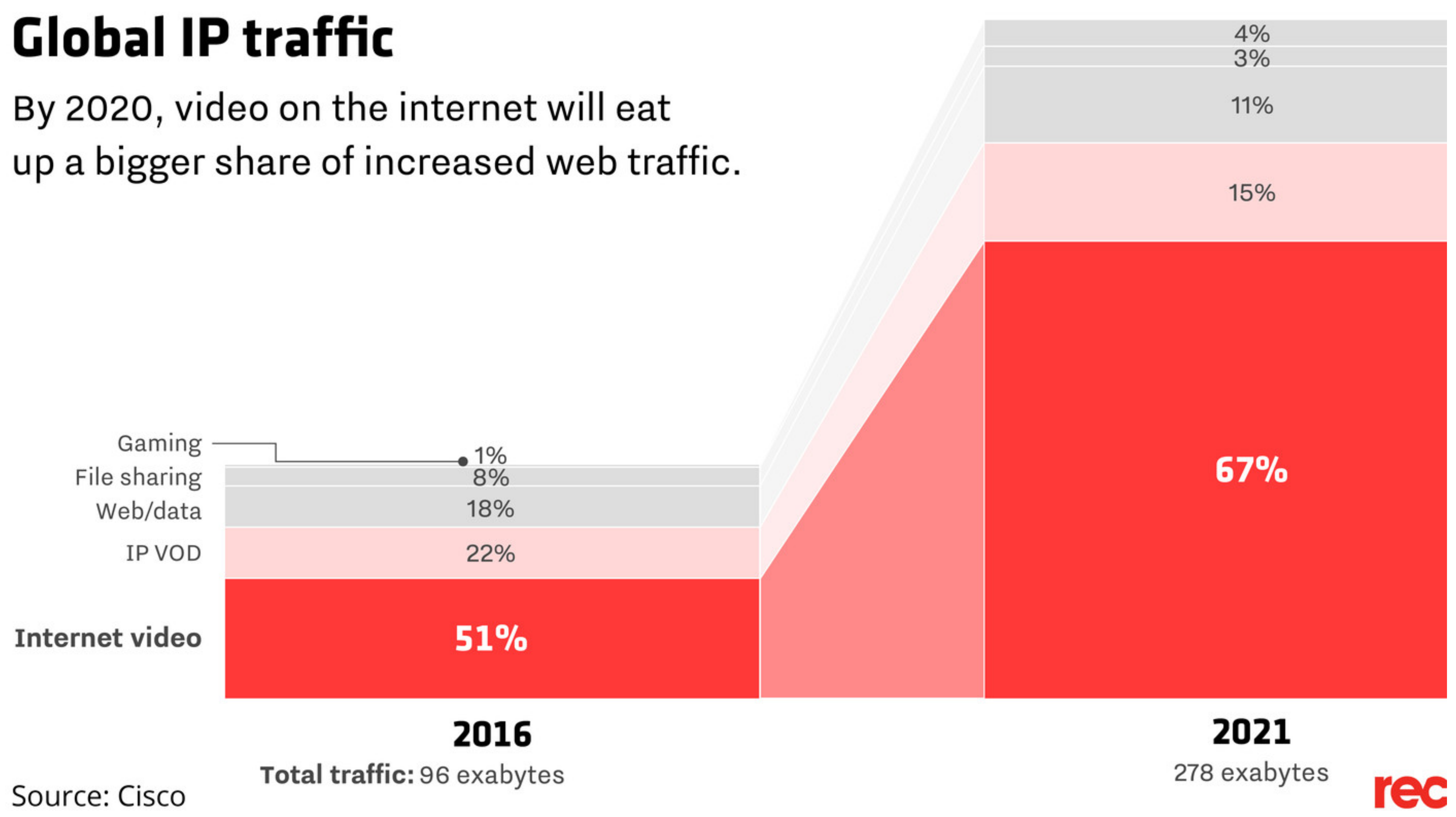
Video Capture

Google's video
codec
(Used in Youtube)

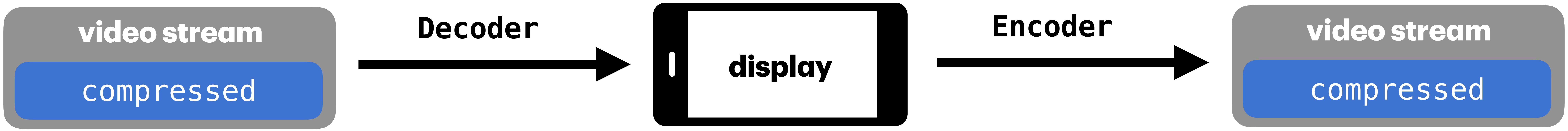
VP9 Motivations

Why analyze video playback and video capture?

1. Youtube, Netflix, Tiktok, Instagram: the videos are not watching themselves!
2. Huge traffic volumes, and set to increase in the future



VP9 Video Playback / Video Capture



Most of the system energy is spent on data movements

➡ Good fit for PiM

The majority of data movement comes from simple primitives

➡ PiM likely feasible

Outline

Paper presentation

1. Introduction and Background
2. Methodology
3. Workload Analysis
4. **Results**

Analysis

5. Strengths
6. Weaknesses
7. Takeaways
8. Discussion

Evaluation Methodology: System Configuration

The gem5 full-system simulator is used with the following system:

SoC

1. 4 OoO cores, 8-wide issue
2. **L1 Cache:** 64 KB **L2 Cache:** 2 MB

PiM Core

1. 1 core per vault, 1-wide issue, 4-wide SIMD
2. **L1 Cache:** 32KB

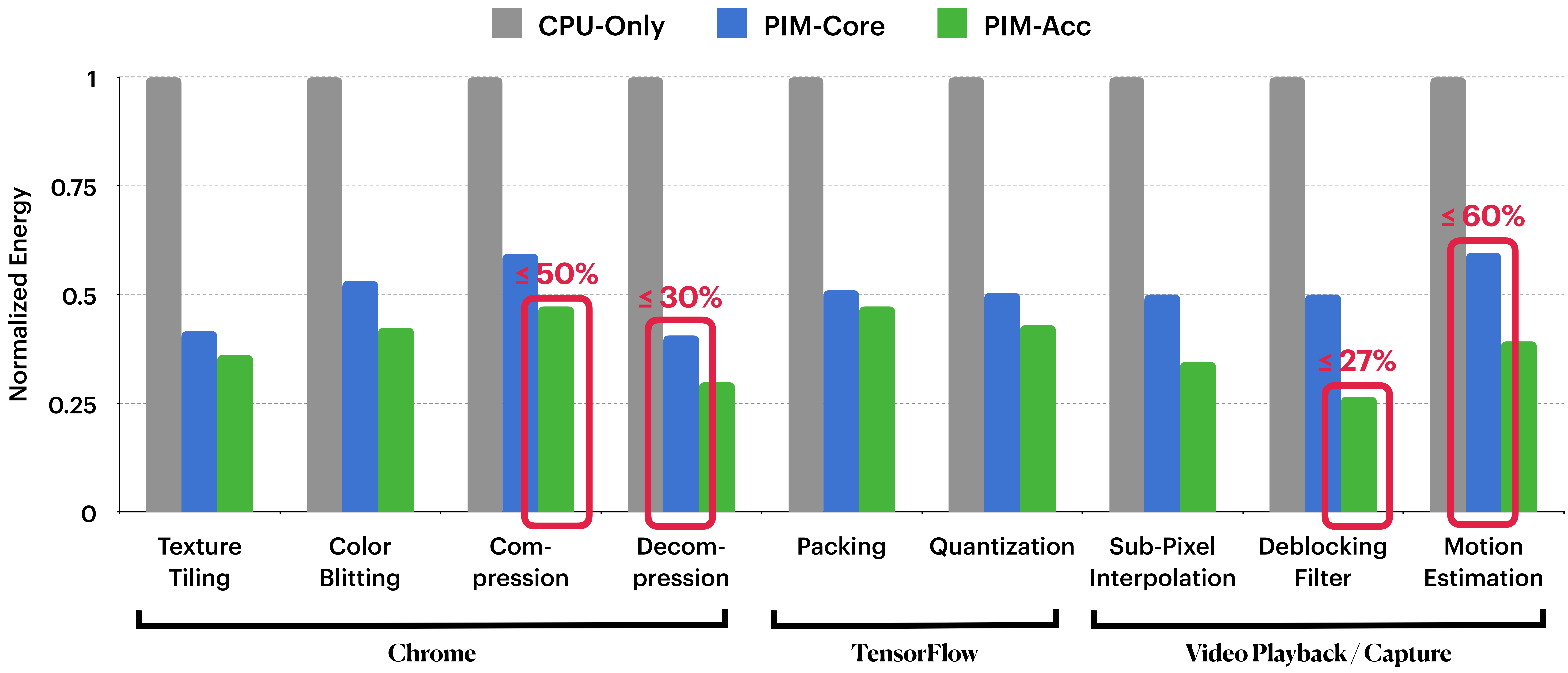
3D-Stacked Memory

1. 2 GB Cube, 16 vaults per cube
2. **Internal Bandwidth:** 256 GB/s **Interface Channel Bandwidth:** 32 GB/s

Baseline Memory

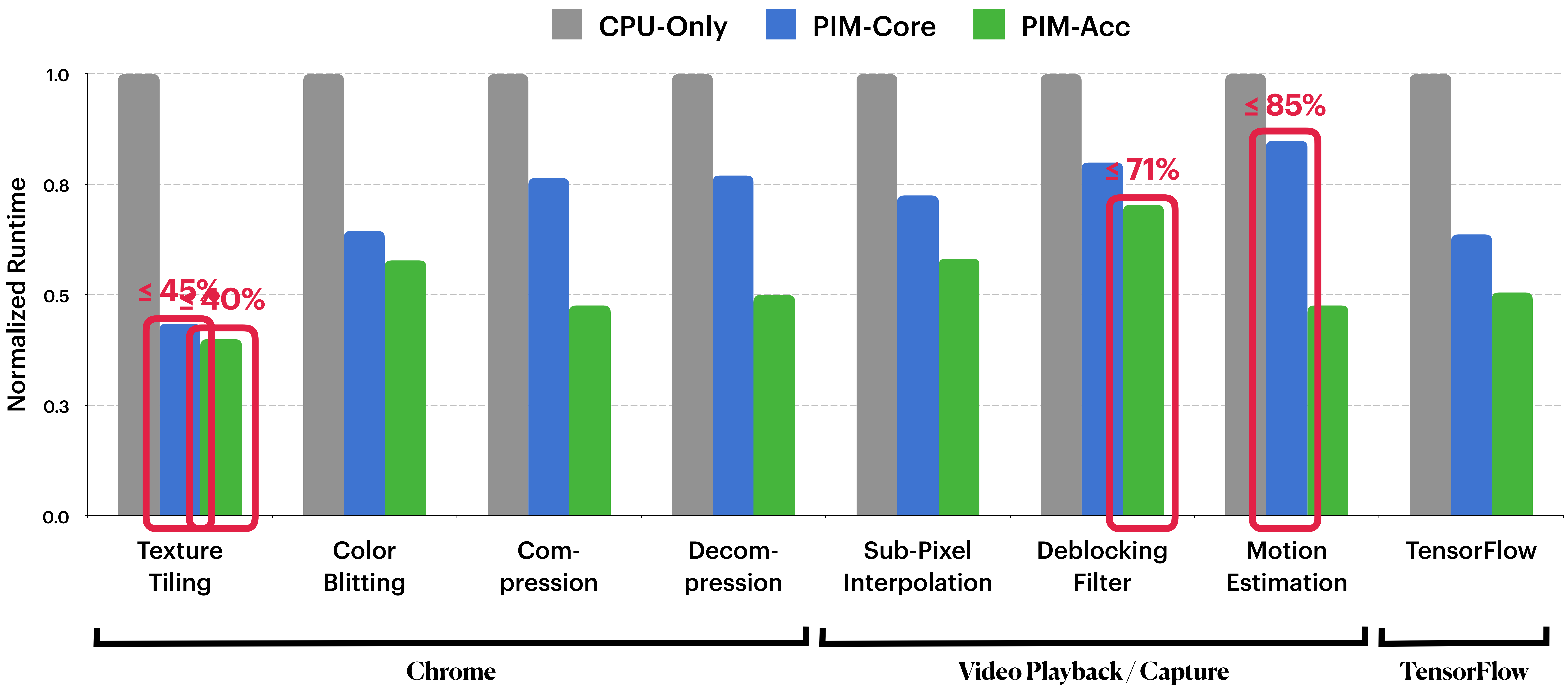
LPDDR3, 2GB, FR-FCFS scheduler

Energy 🚗



CPU-Only
PIM-Core
PIM-Acc
Normalized Energy
1
0.75
0.5
0.25
0
≤ 50%
≤ 30%
≤ 27%
≤ 60%
Texture Tiling
Color Blitting
Com-pression
Decom-pression
Packing
Quantization
Sub-Pixel Interpolation
Deblocking Filter
Motion Estimation
Chrome
TensorFlow
Video Playback / Capture

Runtime



CPU-Only
PIM-Core
PIM-Acc
Normalized Runtime
1.0
0.8
0.5
0.3
0.0
≤ 45%
≤ 40%
≤ 71%
≤ 85%
Texture Tiling
Color Blitting
Com-pression
Decom-pression
Sub-Pixel Interpolation
Deblocking Filter
Motion Estimation
TensorFlow
Chrome
Video Playback / Capture
TensorFlow

Conclusion

Want to make Google devices energy efficient?

