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

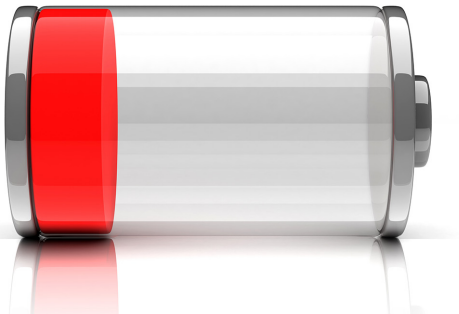
SAFARI Carnegie Mellon Google

SAMSUNG SEOUL NATIONAL UNIVERSITY ETH Zürich

Consumer Devices


Consumer devices are everywhere!

Energy consumption is a first-class concern in consumer devices



Popular Google Consumer Workloads

Chrome

Google's web browser

TensorFlow Mobile

Google's machine learning framework

VP9

Video Playback

Google's video codec

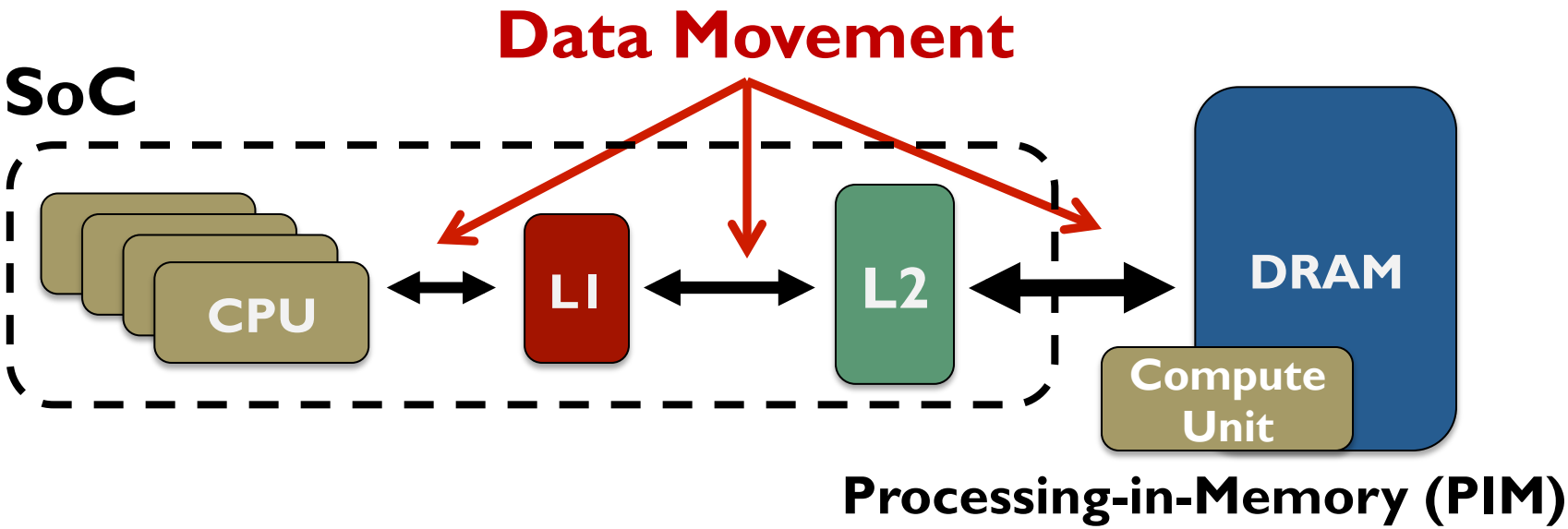
VP9

Video Capture

Google's video codec

Energy Cost of Data Movement

1st key observation: 62.7% of the total system energy is spent on data movement



Potential solution: move computation close to data

Challenge: limited area and energy budget

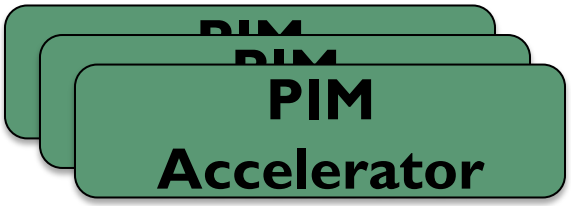
Using PIM to Reduce Data Movement

2nd key observation: a significant fraction of data movement often comes from simple functions

We can design lightweight logic to implement these *simple functions* in memory

Small embedded low-power core

Small fixed-function accelerators



Offloading to PIM logic reduces energy by 55.4% and improves performance by 54.2% on average

Goals

1. Understand the data movement related bottlenecks in modern consumer workloads

2. Analyze opportunities to mitigate data movement by using processing-in-memory (PIM)

3. Design PIM logic that can maximize energy efficiency given the limited area and energy budget in consumer devices

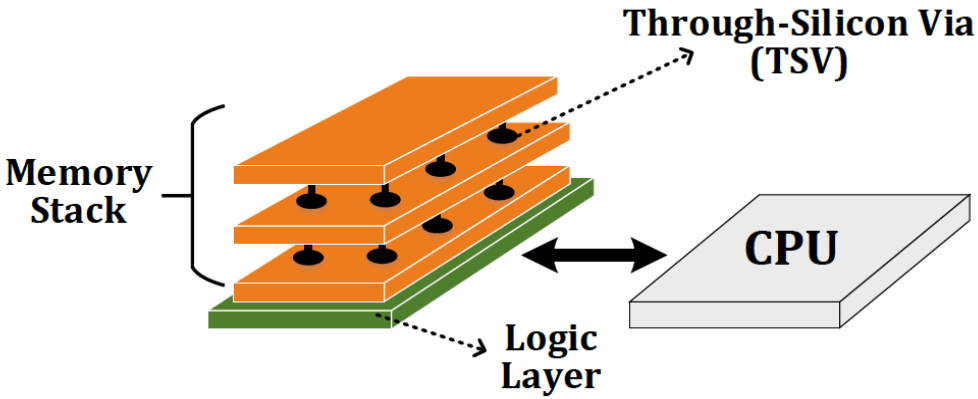
Outline

- Introduction
- **Background**
- Analysis Methodology
- Workload Analysis
- Evaluation
- Conclusion

Potential Solution to Address Data Movement

- **Processing-in-Memory (PIM)**
  - A potential solution to **reduce data movement**
  - **Idea:** move computation close to data
    - ✓ Reduces data movement
    - ✓ Exploits large in-memory bandwidth
    - ✓ Exploits shorter access latency to memory

- **Enabled by recent advances in 3D-stacked memory**



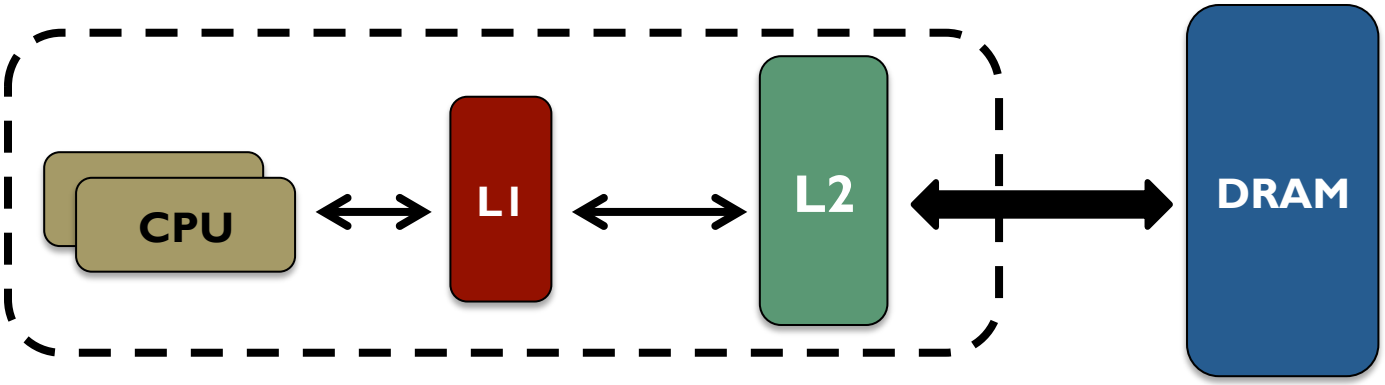
Outline

- Introduction
- Background
- **Analysis Methodology**
- Workload Analysis
- Evaluation
- Conclusion

