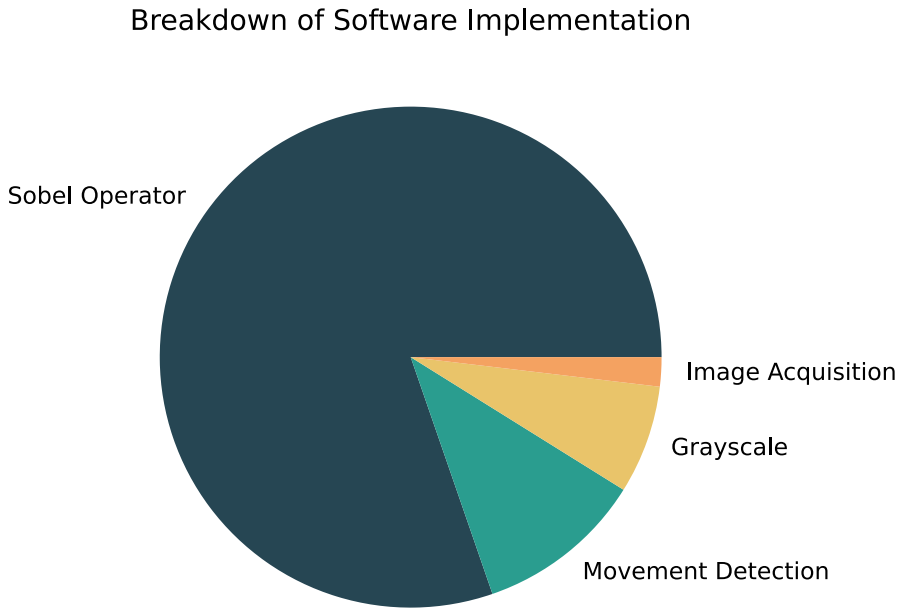
Movement Detection with Sobel Operator and Perceptual Hashing

Designed by

Giorgio Ajmone, Alessandro Cardinale

Profiling

- Pure software-based design
- Sequential computation with blocking operations
- Sobel Filter accounts for more than 80% of the total cycles