But: Tight chip area budget / Tight thermal budget

Realize that it's all data movements

62.7% of system energy

Realize that it's all in simple algorithms

primitives like add, multiply, shift

Accelerate by processing-in-memory

reduces energy consumption by 55.4%

reduces running time by 54.2%

Profit

Free-up area
Make things fast
Free-up energy budget

Outline

Paper presentation

1. Introduction and Background
2. Methodology
3. Workload Analysis
4. Results

Analysis

5. **Strengths**
6. Weaknesses
7. Takeaways
8. Discussion

Strengths

Breadth of exploration

Chrome, **TensorFlow**, **VP9** Encoder (SW + HW), **VP9** Decoder (SW + HW)

Depth of exploration

1. Each workload is **thoroughly analyzed**
2. Each workload not only get its own **PiM feasibility** analysis but also a PiM implementation!
3. Each workload's PiM implementation is **thoroughly analyzed**

First paper to **comprehensively profile and analyze** popular google workloads

Outline

Paper presentation

1. Introduction and Background
2. Methodology
3. Workload Analysis
4. Results

Analysis

5. Strengths
6. **Weaknesses**
7. Takeaways
8. Discussion

Weaknesses

Evaluation methodology vs. analysis methodology

The **evaluation methodology** and the **analysis methodology** do not match.

1. Analysis main CPU has **2 cores** while evaluation CPU has **4 cores**
2. Analysis based on **HMC memory** while evaluation based on **HBM memory**

Lacks comparison against a **GPU** for Chrome, or a **Neural Network Accelerator** for TensorFlow? *(Snapdragon Neural Processing Engine in Motoral phones anounced in 2017)*

Lacks comparison against **accelerators on the SoC** for Chrome and TensorFlow?

Lacks comparison against a 16 core system on the SoC:

We want to **decouple** the performance from PiM itself and the simple fact of having more cores

The baseline memory **uses LPDDR3 (2GB/s)**: why not use the **off-chip bandwidth of 3d-stacked memory (32 GB/s)**?

Outline

Paper presentation

1. Introduction and Background
2. Methodology
3. Workload Analysis
4. Results

Analysis

5. Strengths
6. Weaknesses
7. Takeaways
8. Discussion

Key Takeaways

A lot of the popular workloads involve very **basic operations** that can be accelerated with PiM

Data movement problems are **hard to solve** with current methods but powerfully solved with PiM

Improvements by 2x might not seem monumental, but we are talking about the **most used and optimized algorithms of our times**

Matrix-multiply **optimizations** (quantization, packing) can be **optimized** further with PiM!

GPU is not necessarily the solution to all **graphics** problems!

Outline

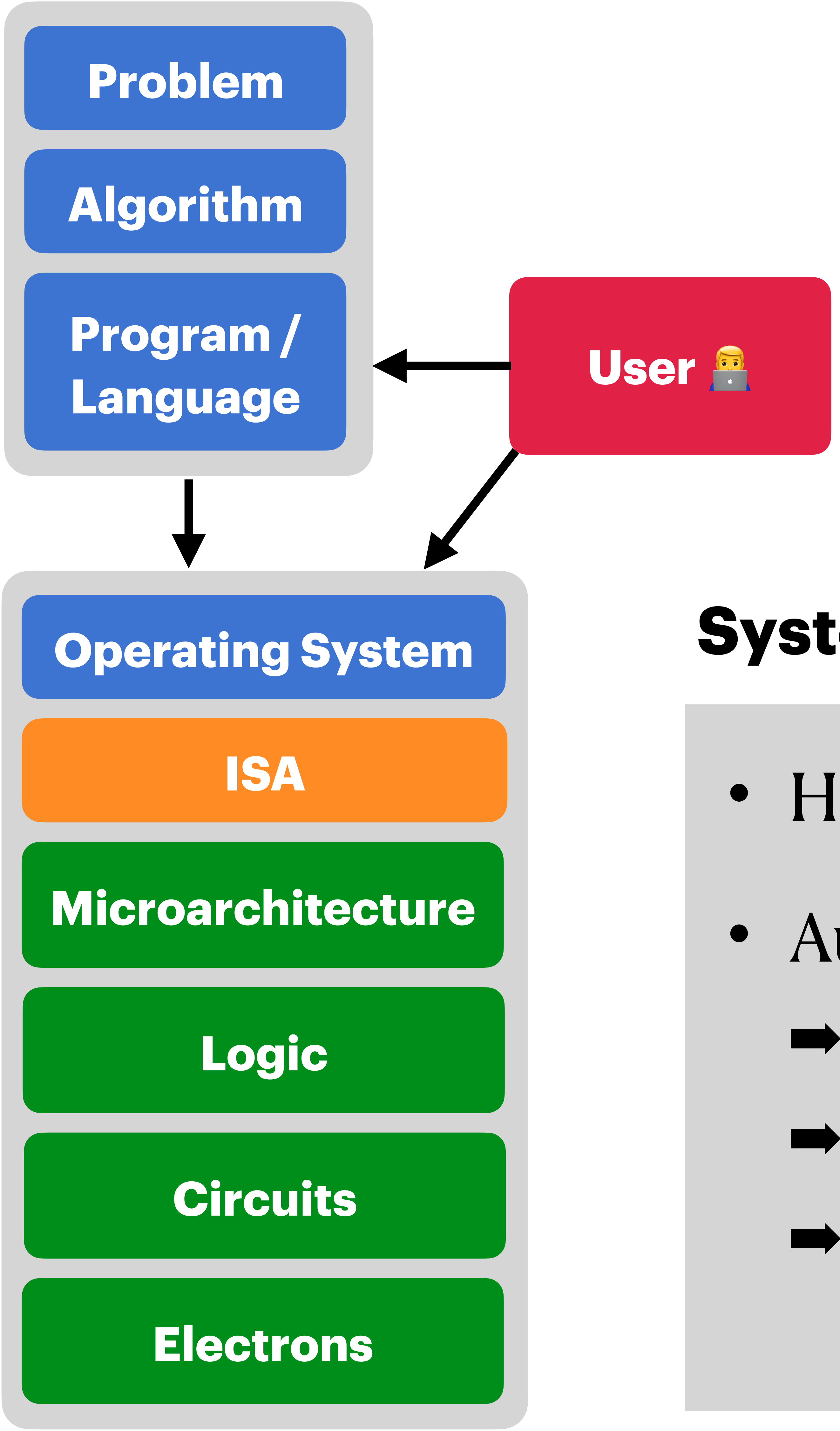
Paper presentation

1. Introduction and Background
2. Methodology
3. Workload Analysis
4. Results

Analysis

5. Strengths
6. Weaknesses
7. Takeaways
8. Discussion

Discussion: Opening Questions

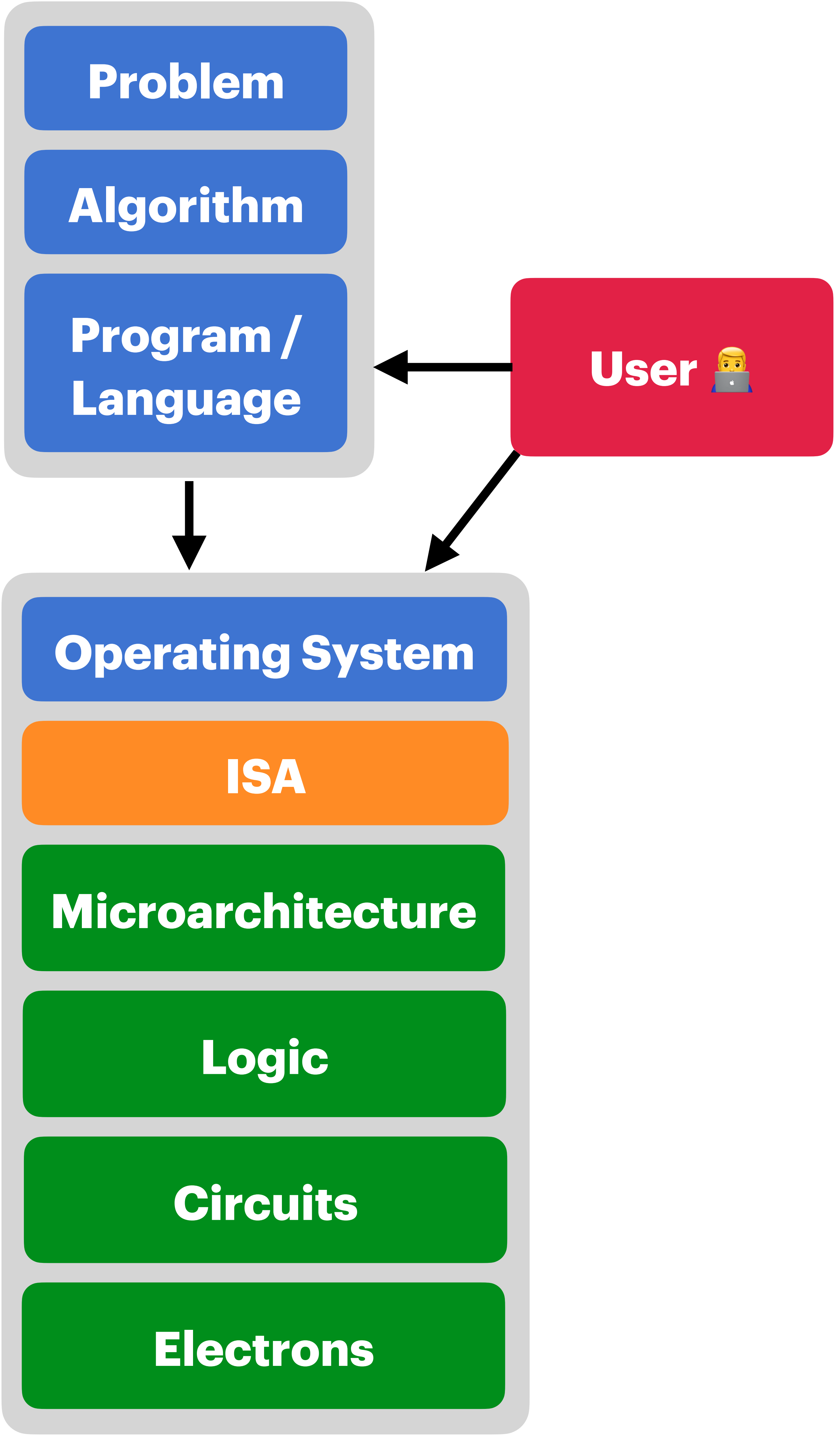


System integration: opening questions

- How to design compiler to deal with PiM workloads?
- Automatically identify portions of code that can be offloaded to PiM Core?
  - ➡ Is it an OS Job?
  - ➡ Is it a compiler Job?
  - ➡ Is it the programmer's job?