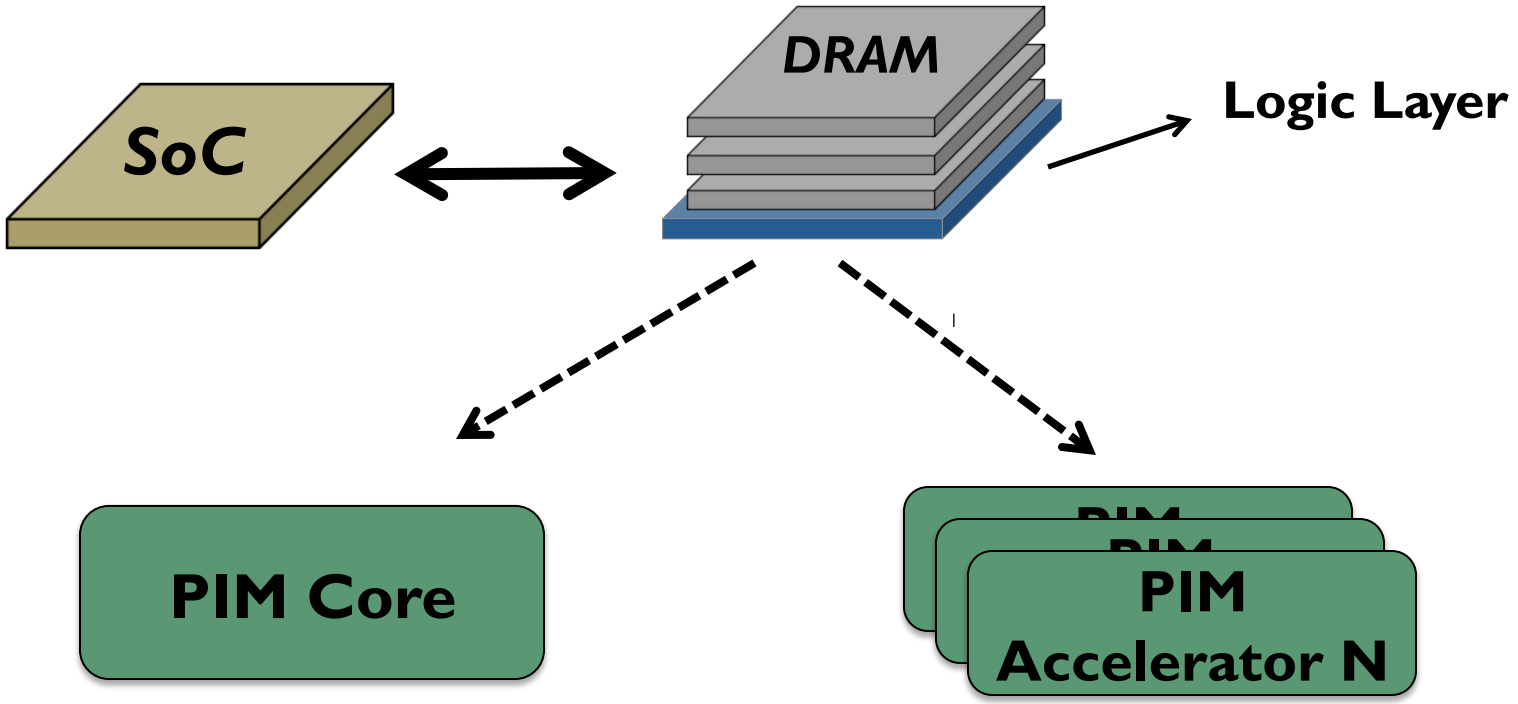
Workload Analysis Methodology

- **Workload Characterization**
  - Chromebook with an Intel Celeron SoC and 2GB of DRAM
  - Extensively use performance counters within SoC

- **Energy Model**
  - Sum of the energy consumption within the **CPU**, **all caches**, **off-chip interconnects**, and **DRAM**