Proposal

Image acquisition

- Ping pong buffer to overlap with computation

Grayscale conversion

- Hardware implementation

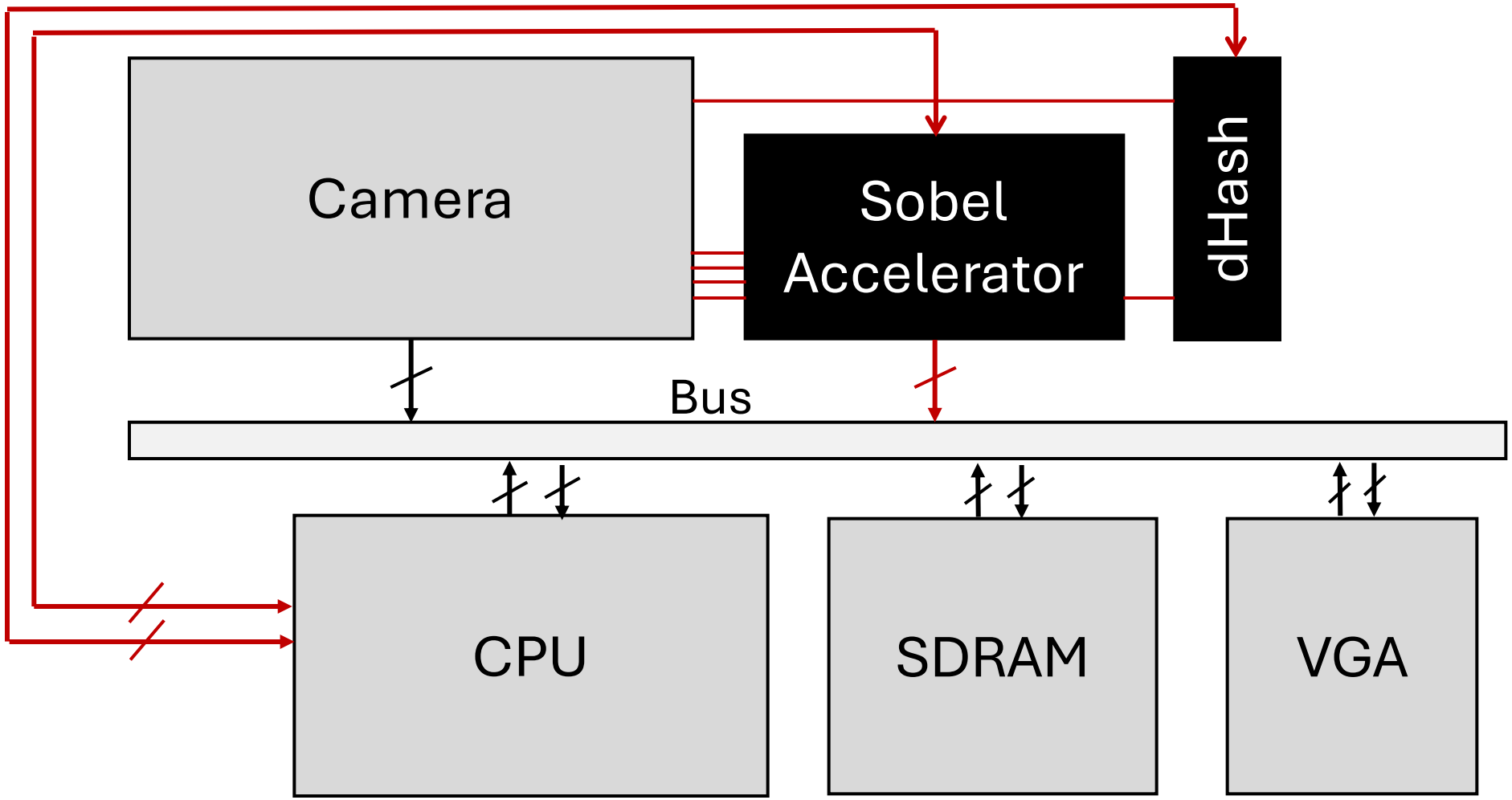
Sobel operator

- DMA streaming accelerator

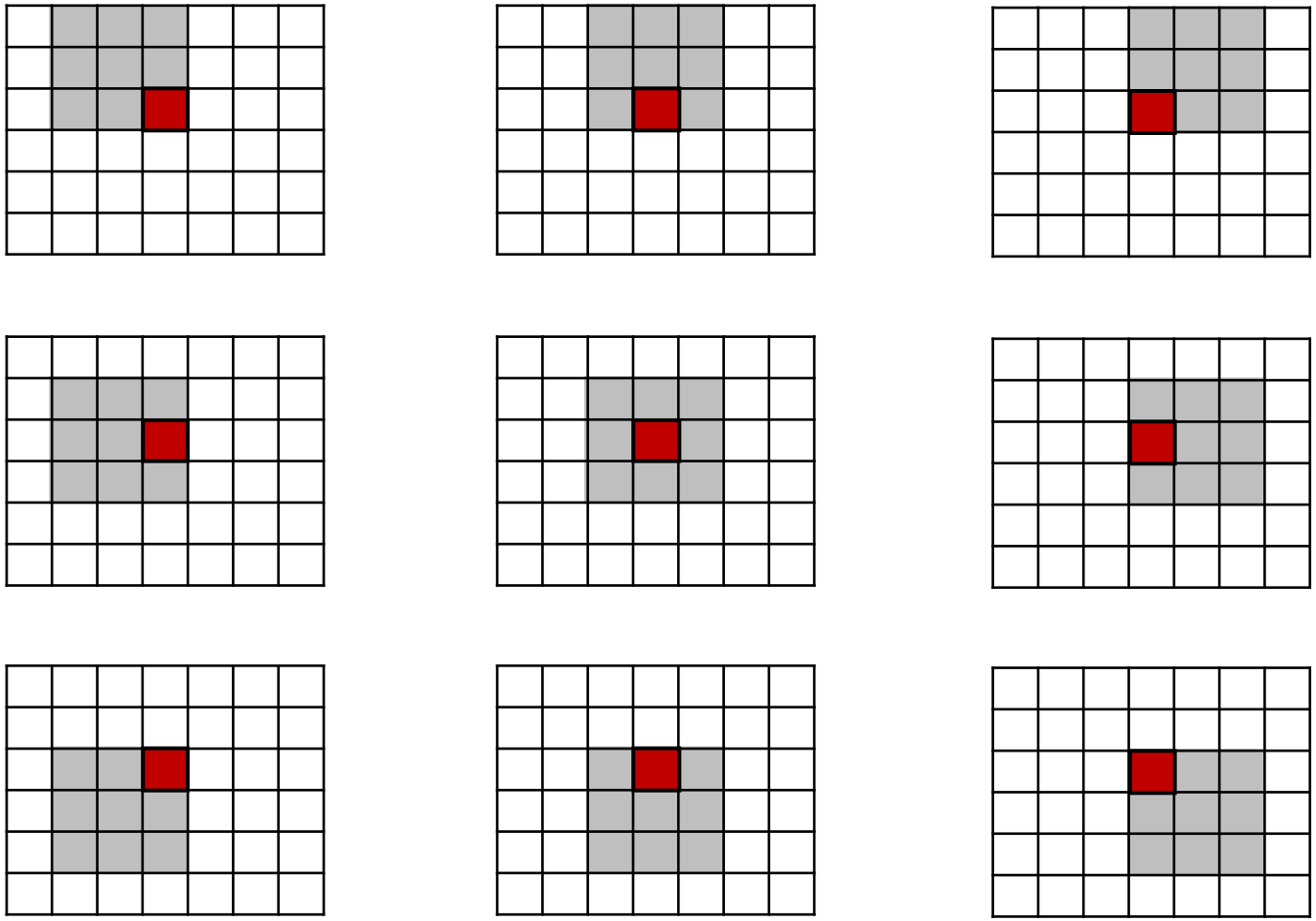
Movement Detection

- Dedicated hashing module and algorithmic optimization

Hardware: High Level



One Pixel - Nine Windows



Algorithmic Challenges

Sobel Operator

- Data reuse of a single pixel
- Computational parallelism across convolutional windows