  If so, think about programming a 1000-core machine in the future!

Discussion: Runtime/Static Analysis



PIMProf (2022)

PIMProf: An Automated Program Profiler for Processing-in-Memory Offloading Decisions

Yizhou Wei*, Minxuan Zhou†, Sihang Liu*, Korakit Seemakhupt*, Tajana Rosing†, and Samira Khan*
*University of Virginia, †University of California San Diego
Email: {yizhouwei, sihangliu, korakit, samirakhan}@virginia.edu, {miz087, tajana}@ucsd.edu

TOM (2016)

Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh‡ Eiman Ebrahimi† Gwangsun Kim* Niladrish Chatterjee† Mike O'Connor†
Nandita Vijaykumar‡ Onur Mutlu§‡ Stephen W. Keckler†
‡Carnegie Mellon University †NVIDIA *KAIST §ETH Zürich

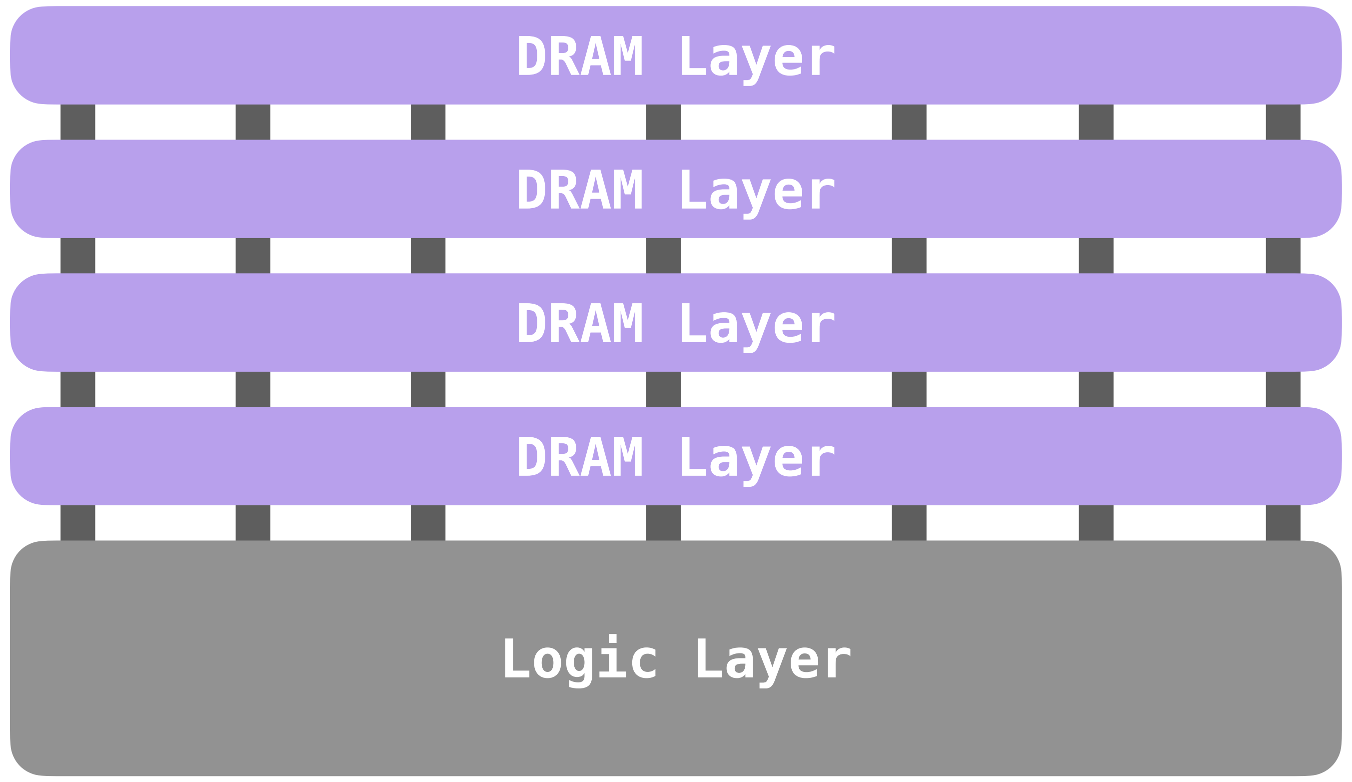
PIM-Enabled Instructions (2015)

PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture

Junwhan Ahn Sungjoo Yoo Onur Mutlu† Kiyoung Choi
junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr
Seoul National University †Carnegie Mellon University

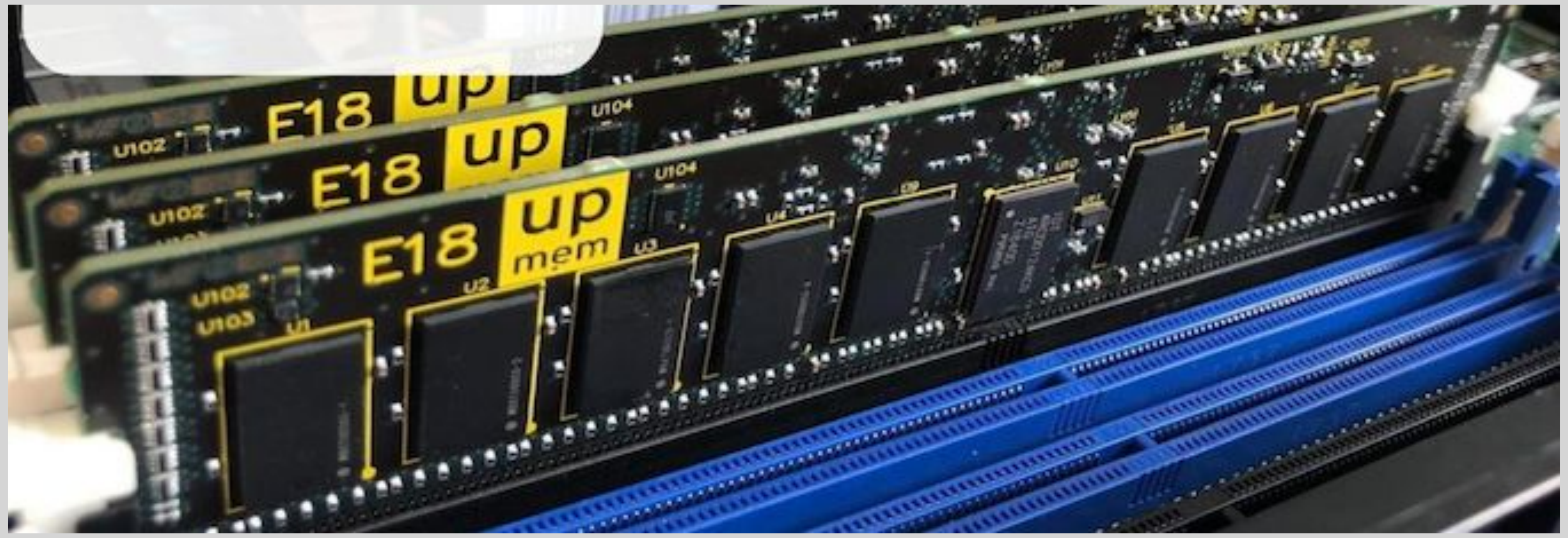
Discussion: Processing-in-Memory

This work: 3D-stacked memory



What about other Processing-in-Memory devices?

up mem



RowClone (2013)

RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization

Vivek Seshadri vseshadr@cs.cmu.edu
Yoongu Kim yoongukim@cmu.edu
Chris Fallin* cfallin@c1f.net
Donghyuk Lee donghyuk1@cmu.edu
Rachata Ausavarungnirun rachata@cmu.edu
Gennady Pekhimenko gpekhime@cs.cmu.edu
Yixin Luo yixinluo@andrew.cmu.edu
Onur Mutlu onur@cmu.edu
Phillip B. Gibbons† phillip.b.gibbons@intel.com
Michael A. Kozuch† michael.a.kozuch@intel.com
Todd C. Mowry tcm@cs.cmu.edu
Carnegie Mellon University †Intel Pittsburgh

Can we use Processing-using-Memory?

AMBIT (2017)

Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology

Vivek Seshadri[1,5] Donghyuk Lee[2,5] Thomas Mullins[3,5] Hasan Hassan[4] Amirali Boroumand[5]
Jeremie Kim[4,5] Michael A. Kozuch[3] Onur Mutlu[4,5] Phillip B. Gibbons[5] Todd C. Mowry[5]

[1]Microsoft Research India [2]NVIDIA Research [3]Intel [4]ETH Zürich [5]Carnegie Mellon University

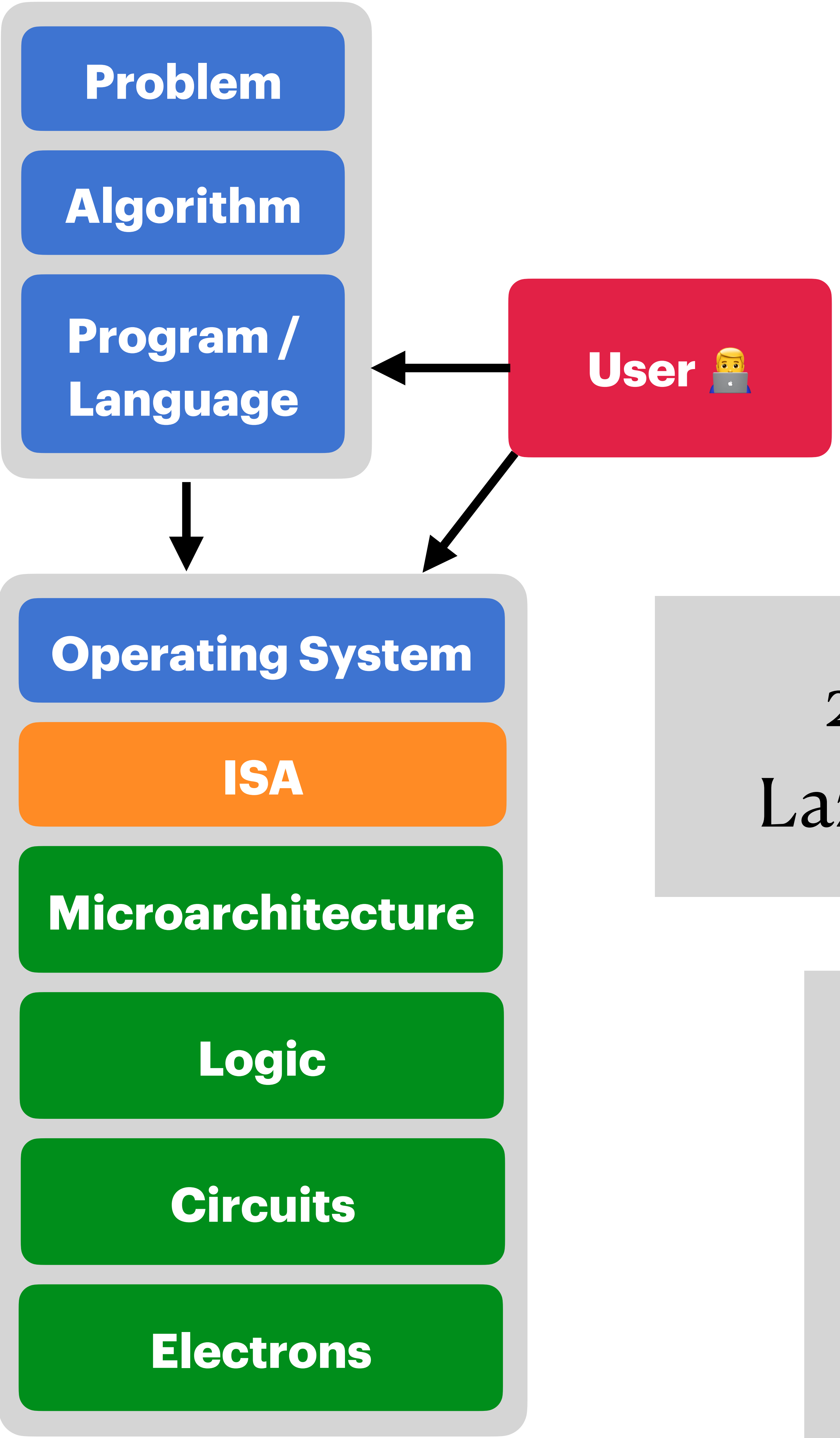
SIMDRAM (2021)