PIM Logic Implementation



Customized embedded general-purpose core

No aggressive ILP techniques

256-bit SIMD unit

Small fixed-function accelerators

Multiple copies of customized in-memory logic unit

Workload Analysis

Chrome

Google's web browser

TensorFlow

Google's machine learning framework

Video Playback

Google's video codec

Video Capture

Google's video Codec

Workload Analysis

Chrome

Google's web browser

TensorFlow

Google's machine learning framework

VP9

YouTube

Video Playback

Google's video codec

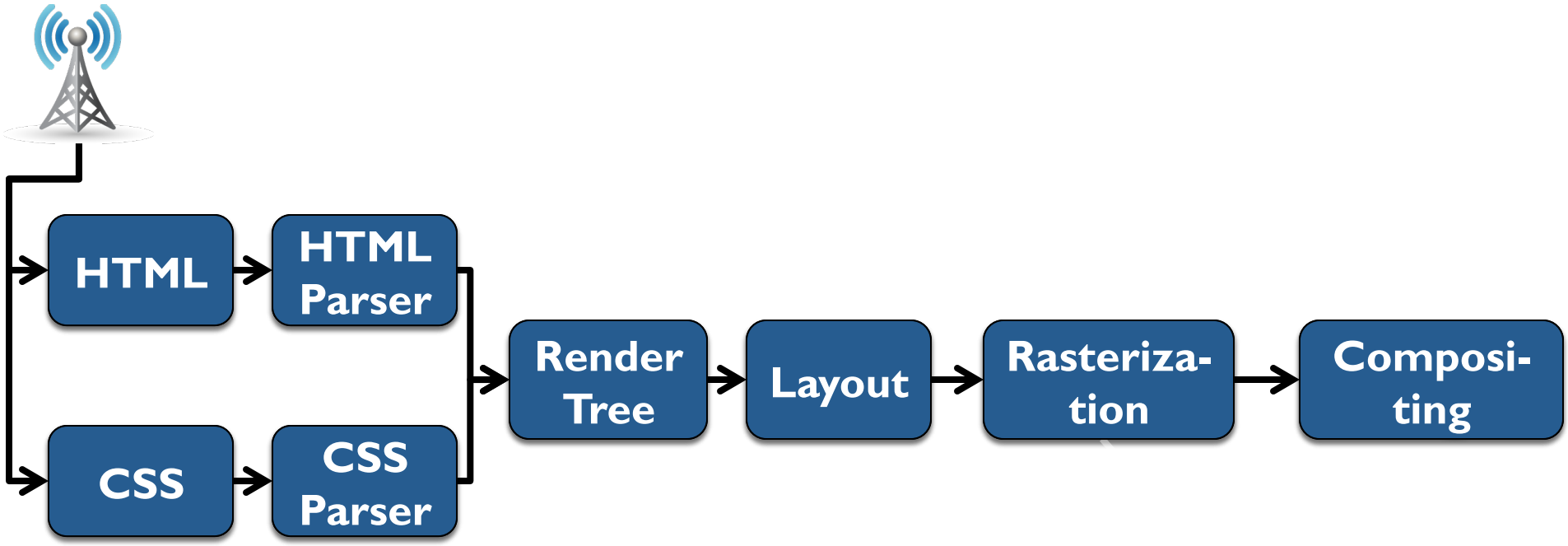
VP9

YouTube

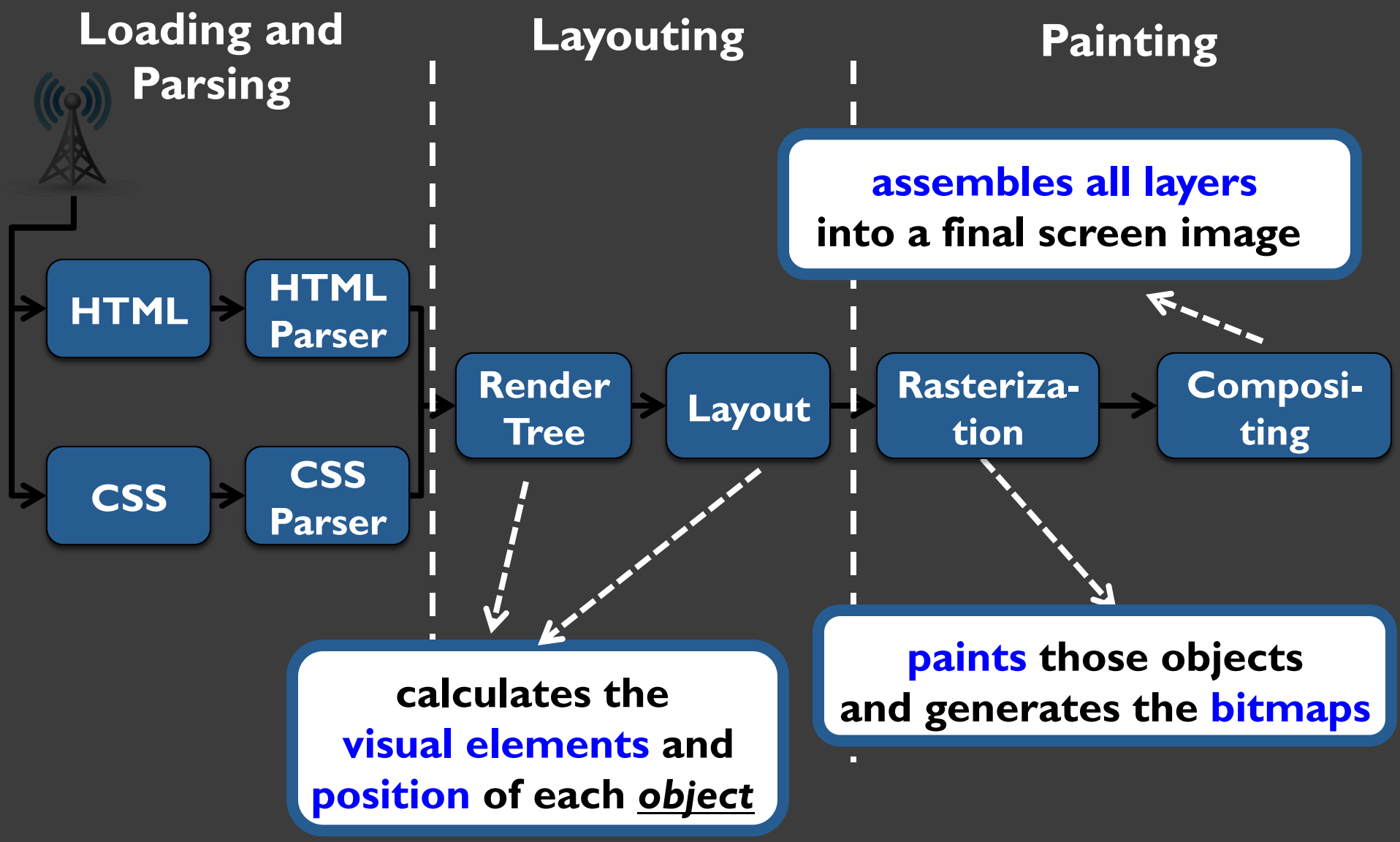
Video Capture

Google's video codec

How Chrome Renders a Web Page



How Chrome Renders a Web Page



Browser Analysis

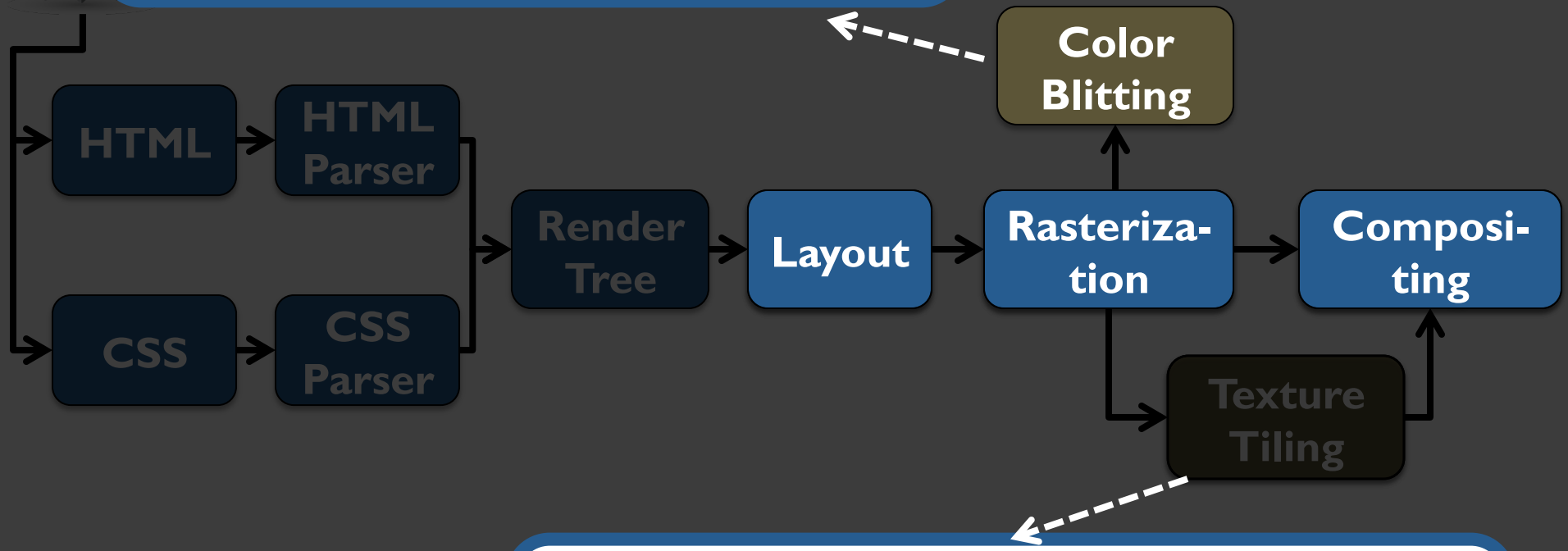
- **To satisfy user experience, the browser must provide:**
  - Fast **loading** of webpages
  - Smooth **scrolling** of webpages
  - Quick **switching** between browser tabs

- **We focus on two important user interactions:**
  1) **Page Scrolling**
  2) **Tab Switching**
  - Both include page loading

Scrolling

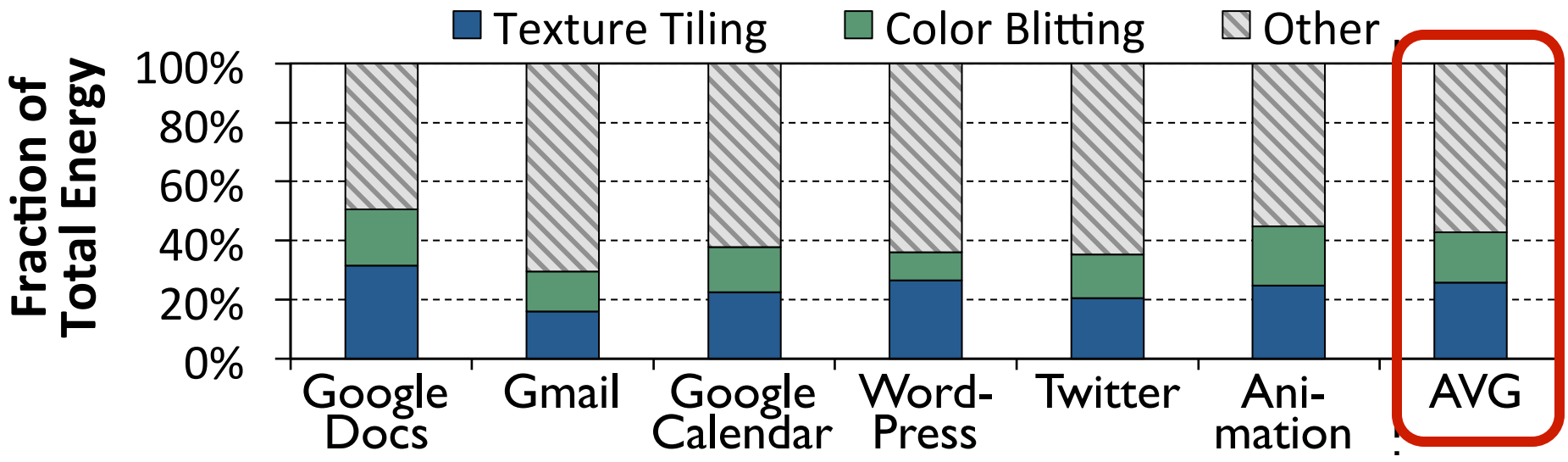
What Does Happen During Scrolling?

rasterization uses color blitters to convert the basic primitives into bitmaps



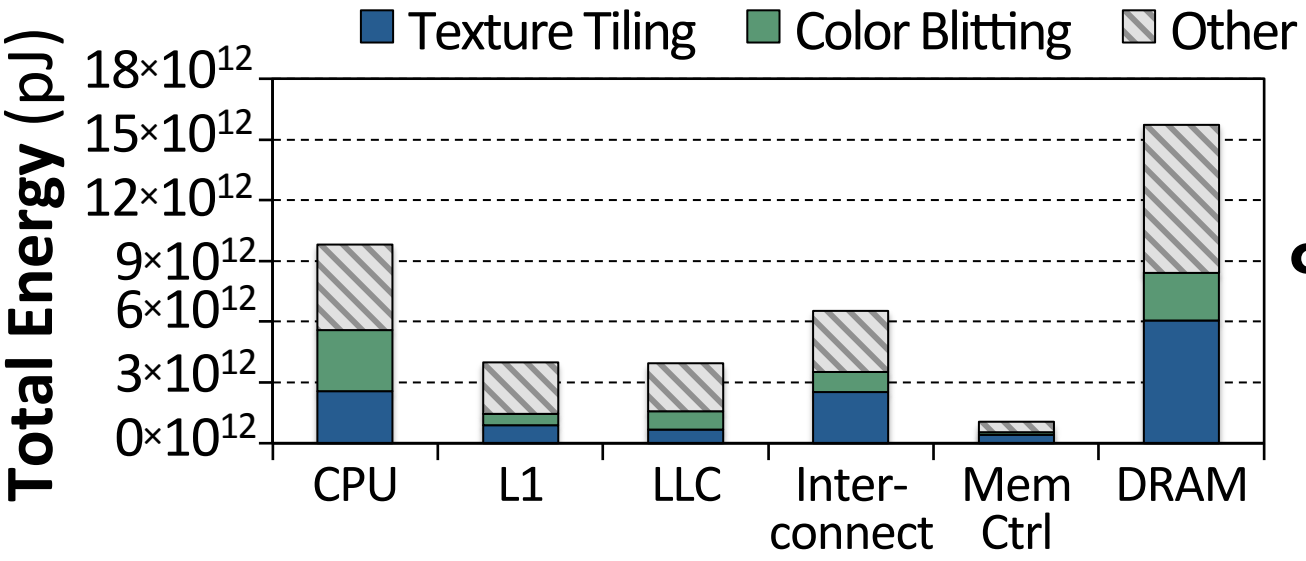
to minimize cache misses during compositing, the graphics driver reorganizes the bitmaps

Scrolling Energy Analysis



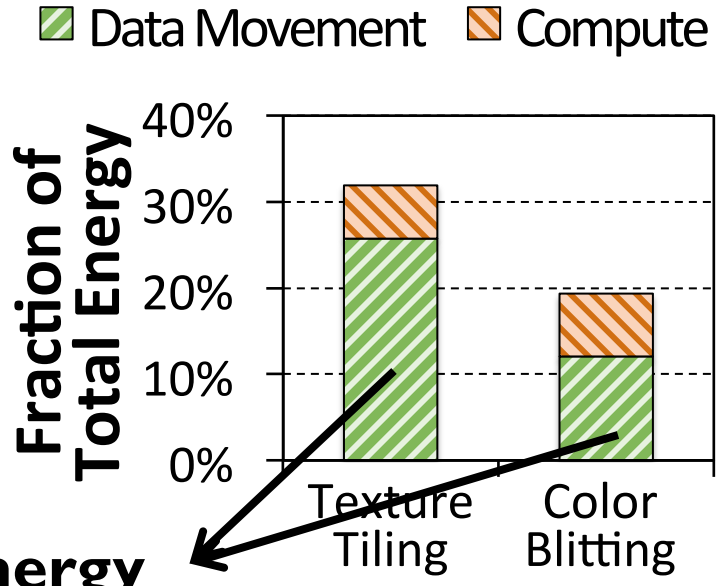
41.9% of page scrolling energy is spent on texture tiling and color blitting

Scrolling a Google Docs Web Page



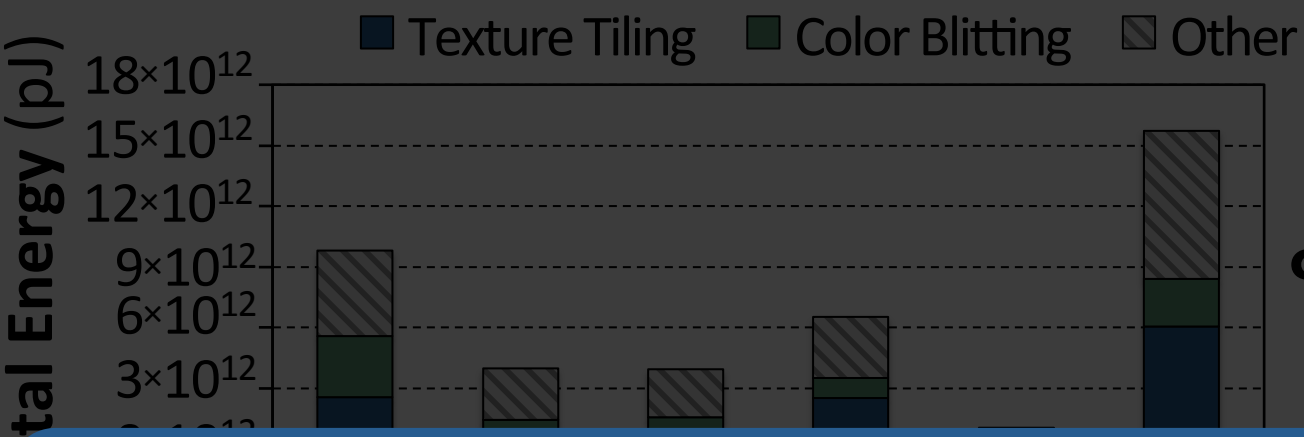
77% of total energy consumption goes to data movement