Camera

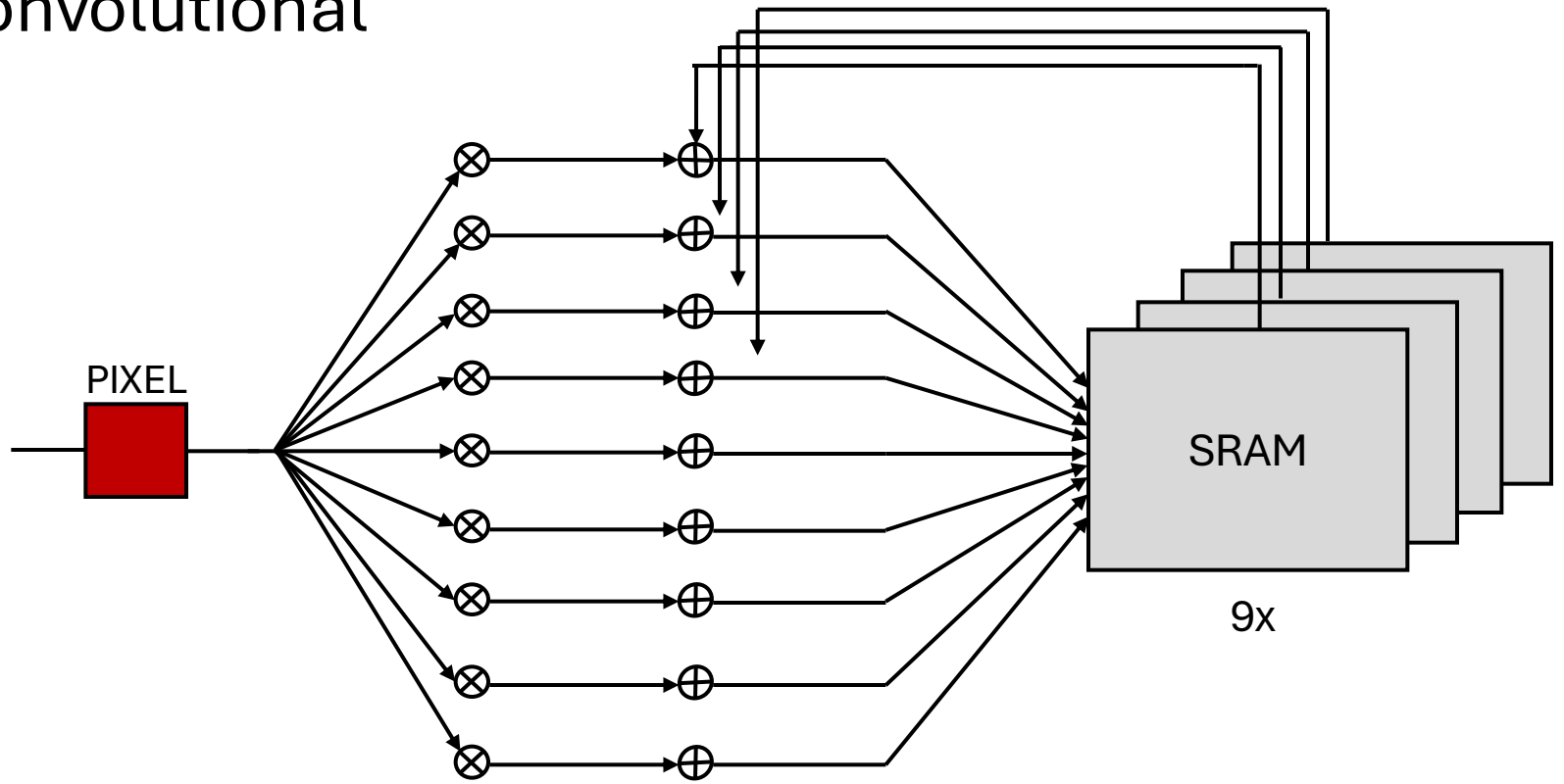
- Sequential stream
- Line-wise input

Approaches

- Collect pixels line by line and perform the whole convolution
- Implement streaming mechanism and store intermediate values

Computation flow

- A memory bank , a multiplier and an adder for each convolutional window



Algorithm Implementation

- For every pixel, the computation is the same
- We can parallelize the nine windows affected by the pixel
  - Interleaved memories to perform all accesses together
  - Rotating incremental addresses and overwrite logic
  - Logic to choose the right coefficient for each window

Movement Detection

Pixel-by-pixel comparison

- CPU compares pixels of two consecutive images and detect differences
- Ping-Pong Buffer to overlap comparison and transfer of the new image
- Bit-wise operations on compressed image (1 bit per pixel)

Signature-based detection

- Hash function to create a signature of the image without noise
- Hamming distance between two signatures as a metric of similarity
- If the hamming distance is higher than threshold , there is movement

Locality-Sensitive Hashing Functions

Hashing functions

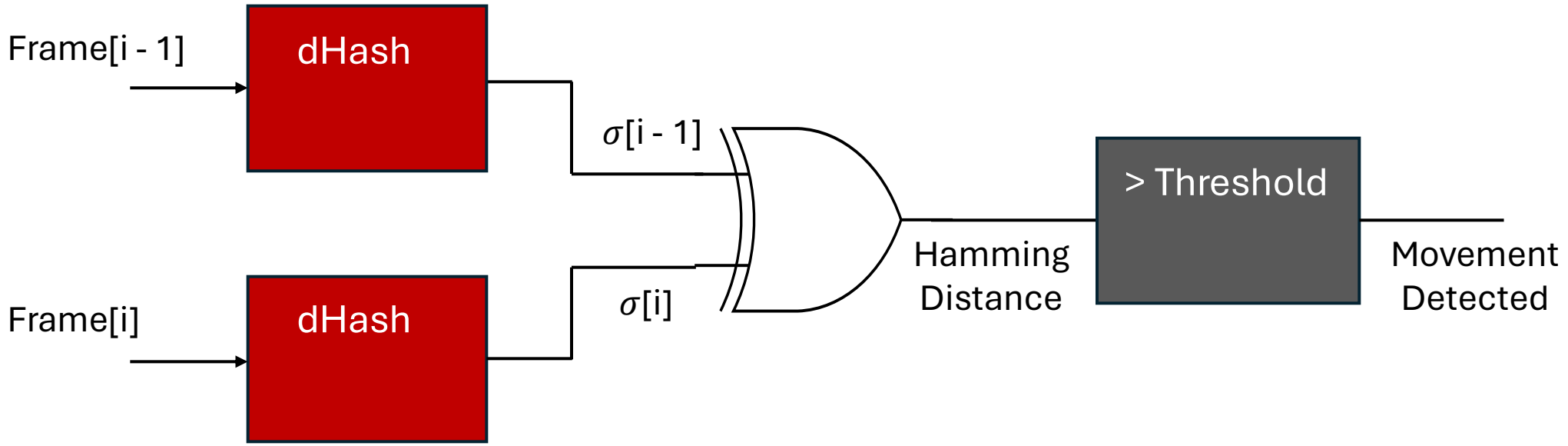
- Maximize the entropy of the image
- Should never create the same signature for two different images
- A small difference results in a completely different signature