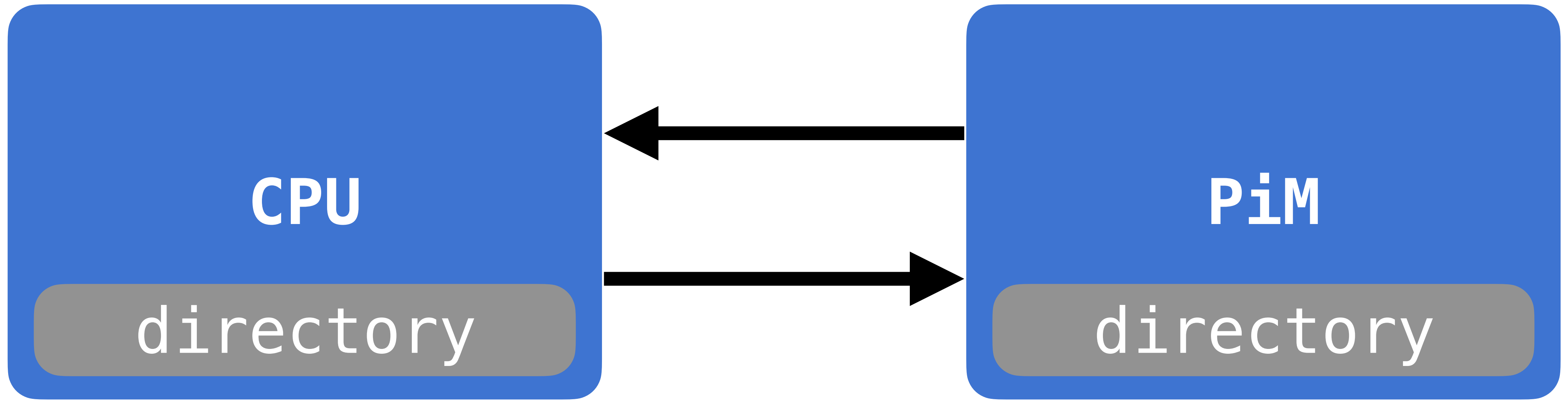
SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM

*Nastaran Hajinazar[1,2] Nika Mansouri Ghiasi[1]
*Geraldo F. Oliveira[1] Minesh Patel[1] Juan Gómez-Luna[1]
Sven Gregorio[1] Mohammed Alser[1] Onur Mutlu[1]
João Dinis Ferreira[1] Saugata Ghose[3]

[1]ETH Zürich [2]Simon Fraser University [3]University of Illinois at Urbana–Champaign

Discussion: Coherence





However: Off-chip coherence traffic is expensive:
How do we solve the data coherence problem?

2016
LazyPIM

LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory

Amirali Boroumand†, Saugata Ghose†, Minesh Patel†, Hasan Hassan†§, Brandon Lucia†, Kevin Hsieh†, Krishna T. Malladi⋆, Hongzhong Zheng⋆, and Onur Mutlu‡†

†Carnegie Mellon University ⋆Samsung Semiconductor, Inc. §TOBB ETÜ ‡ETH Zürich

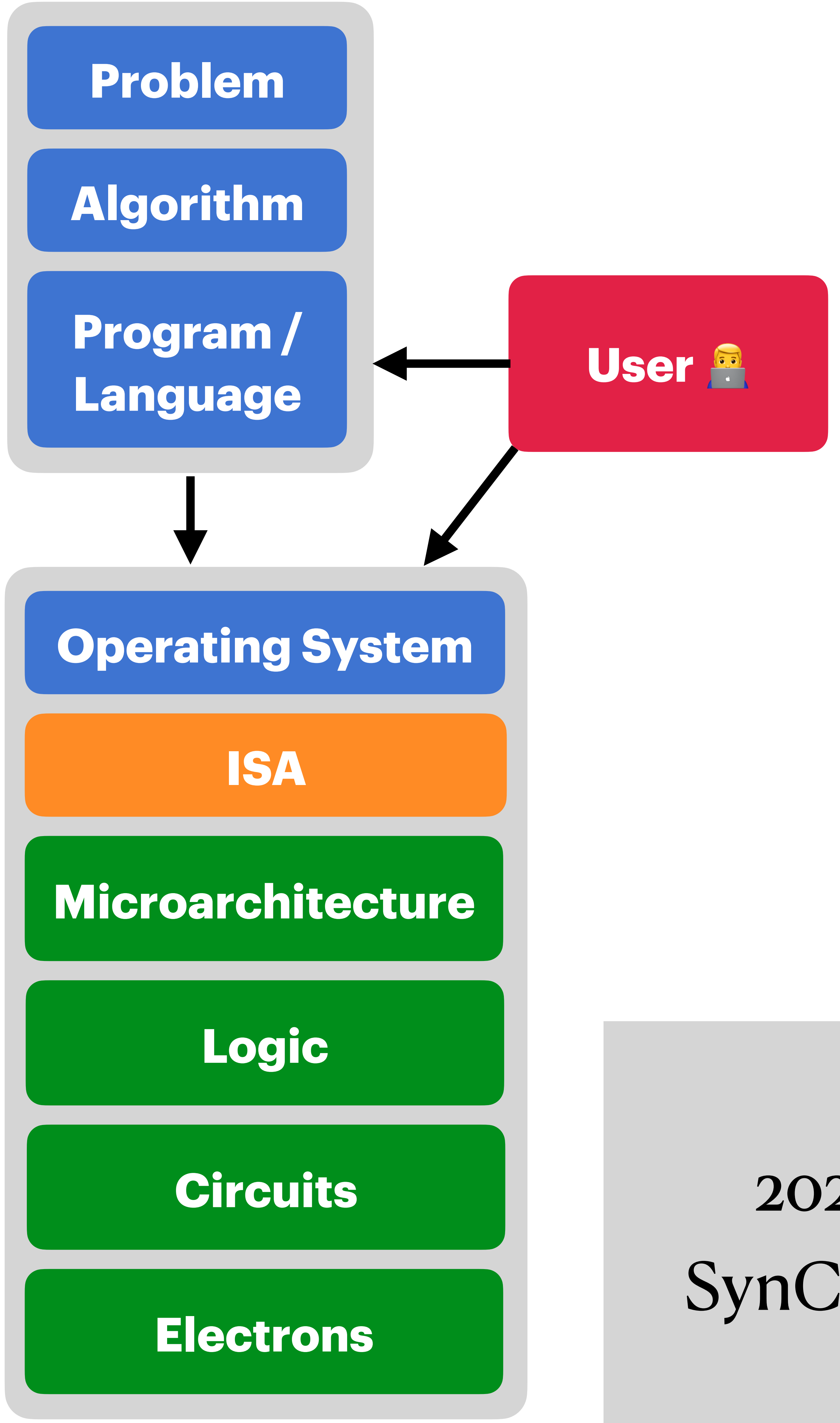
2019
CoNDA

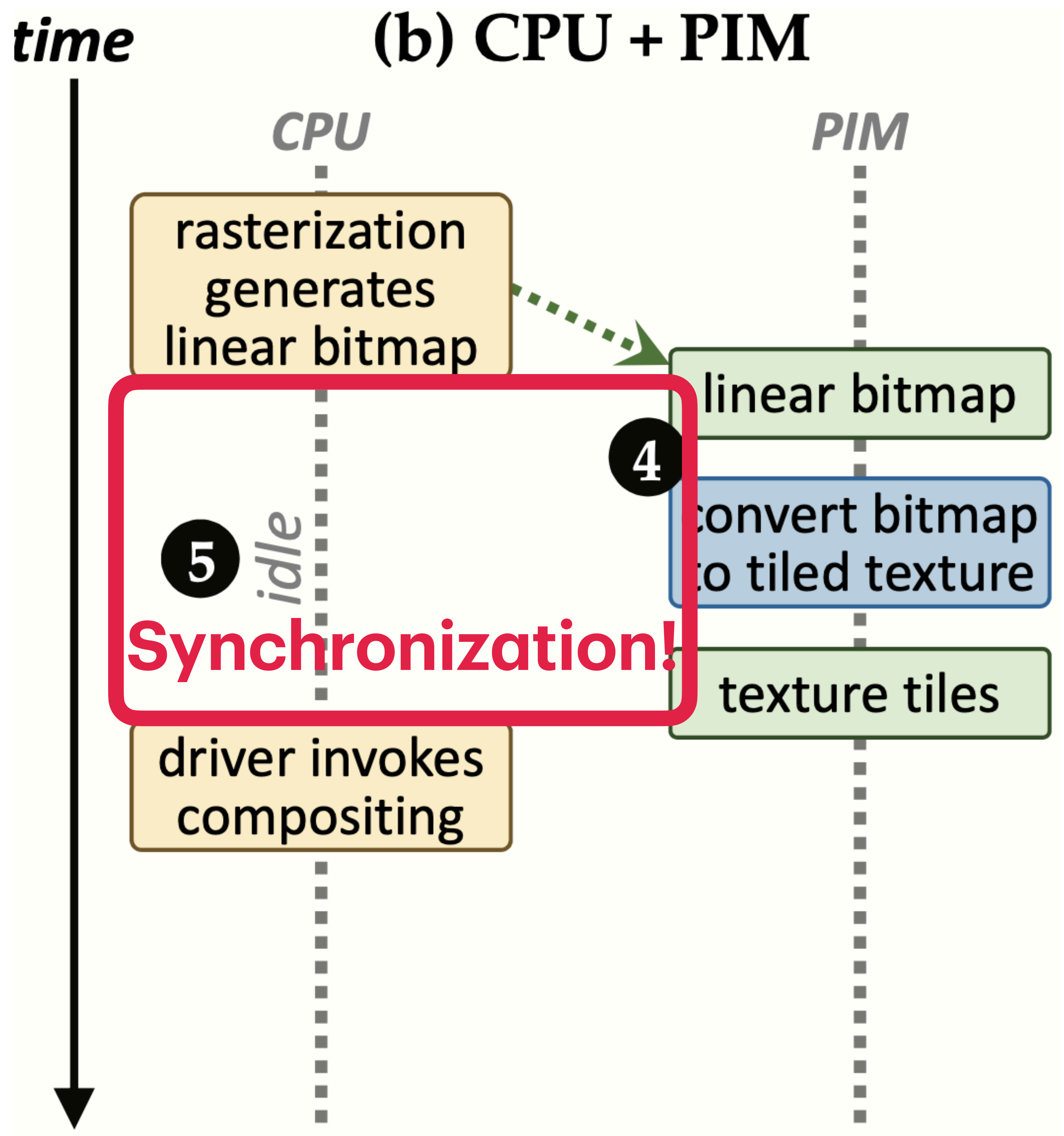
CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators

Amirali Boroumand† Saugata Ghose† Minesh Patel⋆ Hasan Hassan⋆ Brandon Lucia† Rachata Ausavarungnirun†‡ Kevin Hsieh† Nastaran Hajinazar◇† Krishna T. Malladi§ Hongzhong Zheng§ Onur Mutlu⋆†

†Carnegie Mellon University ⋆ETH Zürich ‡KMUTNB ◇Simon Fraser University §Samsung Semiconductor, Inc.