A significant portion of total data movement comes from texture tiling and color blitting



37.7% of total system energy

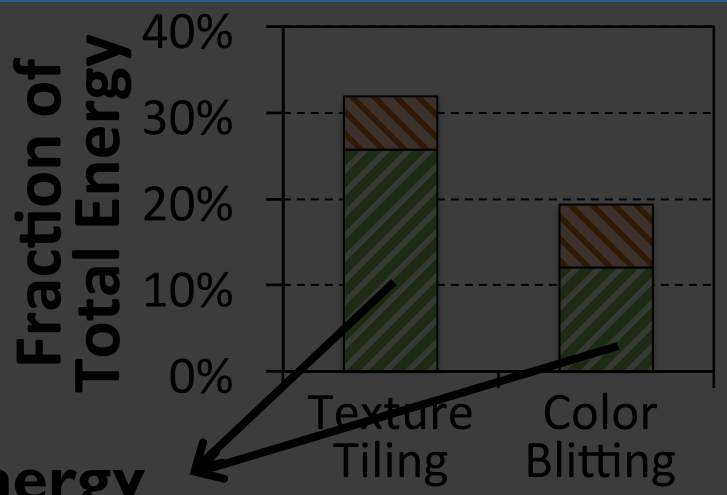
Scrolling a Google Docs Web Page



77% of total energy consumption goes to **data movement**

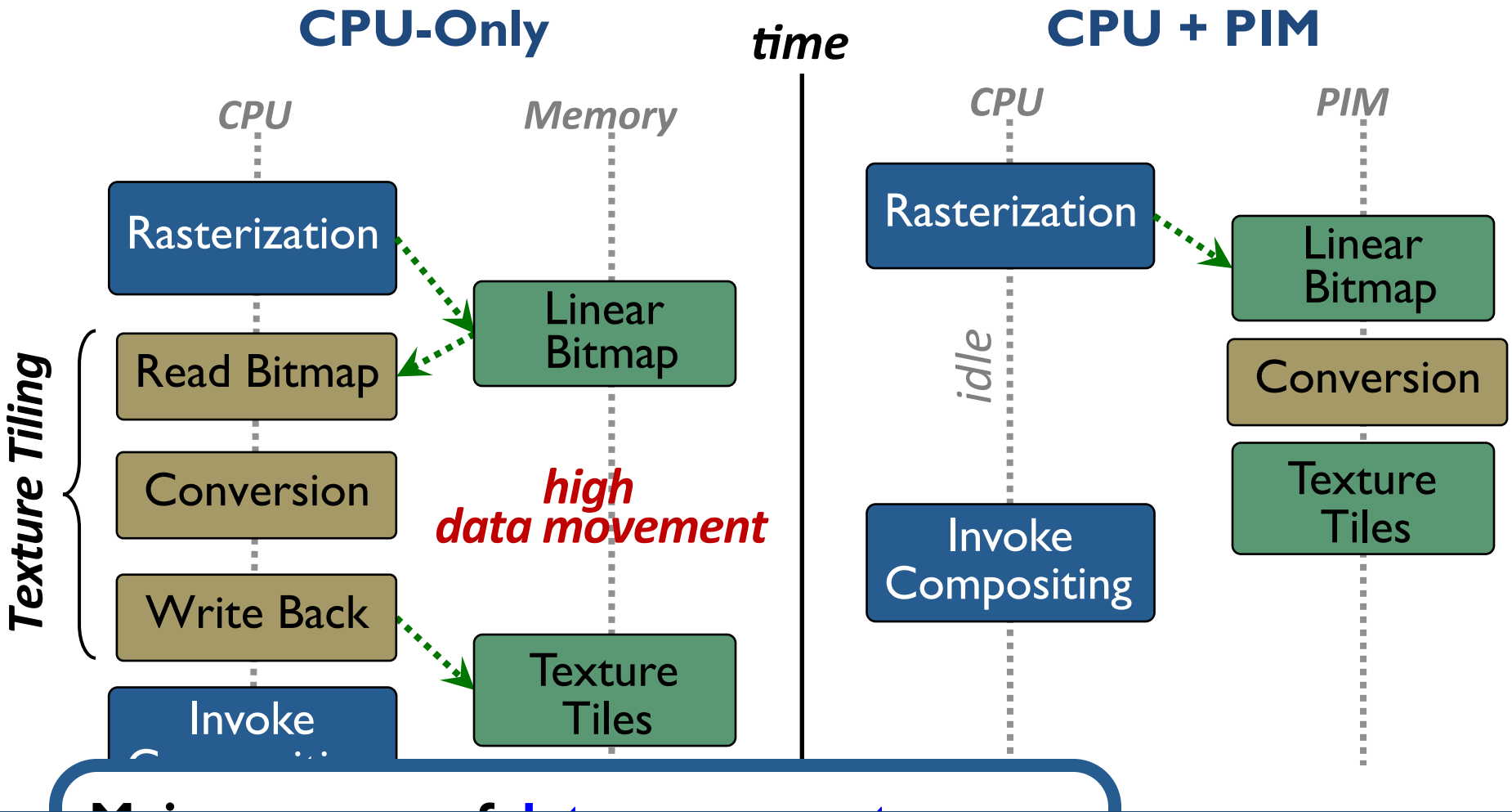
Can we use PIM to mitigate the data movement cost for texture tiling and color blitting?

A significant portion of total data movement comes from texture tiling and color blitting



37.7% of total system energy

Can We Use PIM for Texture Tiling?



Texture tiling is a good candidate for PIM execution

Can We Implement Texture Tiling in PIM Logic?

Requires simple primitives: memcopy, bitwise operations, and simple arithmetic operations

PIM Core

9.4% of the area available for PIM logic

PIM Accelerator

7.1% of the area available for PIM logic

PIM core and PIM accelerator are feasible to implement in-memory Texture Tiling

Color Blitting Analysis

Generates a large amount of data movement

Accounts for 19.1% of the total system energy during scrolling

Color blitting is a good candidate for PIM execution

Requires low-cost operations:

Memset, simple arithmetic, and shift operations

It is feasible to implement color blitting in PIM core and PIM accelerator

Scrolling Wrap Up

Texture tiling and color blitting account for a significant portion (41.9%) of energy consumption

↓

37.7% of total system energy goes to data movement generated by these functions

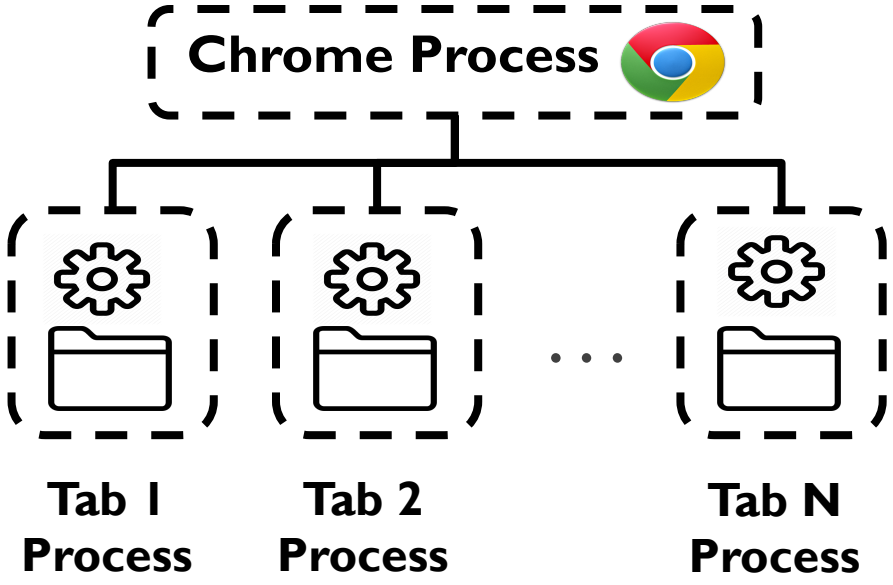
1 **Both functions can benefit significantly from PIM execution**

2 **Both functions are feasible to implement as PIM logic**

Tab Switching

What Happens During Tab Switching?

- **Chrome employs a multi-process architecture**
  - Each tab is a separate process

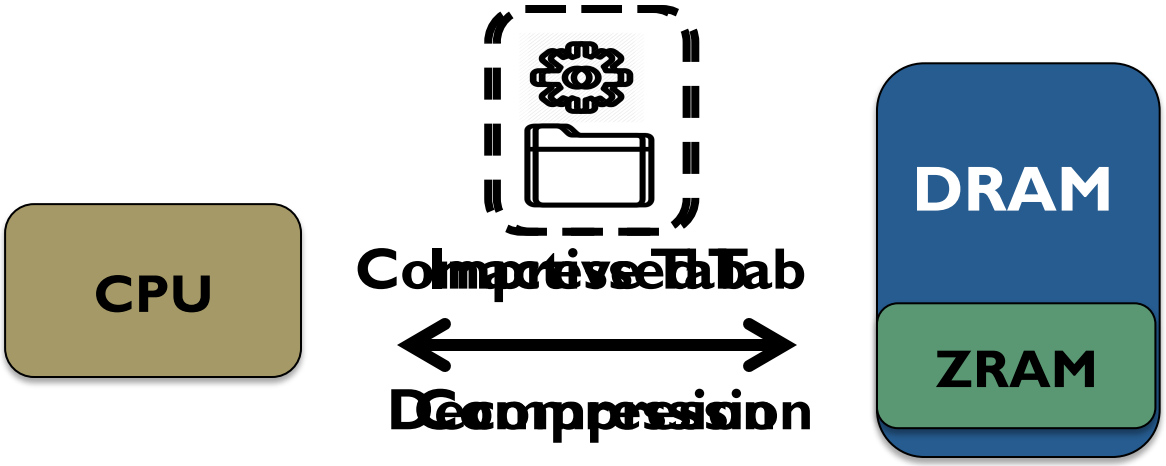


- **Main operations during tab switching:**
  - Context switch
  - Load the new page