Locality-Sensitive Hashing functions (LSH)

- Minimize the entropy of the image
- Two similar images will have similar signatures
- Several algorithms, state-of-the-art is dHashing (difference hashing)

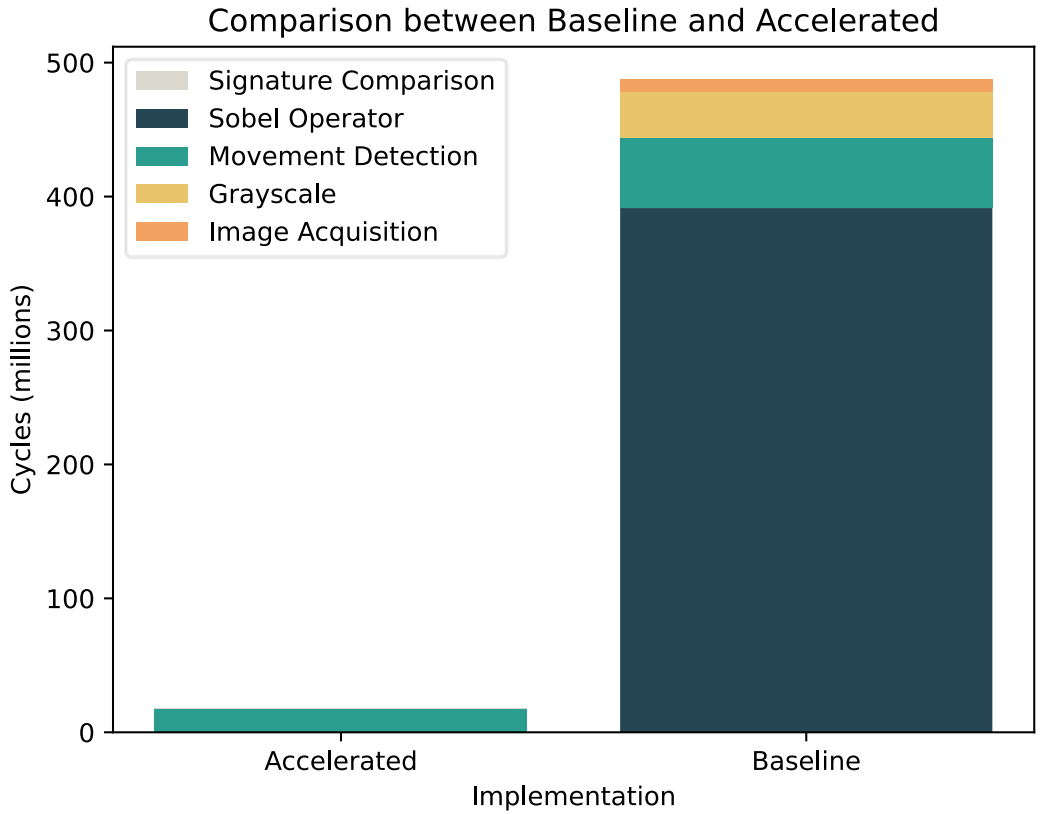
LSH Implementation

- Dedicated module creates the signature from the pixel streaming coming from the camera
- CPU reads the signature with custom instruction and measure the Hamming Distance in software