Discussion: Synchronization





2021
SynCron

***SynCron*: Efficient Synchronization Support for Near-Data-Processing Architectures**

Christina Giannoula[†‡] Nandita Vijaykumar[*‡] Nikela Papadopoulou[†] Vasileios Karakostas[†] Ivan Fernandez[§‡]
Juan Gómez-Luna[‡] Lois Orosa[‡] Nectarios Koziris[†] Georgios Goumas[†] Onur Mutlu[‡]
[†]*National Technical University of Athens* [‡]*ETH Zürich* [*]*University of Toronto* [§]*University of Malaga*

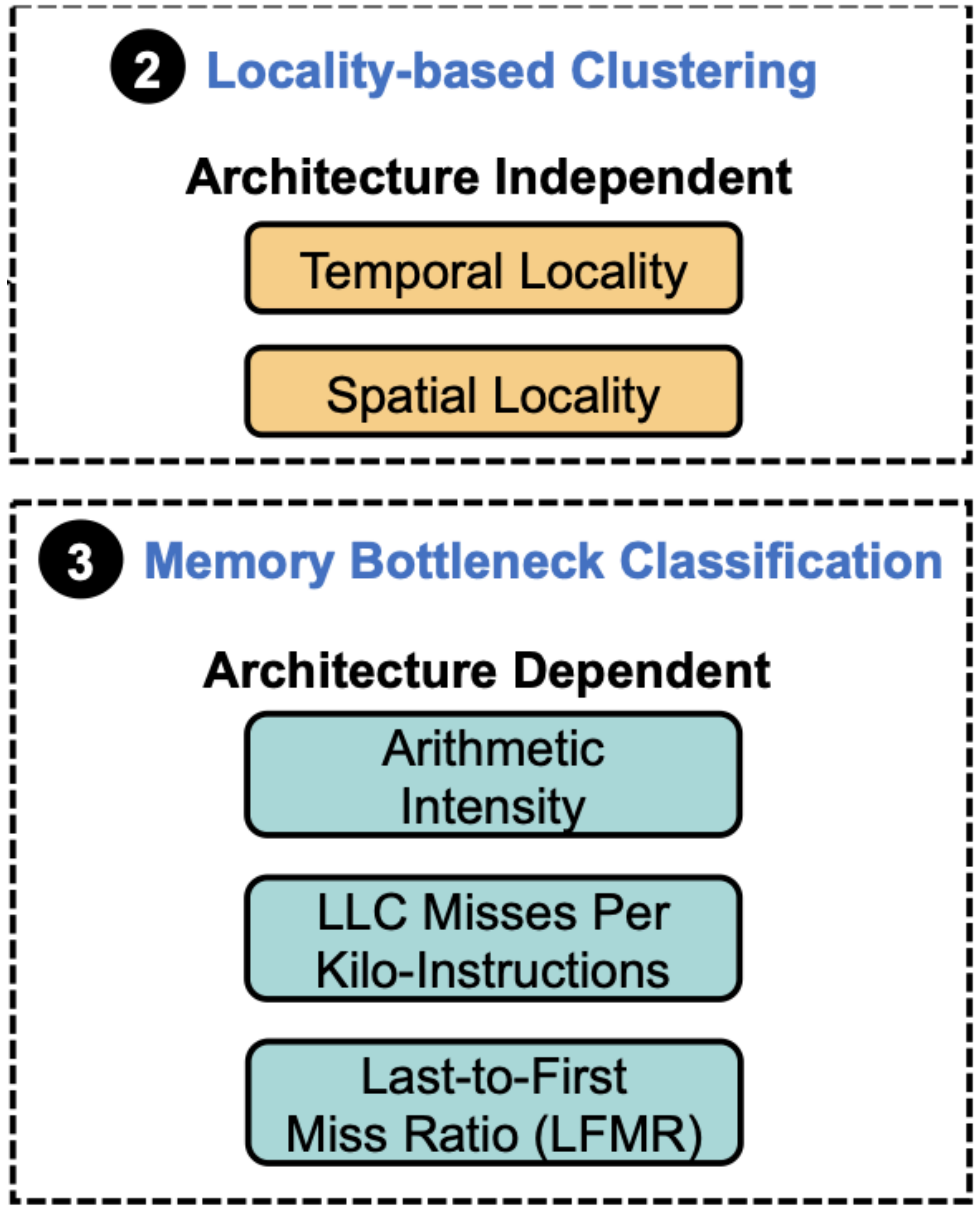
Discussion: Research Methodology

This work: use energy as the primary target metric

DAMOV (2021)

DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
LOIS OROSA, ETH Zürich, Switzerland
SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA
NANDITA VIJAYKUMAR, University of Toronto, Canada
IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland
MOHAMMAD SADROSADATI, ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland



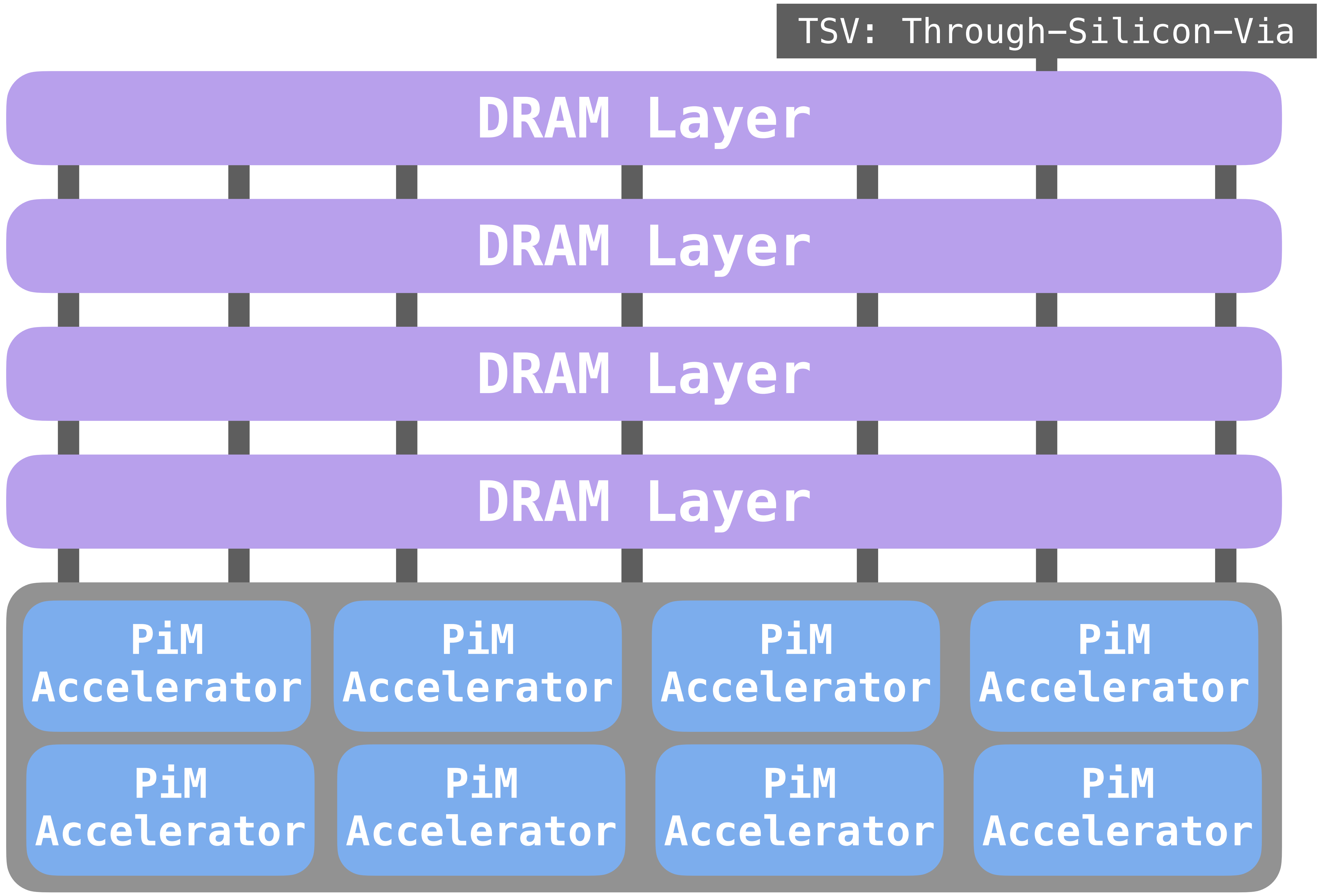
Backup slides

Discussion: PiM in the market

We have to choose what to put in PiM accelerators

Tradeoff between:

1. Generality (Accelerator reuse)
2. Performance
3. Cost



Today PiM-enabled memory is expensive

Is there a future where even cheap mobile devices are PiM-enabled? When?

Executive Summary

Problems and motivation

- Mobile consumer devices are subject to tight circuit area, thermal heat and energy budget.

➡ *How to make Google Consumer Devices more energy-efficient?* 🚙 = 🏎️

Key ideas and insights

- Among popular workloads, data movement is a prime contributor to total system energy expenditure.
- A few functions are responsible for a large chunk of the total energy cost.

Mechanisms and implementation

- Analyzing data movement related costs in Google Chrome, TensorFlow, Video Playback and Capture.
- Investigation of efficiency gains from using Processing-in-memory (PiM).
- Determining for which workloads it is a good idea to use PiM, and which type of PiM to use.

Results

- Reduces energy costs by an average of 55.4% across tested workloads. 🚙
- Reduces execution time by an average of 54.2%. 🏎️

Executive Summary

Problem and motivation

Tight energy / chip area / thermal budgets

Want to make Google devices energy efficient?