Memory Consumption

- **Primary concerns during tab switching:**
  - How fast a new tab **loads** and **becomes interactive**
  - **Memory consumption**

Chrome uses compression to reduce each tab's memory footprint



Data Movement Study

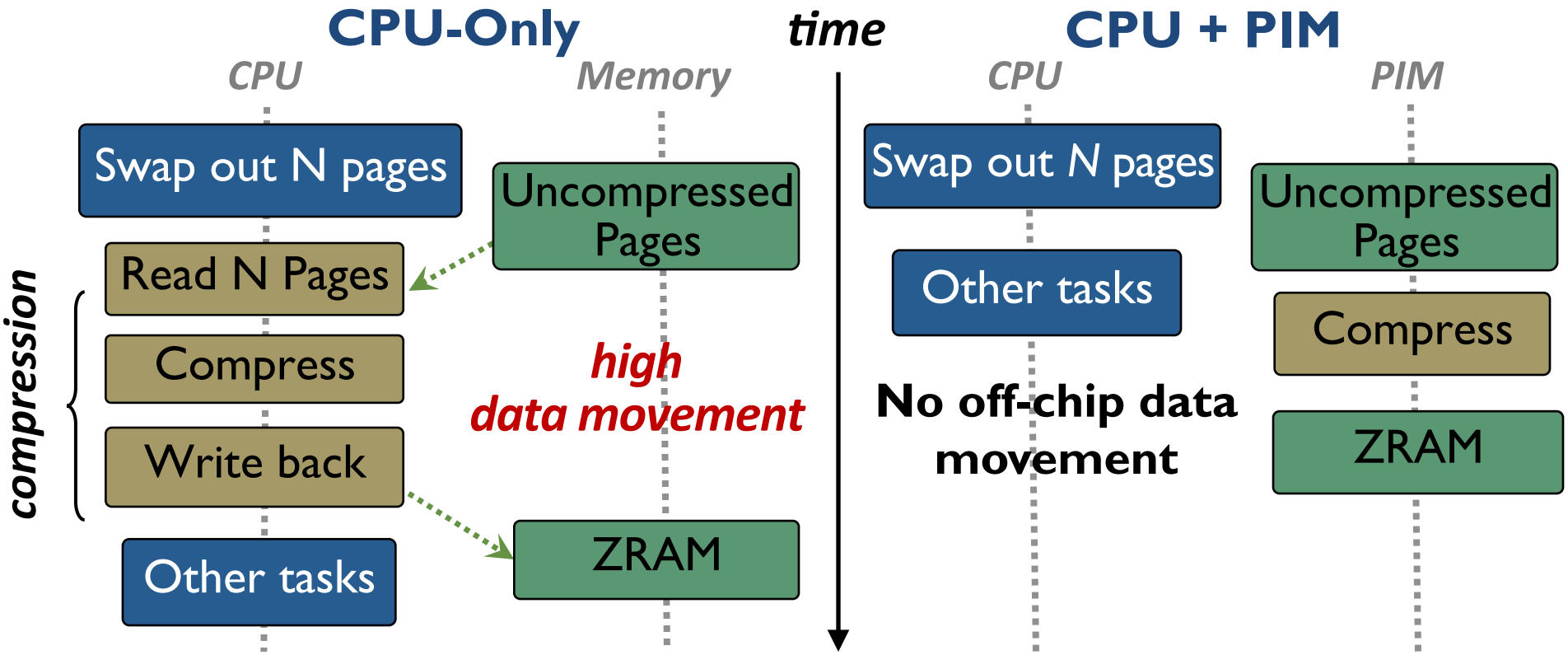
- To study **data movement** during tab switching, we emulate a user switching through 50 tabs

We make two key observations:

1 **Compression** and **decompression** contribute to **18.1%** of the total system energy

2 **19.6 GB** of data moves between **CPU** and **ZRAM**

Can We Use PIM to Mitigate the Cost?



PIM core and PIM accelerator are feasible to implement in-memory compression/decompression

Tab Switching Wrap Up

A large amount of data movement happens during tab switching as Chrome attempts to compress and decompress tabs

Both functions can benefit from PIM execution and can be implemented as PIM logic

Workload Analysis

Chrome

Google's web browser

TensorFlow

Google's machine learning framework

VP9

YouTube

Video Playback

Google's video codec

VP9

YouTube

Video Capture

Google's video codec

Workload Analysis

Chrome

Google's web browser

TensorFlow

Google's machine learning framework

VP9

Video Playback

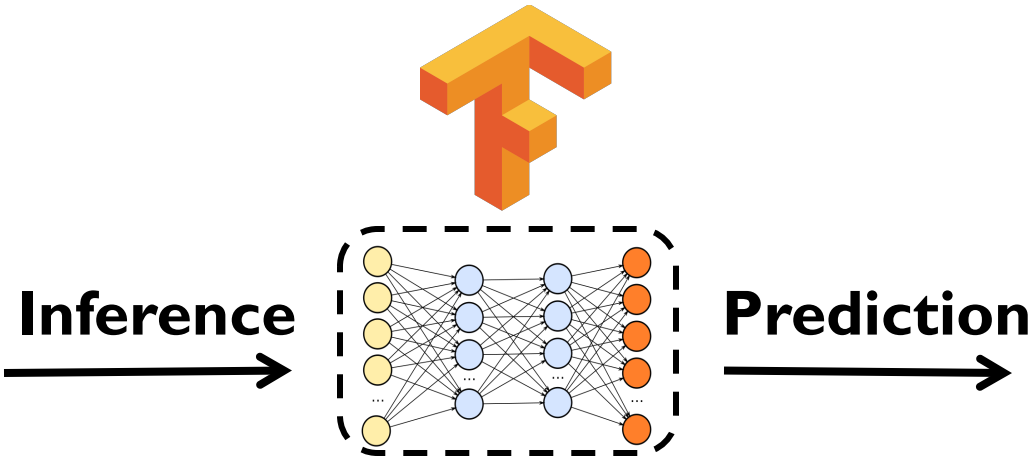
Google's video codec

VP9

Video Capture

Google's video codec

TensorFlow Mobile



57.3% of the inference energy is spent on **data movement**

↓

54.4% of the **data movement** energy comes from **packing/unpacking** and **quantization**

Packing

Reorders elements of matrices to minimize **cache misses** during **matrix multiplication**

Up to **40%** of the inference **energy** and **31%** of inference **execution time**

Packing's data movement accounts for up to **35.3%** of the inference **energy**

A simple data reorganization process that requires simple arithmetic

Quantization

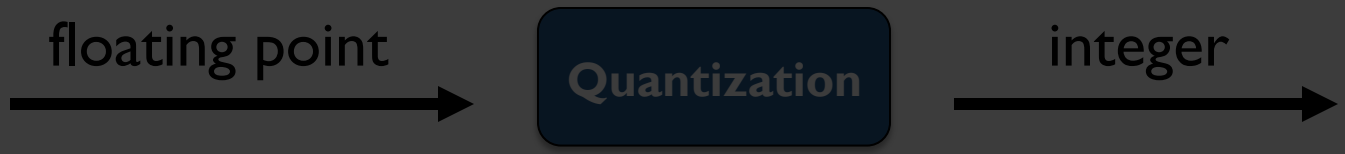
Converts 32-bit floating point to 8-bit integers to improve inference execution time and energy consumption

Up to 16.8% of the inference energy and 16.1% of inference execution time

Majority of quantization energy comes from data movement

A simple data conversion operation that requires shift, addition, and multiplication operations

Quantization



Converts 32-bit floating point to 8-bit integers to improve inference execution time and energy consumption

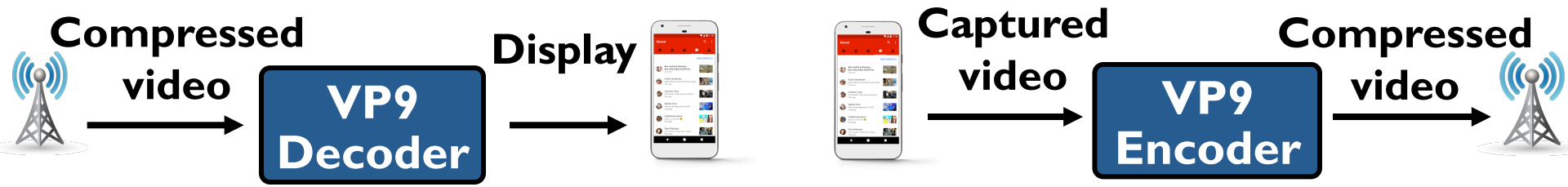
Based on our analysis, we conclude that:

- Both functions are good candidates for PIM execution
- It is feasible to implement them in PIM logic

A simple data conversion operation that requires shift, addition, and multiplication operations

Video Playback and Capture

VP9



Majority of energy is spent on data movement

Majority of data movement comes from simple functions in decoding and encoding pipelines