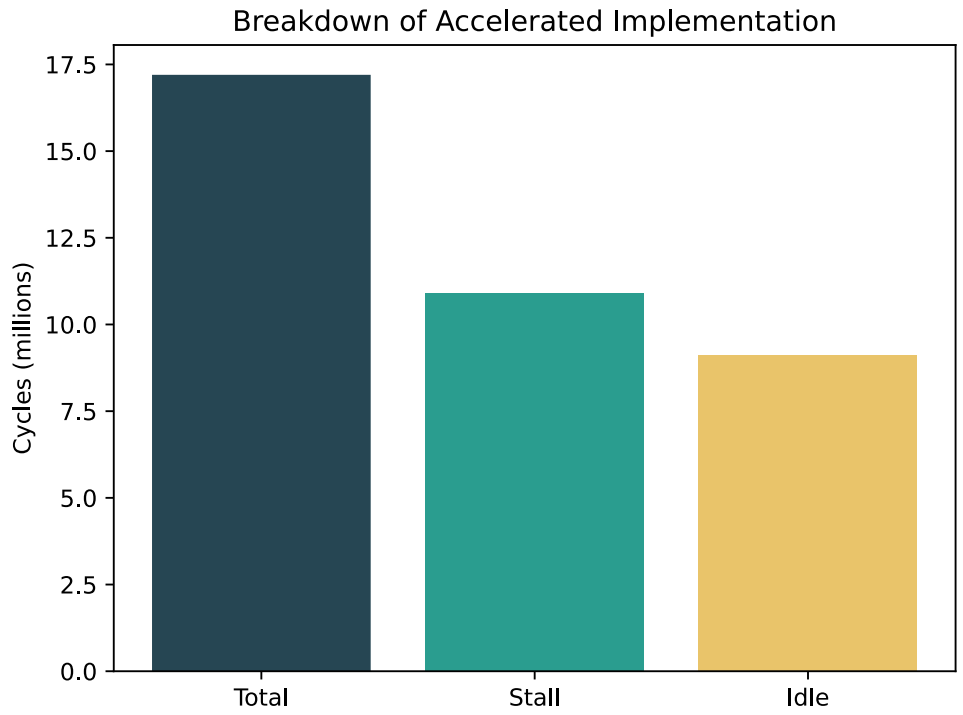
Results

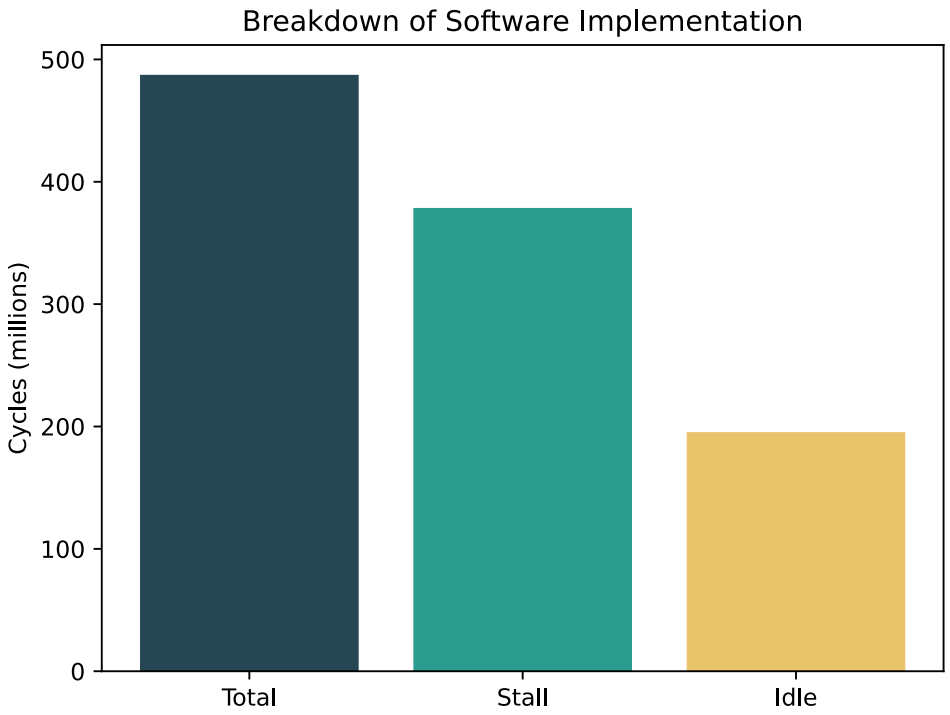
- 28x improvement
- Movement detection is now the bottleneck
- Further improvement requires dedicated memory



Thank you for the attention

Profiling


Breakdown of Accelerated Implementation
Cycles (millions)
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Total
Stall
Idle


Breakdown of Software Implementation
Cycles (millions)
500
400
300
200
100
0
Total
Stall
Idle