Observation

Realize that it's all data movements

62.7% of system energy

Realize that it's all in simple algorithms

primitives like add, multiply, shift

Contribution and result

Accelerate by processing-in-memory

reduces energy consumption by 55.4%

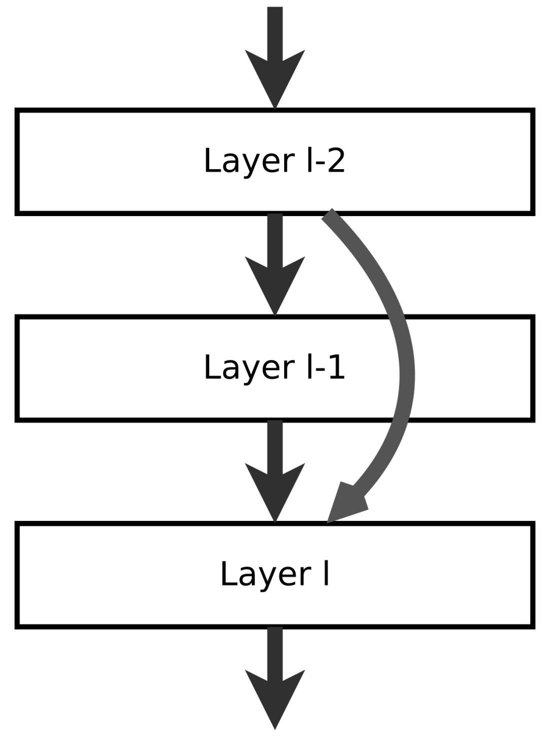
reduces running time by 54.2%

Profit

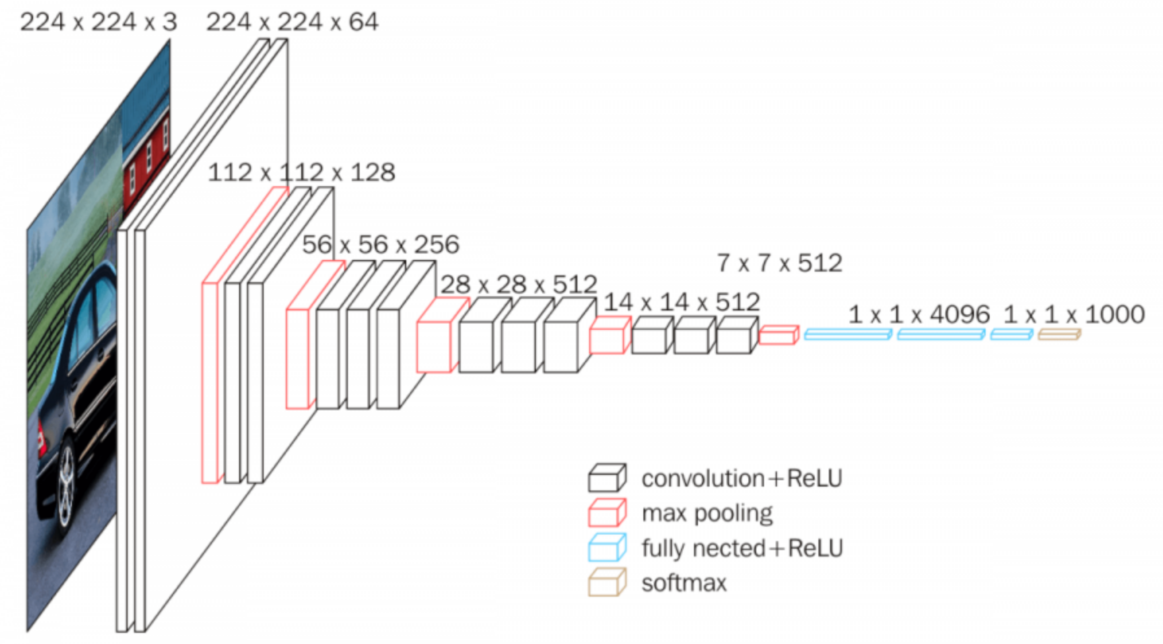
Free-up area
Enable more power
Free-up energy budget

Neural Network Architectures analyzed

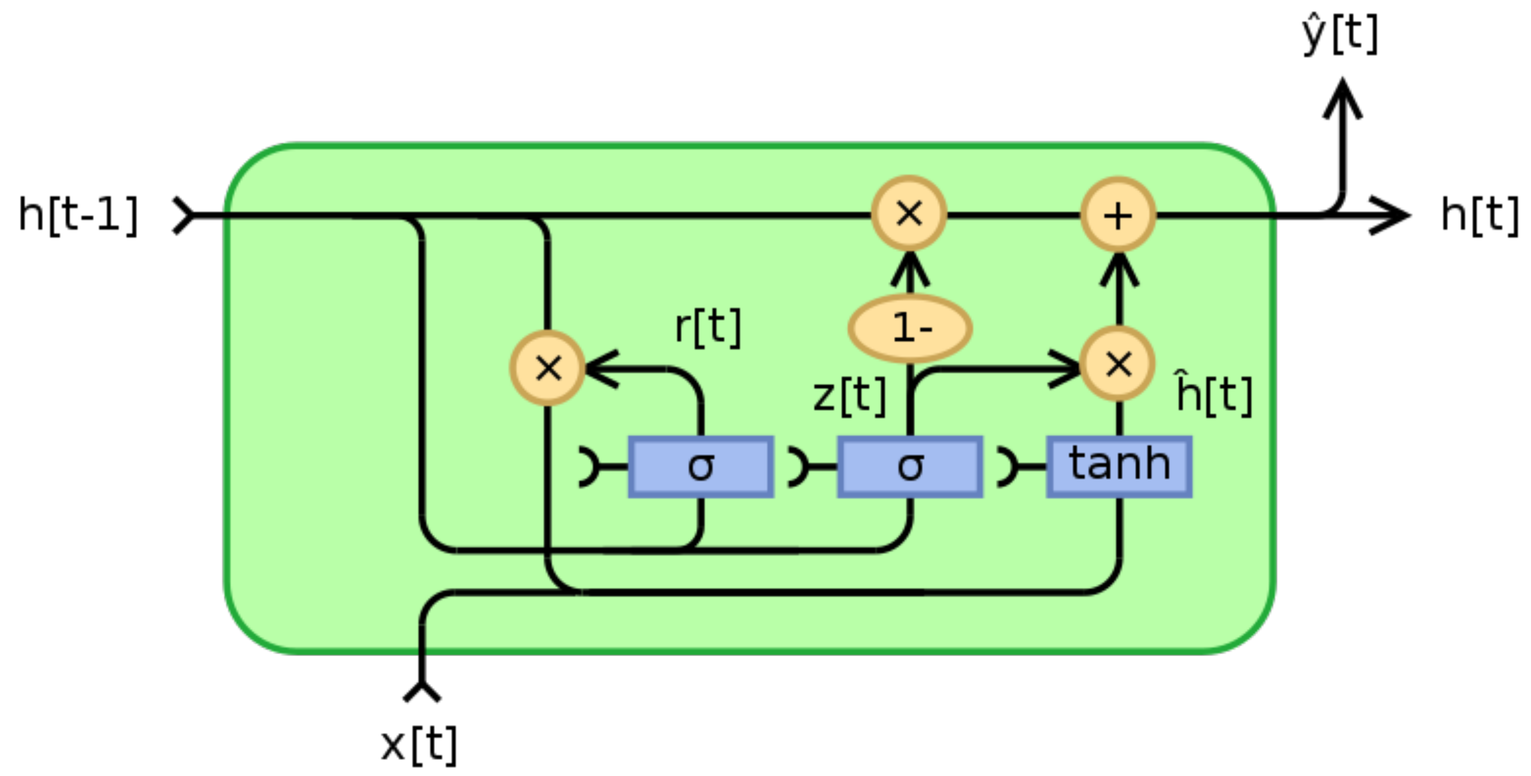
ResNet (2015)



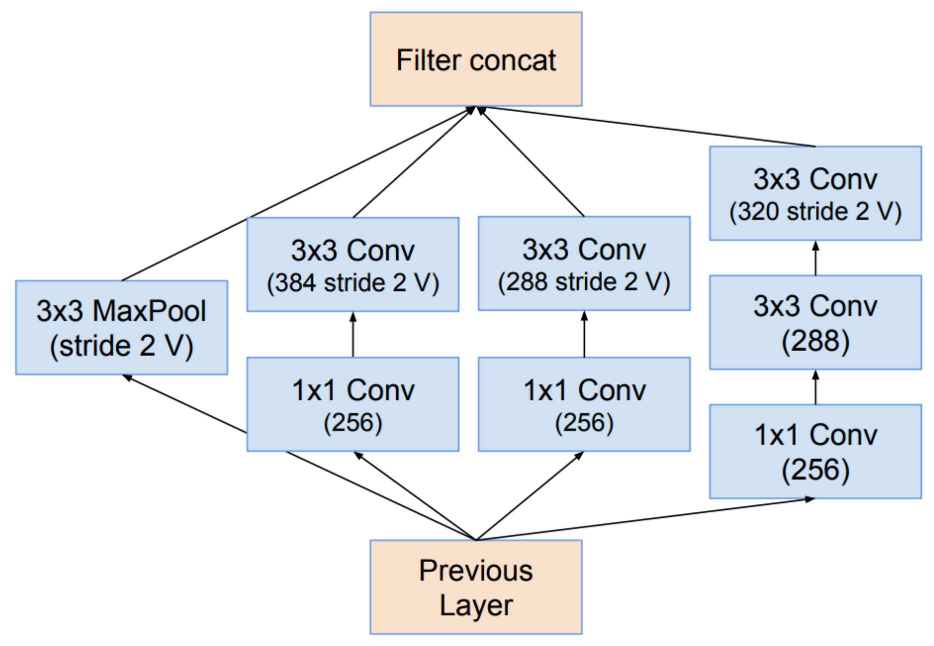
VGG-19 (2014)



GRU (2014)



Inception-ResNet (2016)



VP9 Video Playback

What is video playback anyway?

➡ A decoder: The VP9 decoder!

Decompresses and decodes the raw streaming video data and renders it on the device.

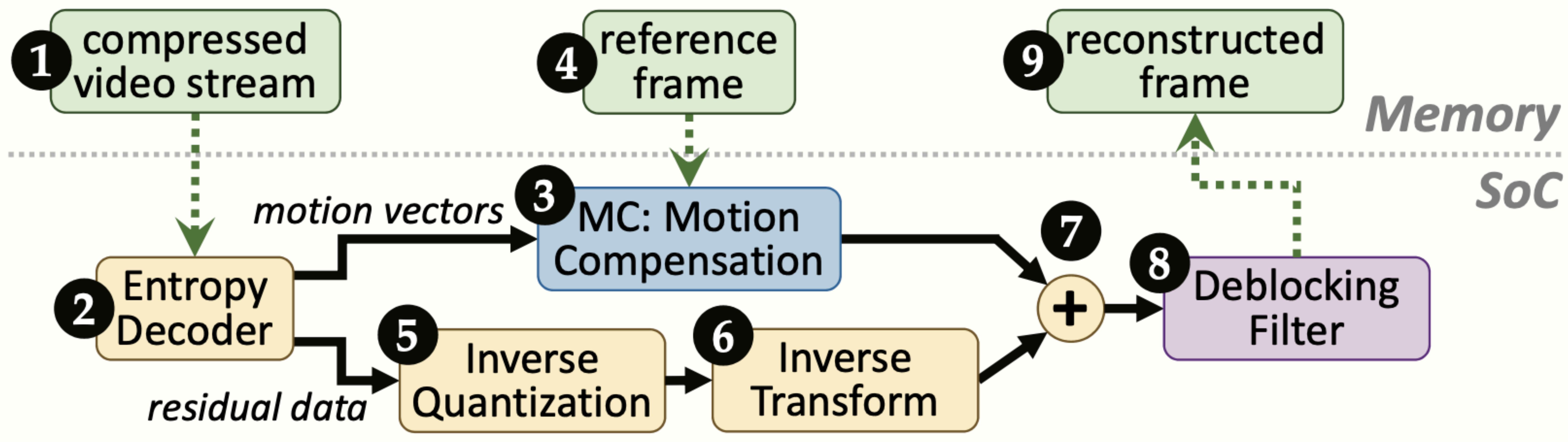


Figure 9. General overview of the VP9 decoder.