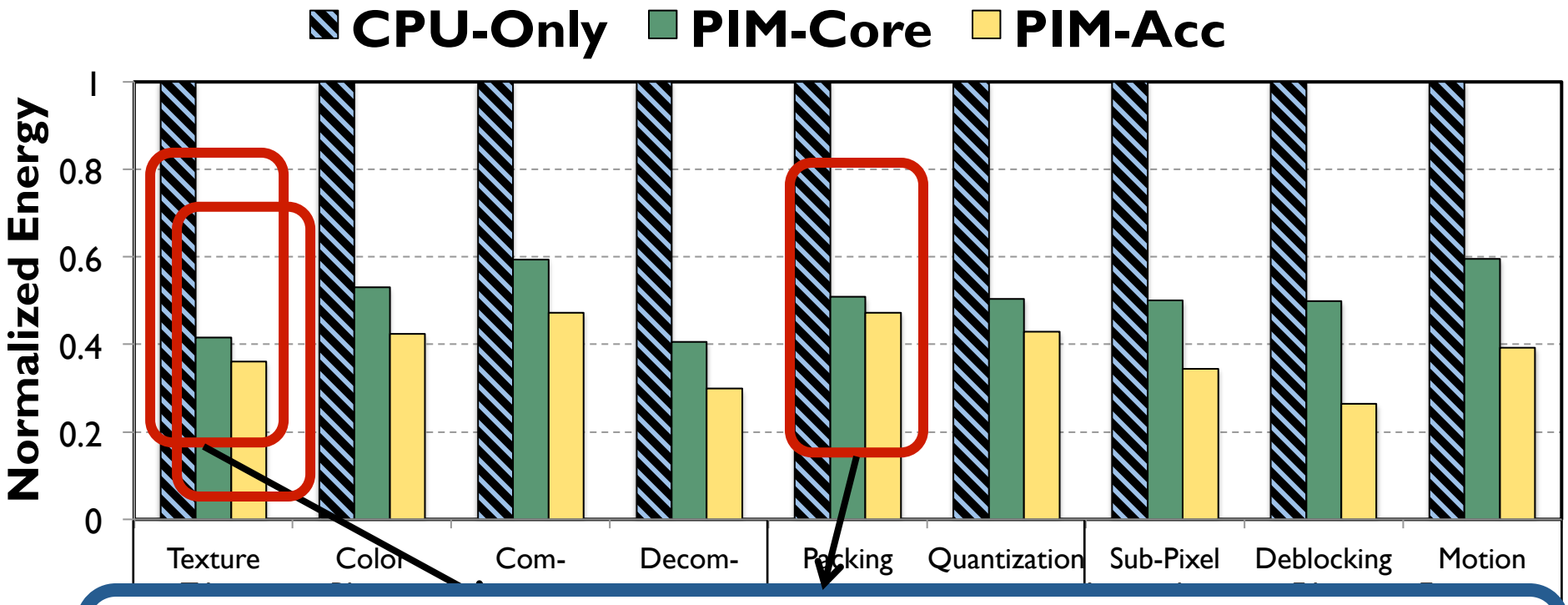
Outline

- Introduction
- Background
- Analysis Methodology
- Workload Analysis
- **Evaluation**
- Conclusion

Evaluation Methodology

- **System Configuration (gem5 Simulator)**
  - **SoC:** 4 OoO cores, 8-wide issue, 64 kB L1 cache, 2MB L2 cache
  - **PIM Core:** 1 core per vault, 1-wide issue, 4-wide SIMD, 32kB L1 cache
  - **3D-Stacked Memory:** 2GB cube, 16 vaults per cube
    - Internal Bandwidth: 256GB/S
    - Off-Chip Channel Bandwidth: 32 GB/s
  - **Baseline Memory:** LPDDR3, 2GB, FR-FCFS scheduler
- **We study each target in isolation and emulate each separately and run them in our simulator**

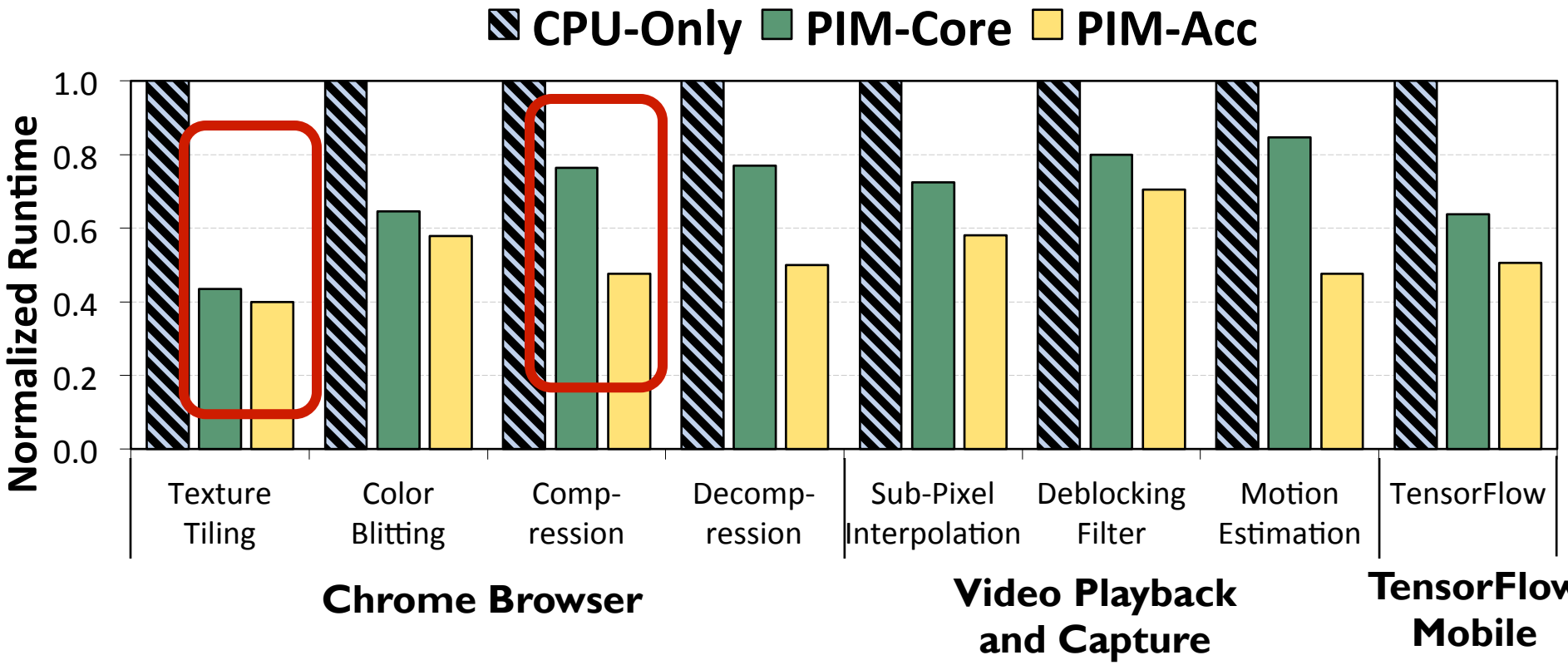
Normalized Energy



77.7% and **82.6%** of energy reduction for **texture tiling** and **packing** comes from eliminating **data movement**

PIM core and **PIM accelerator** reduces **energy consumption** on average by **49.1%** and **55.4%**

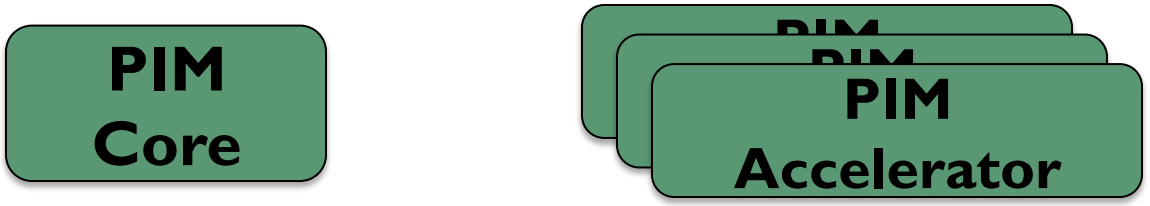
Normalized Runtime



Offloading these kernels to PIM core and PIM accelerator improves performance on average by 44.6% and 54.2%

Conclusion

- Energy consumption is a major challenge in consumer devices
- We conduct an in-depth analysis of popular Google consumer workloads
  - 62.7% of the total system energy is spent on data movement
  - Most of the data movement often come from simple functions that consists of simple operations
- We use PIM to reduce data movement cost
  - We design lightweight logic to implement simple operations in DRAM



  - Reduces total energy by 55.4% on average
  - Reduces execution time by 54.2% on average

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

SAFARI Carnegie Mellon Google

SAMSUNG SEOUL NATIONAL UNIVERSITY ETH Zürich

Backup

Video Playback

63.5% of the system energy is spent on **data movement**

↓

80.4% of the **data movement** energy comes from **sub-pixel interpolation** and **deblocking filter**

↓

Sub-pixel interpolation: interpolates the value of pixels at **non-integer** location

Deblocking filter: a simple **low-pass** filter that attempts to remove discontinuity in pixels

Video Playback

Based on our analysis, we conclude that:

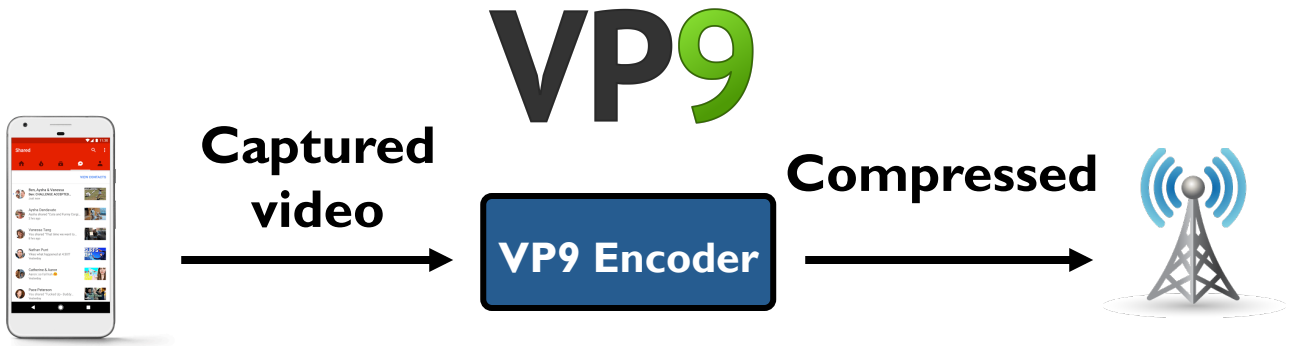
- Both Functions can benefit from PIM execution
- They can be implemented in PIM logic

sub-pixel interpolation and deblocking filter

Sub-pixel interpolation: interpolates the value of pixels at non-integer location

Deblocking filter: a simple low-pass filter that attempts to remove discontinuity in pixels

Video Capture



59.1% of the system energy is spent on data movement

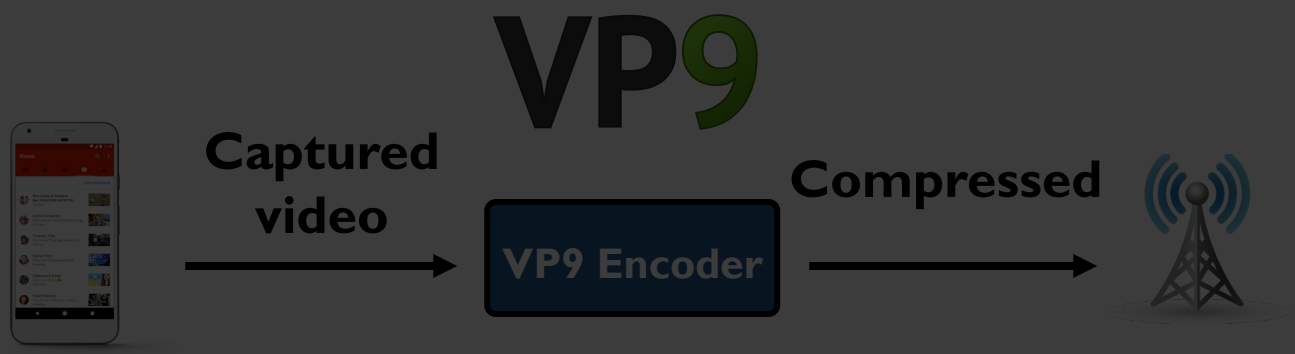
↓

Majority of the data movement energy comes from motion estimation which accounts for 21.3% of total system energy

↓

Motion estimation: compresses the frames using temporal redundancy between them

Video Capture



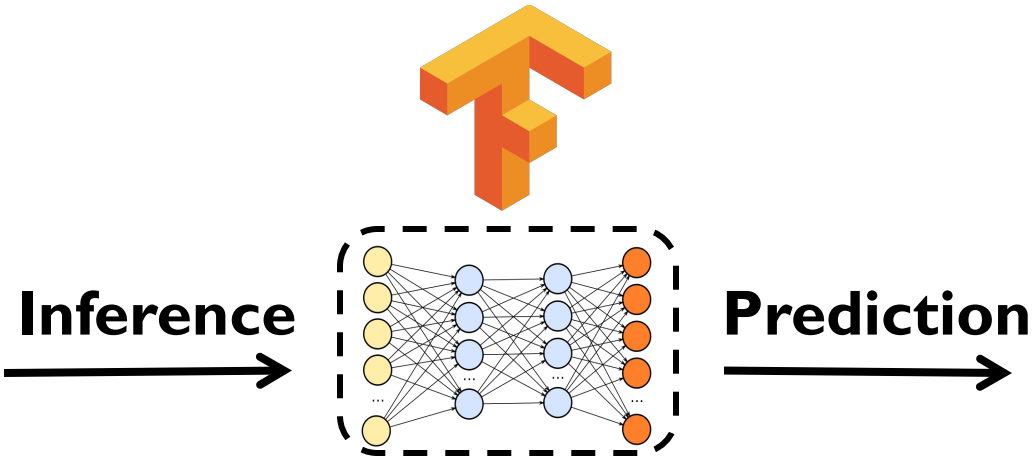
Motion Estimation is a good candidate for PIM execution and is feasible to be implemented at PIM logic

Majority of the data movement energy comes from motion estimation which accounts for 21.3% of total system energy

↓

Motion estimation: compresses the frames using temporal redundancy between them

TensorFlow Mobile



57.3% of the inference energy is spent on **data movement**

↓

54.4% of the **data movement** energy comes from **packing/unpacking** and **quantization**

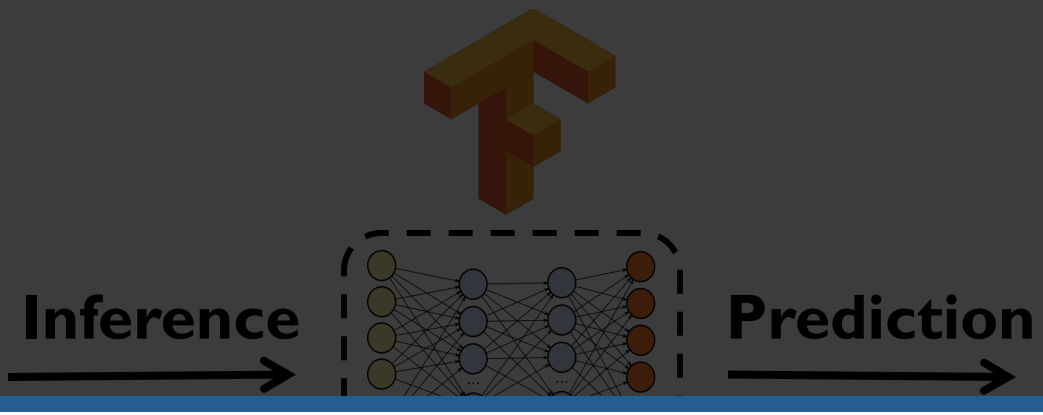
↓

Packing: A simple **data re-organization** process that **reorders** elements of matrices

↓

Quantization: A simple **data conversion** operation that converts **32-bit floating** point to **8-bit integers**

TensorFlow Mobile



Based on our analysis, we conclude that:

- Both Functions are good candidate for PIM execution
- It is feasible to implement them in PIM logic

54.4% of the data movement energy comes from packing/unpacking and quantization

Packing: A simple data re-organization process that reorders elements of matrices

Quantization: A simple data conversion operation that converts 32-bit floating point to 8-bit integers

Data Movement Study

- To study data movement during tab switching, we did an experiment:
  - A user opens 50 tabs (most-accessed websites)
  - Scrolls through each for a few second
  - Switches to the next tab

- We make two key observations:
  - Compression and decompression contribute to 18.1% of the total system energy
  - 19.6 GB of data swapped in and out of ZRAM

Other Details and Results in the paper

- **Detailed discussion of the other workloads**
  - Tensorflow Mobile
  - Video Playback
  - Video Capture

- **Detailed discussion of VP9 hardware decoder and encoder**

- **System Integration**
  - Software interface
  - Coherence

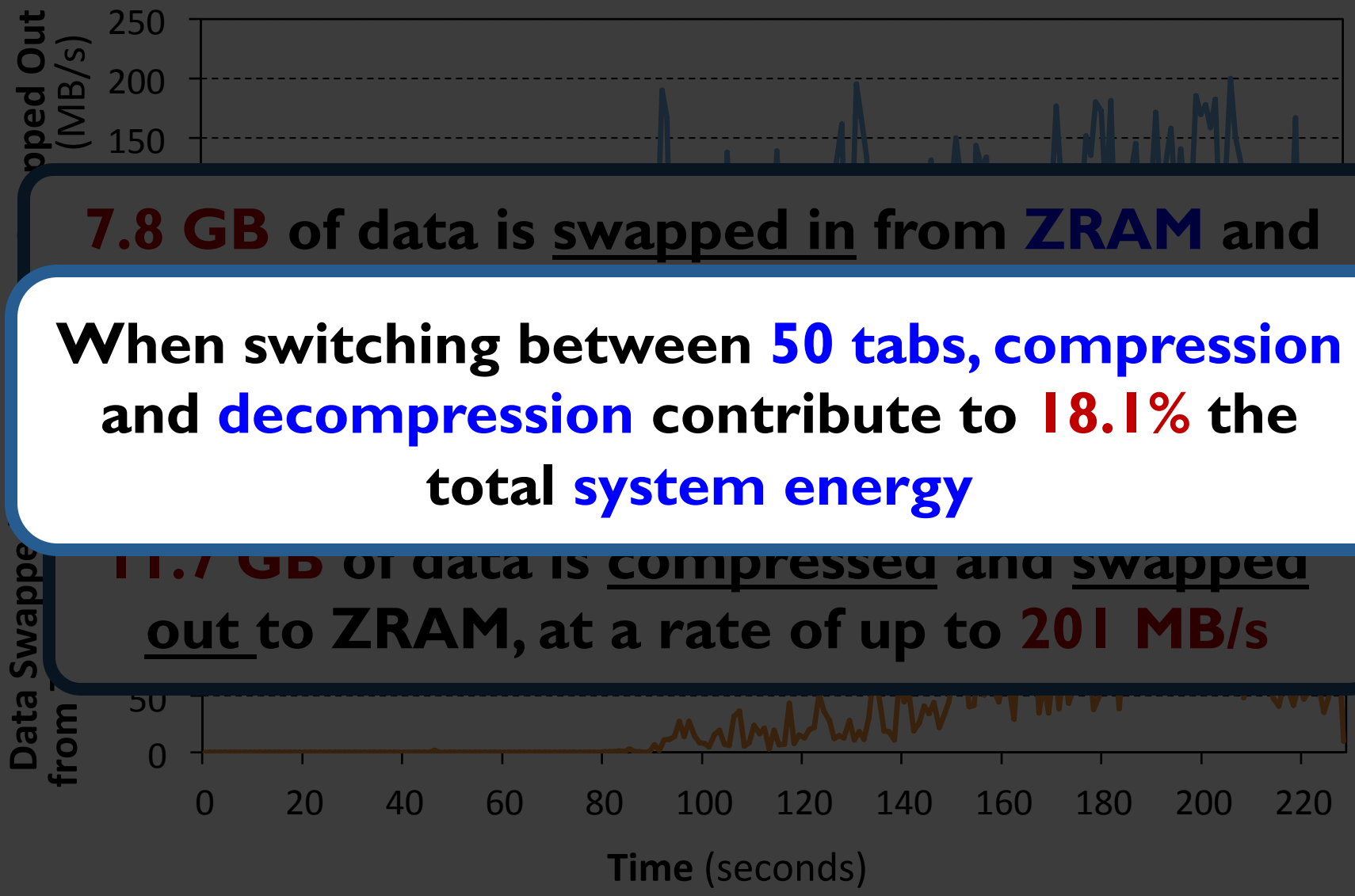
Color Blitting Analysis

- Color blitting generates a large amount of data movement
  - Accounts for 19.1% of the total system energy used during page scrolling
  - 63.9% of the energy consumed by color blitting is due to data movement