VP9 Video Playback

Step-by-step explanation

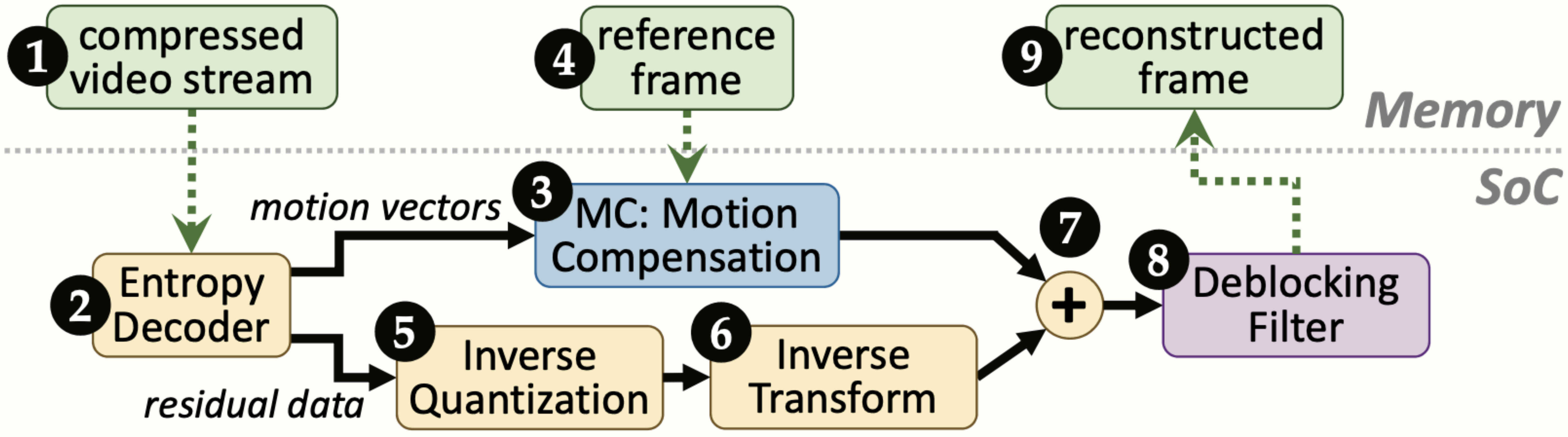
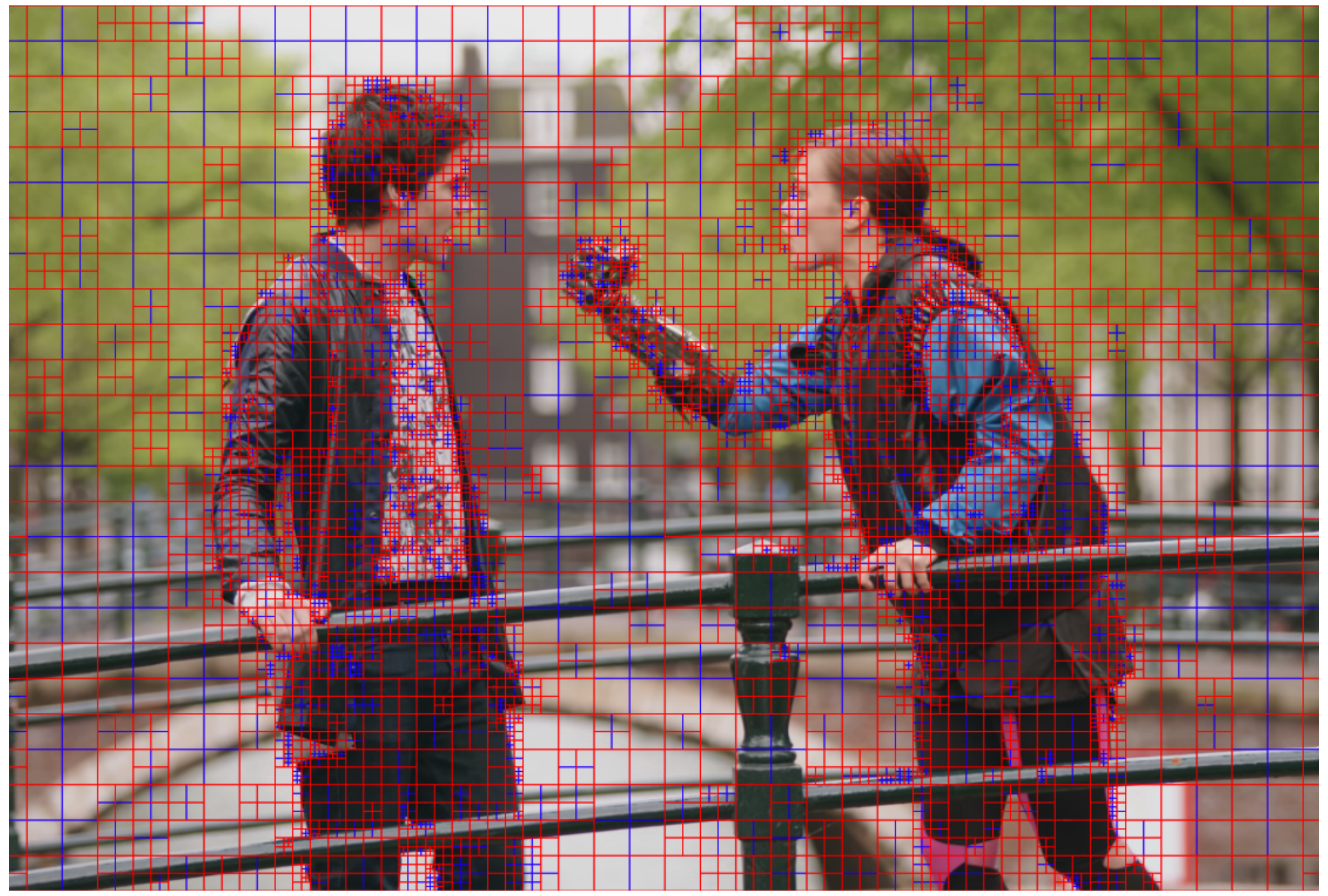


Figure 9. General overview of the VP9 decoder.

❸ macro-blocks



❸ motion vectors:

resolution can be as low as 1/8 of a pixel!

➡ Sub-pixel interpolation

❽ deblocking filter

Fig. 1. Deblocking a video frame[3]

VP9 Energy Analysis

Energy Analysis: VP9 software decoder

Using 4K resolution (3840x2160-pixel)

- 53.4% of energy spent on MC.
- 37.5% of energy spent on Sub-Pixel Interpolation.

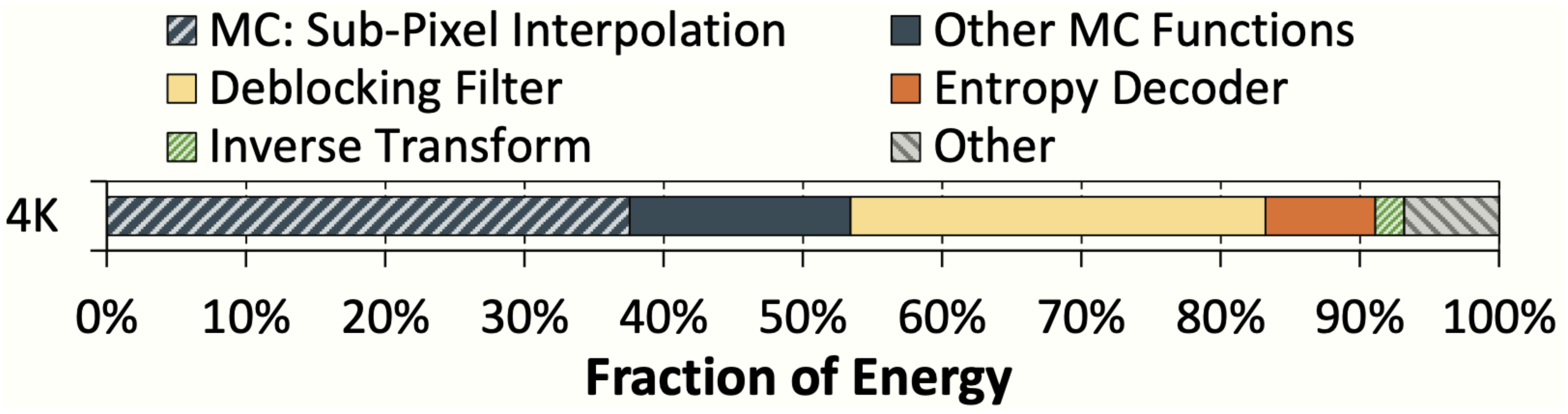


Figure 10. Energy analysis of VP9 software decoder.

- 63.5% of total energy spent on data movement
- 80.4% of which is provoked by MC.
- 42.6% of total data movement happens in Sub-Pixel Interpolation.

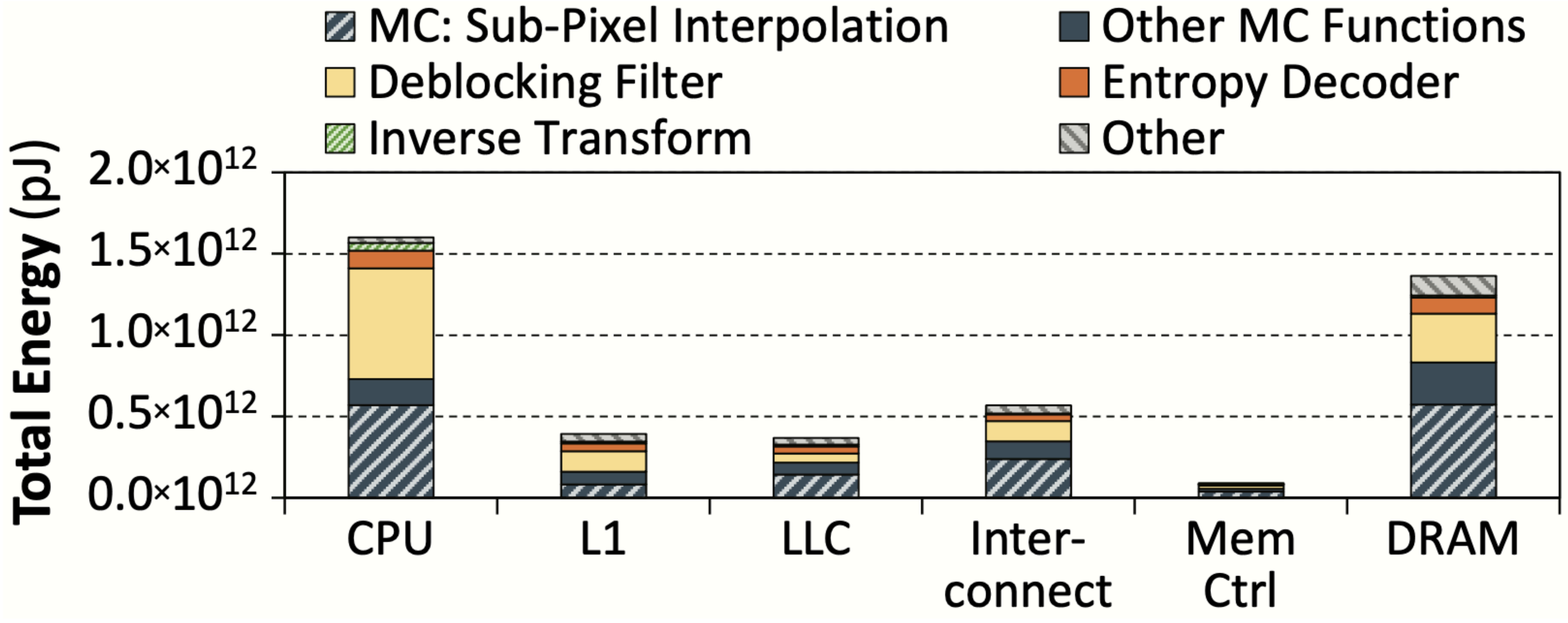
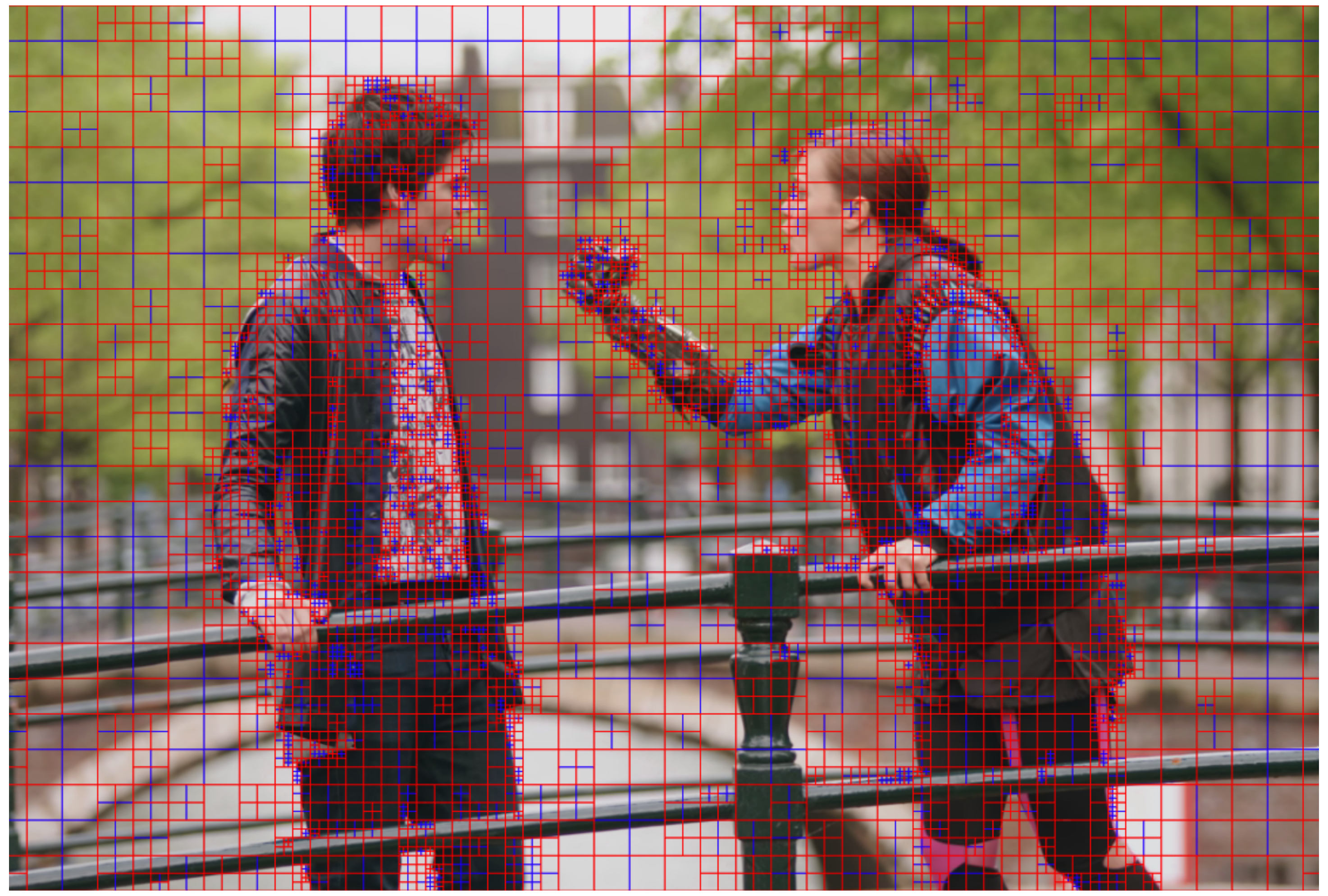


Figure 11. Energy breakdown of VP9 software decoder.

VP9 PiM Feasibility: Sub-Pixel Interpolation



CPU problems:

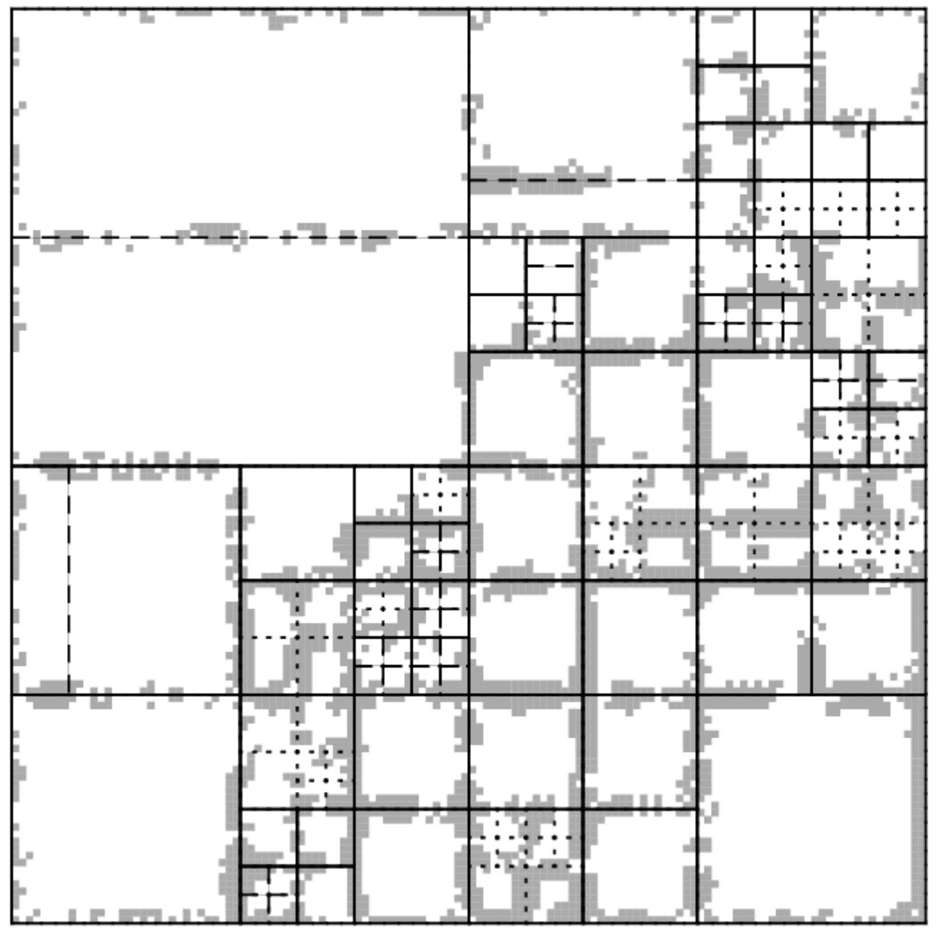
- Each subpixel interpolation needs multiple pixels to be fetched from memory (11 by 11 pixels at worse)
- Motion vectors can point to any point in reference frame: poor data locality!
- 65% of data movement between DRAM and CPU!

PiM solution:

- Filters used only require addition, multiplication and shifting.
- PiM accelerator would only use 0.25 mm^2 (6% per vault)

VP9 PiM Feasibility: Deblocking Filter

Fig. 1. Deblocking a video frame[3]



CPU problems:

- Low-pass filter on each edge between two superblocks.
- Poor cache locality.
- 71.1% of data movement happens off-chip

PiM solution:

- Simple lowpass requires only arithmetic and bitwise operations.
- PiM accelerator would only use 0.12 mm^2 (3.4% per vault)

VP9 Video Playback

A new challenger has arrived: VP9 hardware decoder

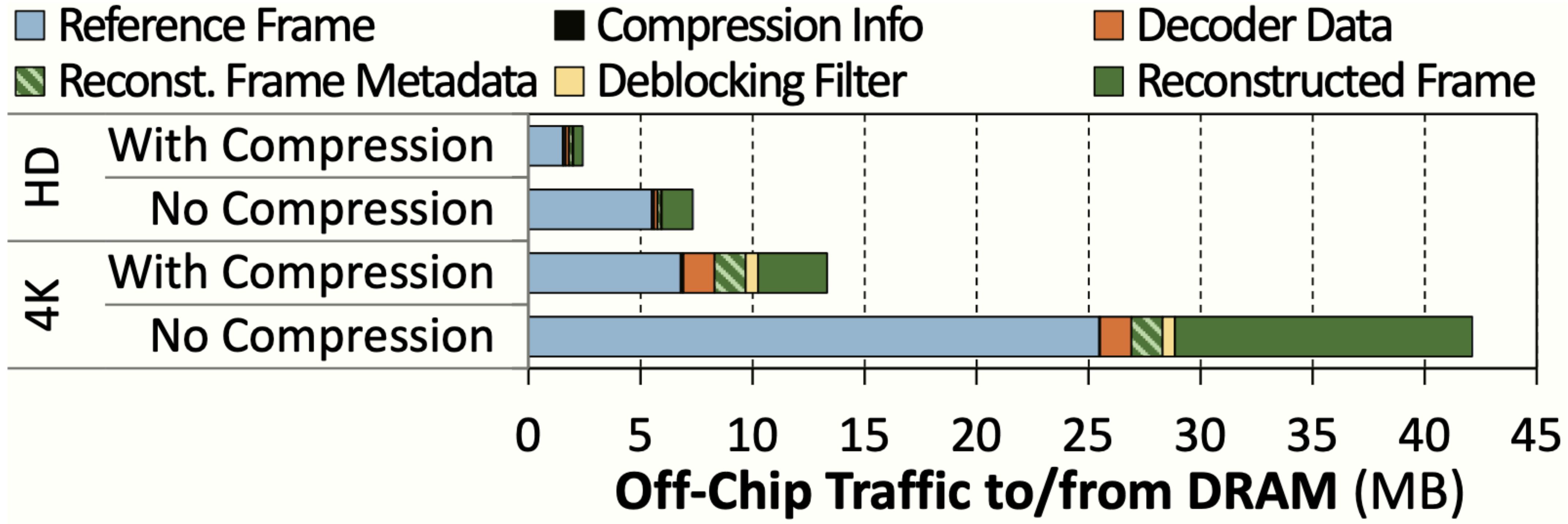


Figure 12. Off-chip traffic breakdown of VP9 hardware decoder.

VP9 Video Playback

Back to PiM Feasibility. Does it hold against the hardware decoder?

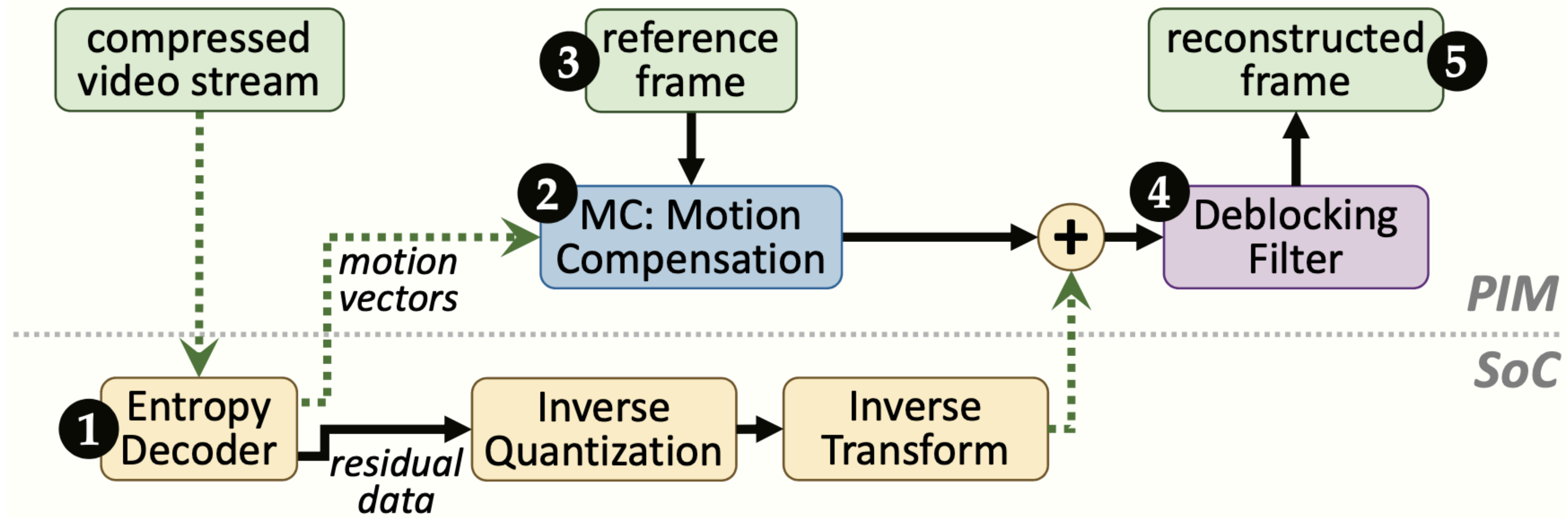


Figure 13. Modified VP9 decoder with in-memory MC.