- **Based on our analysis, we conclude that:**
  - Color blitting is a good candidate for PIM execution
  - It is feasible to implement color blitting in PIM logic

- Color blitting require low-cost computation operations
  - Memset, simple arithmetic for alpha blending, shift operations

Data Movement During Tab-Switching

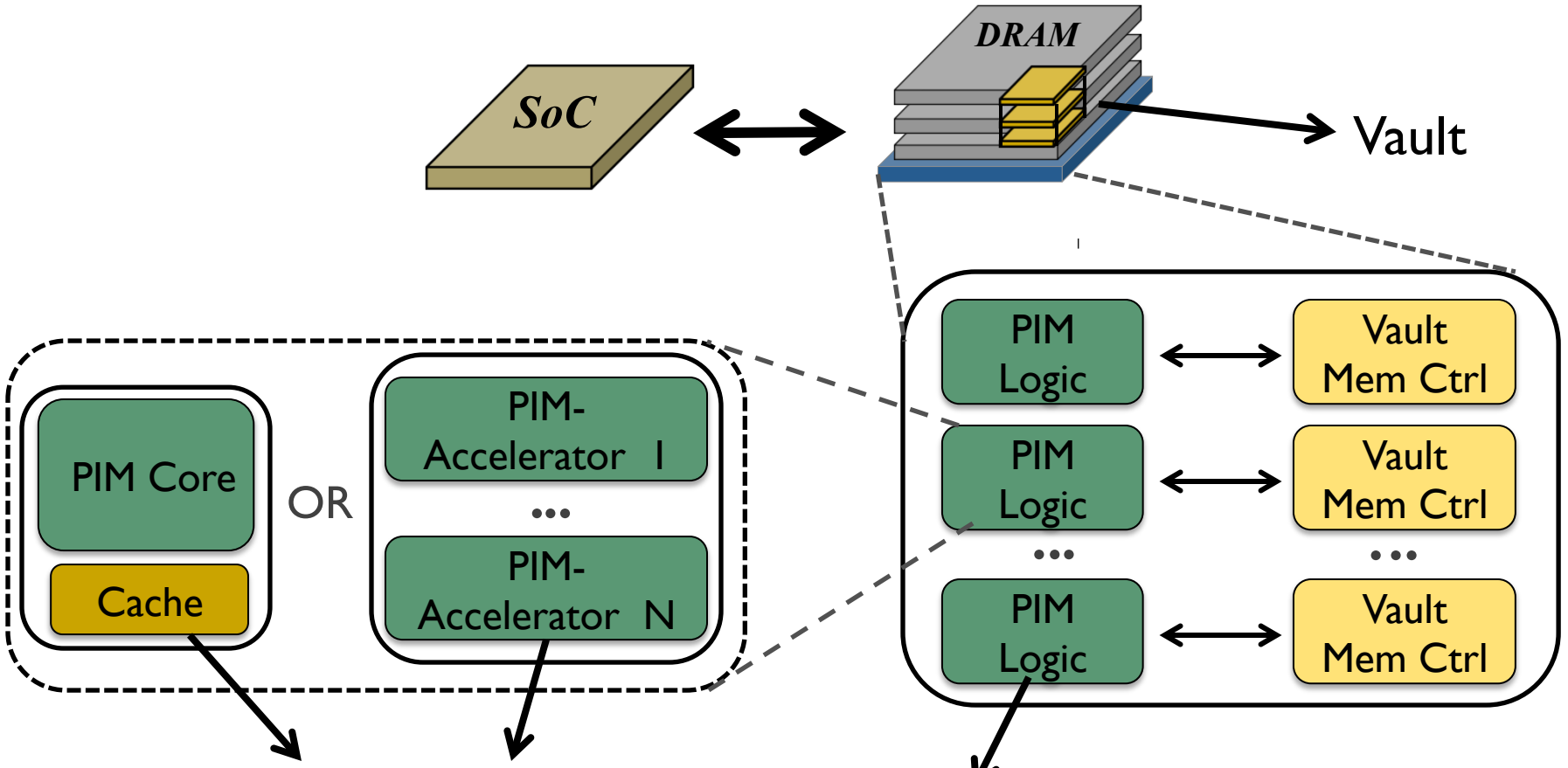


7.8 GB of data is swapped in from **ZRAM** and

When switching between 50 tabs, compression and decompression contribute to 18.1% the total system energy

11.7 GB of data is compressed and swapped out to ZRAM, at a rate of up to **201 MB/s**

Methodology: Implementing PIM Targets



General-purpose PIM core

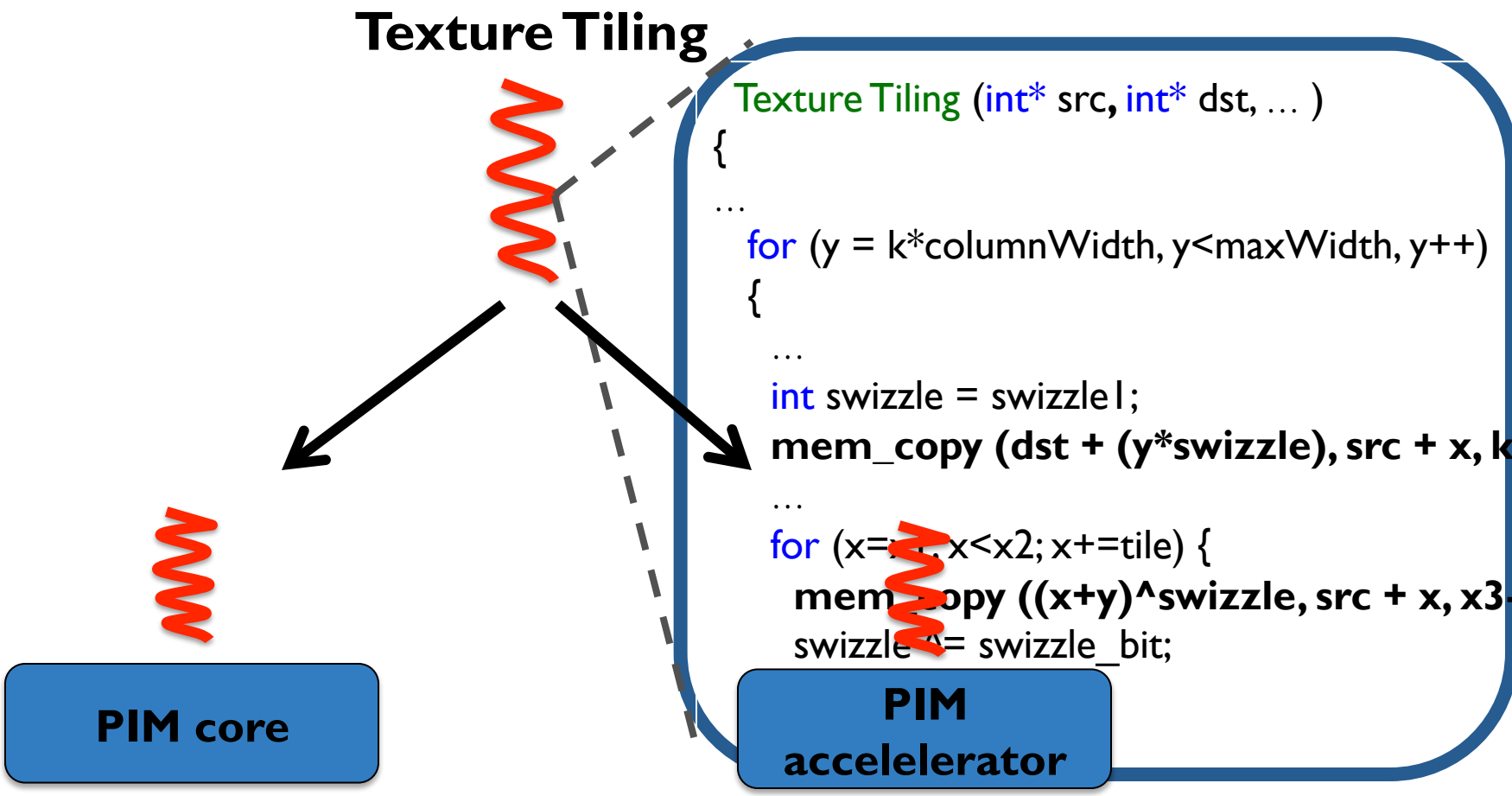
- A customized low-power single-issue core
- A 4-wide SIMD unit

Fixed-function PIM Accelerator

- A customized in-memory logic unit
- performs a single thread of the PIM target

3.5-4.4 mm² area budget for the PIM logic

Cost Analysis: Texture Tiling



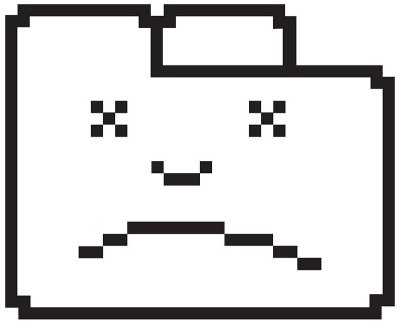
SIMD 4-wide
0.33 mm^2
(9.4% of the area available per vault)

Multiple In-memory tiling unit
0.25 mm^2
(7.1% of the area available per vault)

Requires simple primitives: memcopy, bitwise operations, and simple arithmetic operations

Memory Consumption

- **Potential Solution: kill inactive background tabs**
  - When the user accesses those tabs again, reload the tab pages from the disk
  - **Downsides:** 1) high latency 2) loss of data



- **Chrome uses compression to reduce each tab's memory footprint**
  - Uses compression to compress inactive pages and places them into a DRAM-based memory pool (ZRAM)
  - When the user switches to a previously-inactive tab, Chrome loads the data from ZRAM and decompresses it

Methodology: Identifying PIM targets

- **We pick a function as *PIM target* candidate if:**
  - Consumes **the most energy** out of the all functions
  - Significant **data movement** cost (**MPKI > 10**)
  - Bounded by **data movement**, not computation

- **We drop any candidate if:**
  - ✗ Incurs **any performance loss** when runs on **PIM logic**
  - ✗ Requires **more area than is available** in the logic layer of 3D-stacked memory