A Mutual Information Accelerator for Autonomous Robot Exploration

Peter Zhi Xuan Li, Sertac Karaman, Vivienne Sze

Massachusetts Institute of Technology

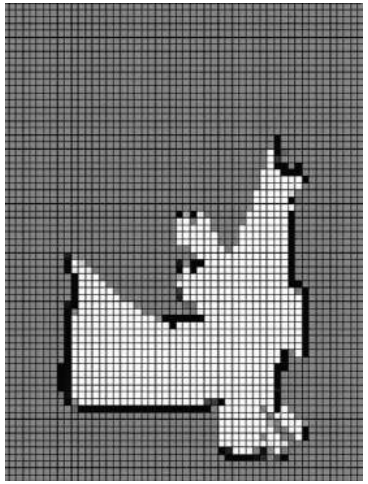


Website: https://lean.mit.edu/highlights/mutual-information

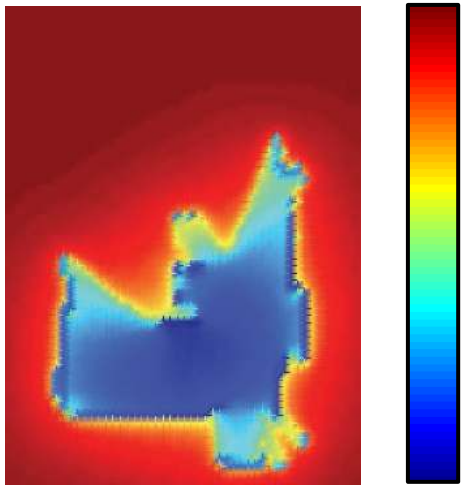
Abstract

Robotic Exploration

Where should the robot move next to learn the most **new information** about its environment?



Occupancy Map



Mutual Information (MI)

$$H(M|Z) = H(M) - I(M;Z)$$

Theoretically Proven Approach

Move to the location that **maximizes the mutual information** between prospective range measurements and the map for **faster mapping of the unknown environment.**

Challenge

Computing the mutual information $I(M;Z)$ is **slow** (high time complexity) and thus becomes the **bottleneck** of autonomous exploration system.

Our Contribution

Proposed Multicore MI Accelerator

✓ **High-throughput:** Computes the MI for the **entire map** of size 10.05mx10.05m with 0.05m resolution for **the first time at 11Hz on an ASIC** (**88x faster** than a typical ARM CPU used on robotic racecar) while consuming only **164mW**.

Benefits for Autonomous Exploration

✓ Enables more optimal selection of exploration path, which **reduces total exploration time and trajectory length**.

Motivation

- Applications for robotics exploration:

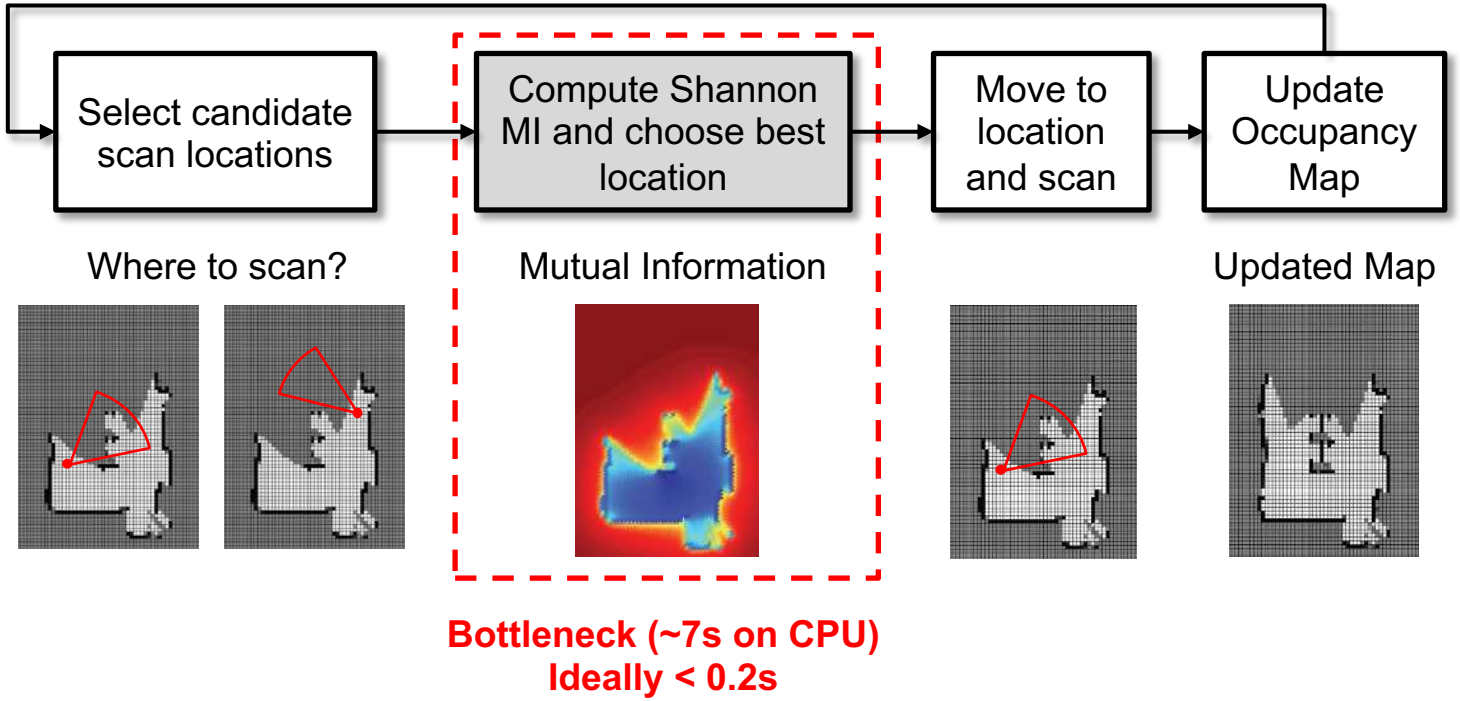
Search and rescue

Space exploration

- Autonomous exploration requires the robot to produce a complete map of the environment in the **shortest amount of time.**

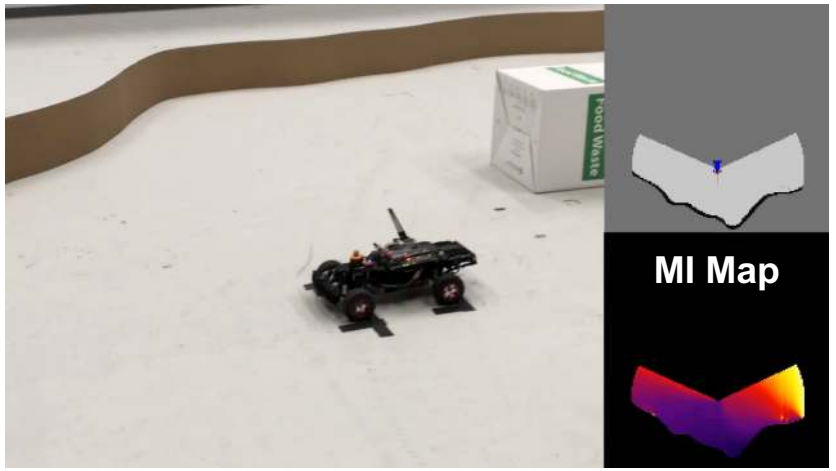
Autonomous Exploration

- **Goal:** explore and map the environment in the shortest amount of time



Autonomous exploration with a mini racecar (4x speed)

Occupancy map with planned path

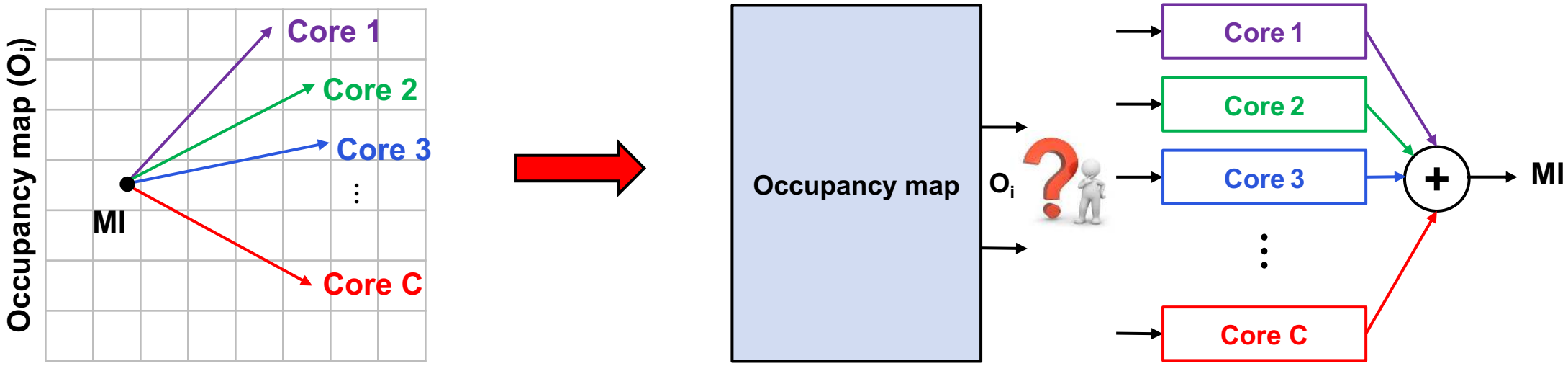


Link to video: https://youtu.be/6la0conjKMQ

Fast MI compute for the entire map = optimal selection for the next best scan location

Hardware Design Challenge: Data Delivery to MI Cores

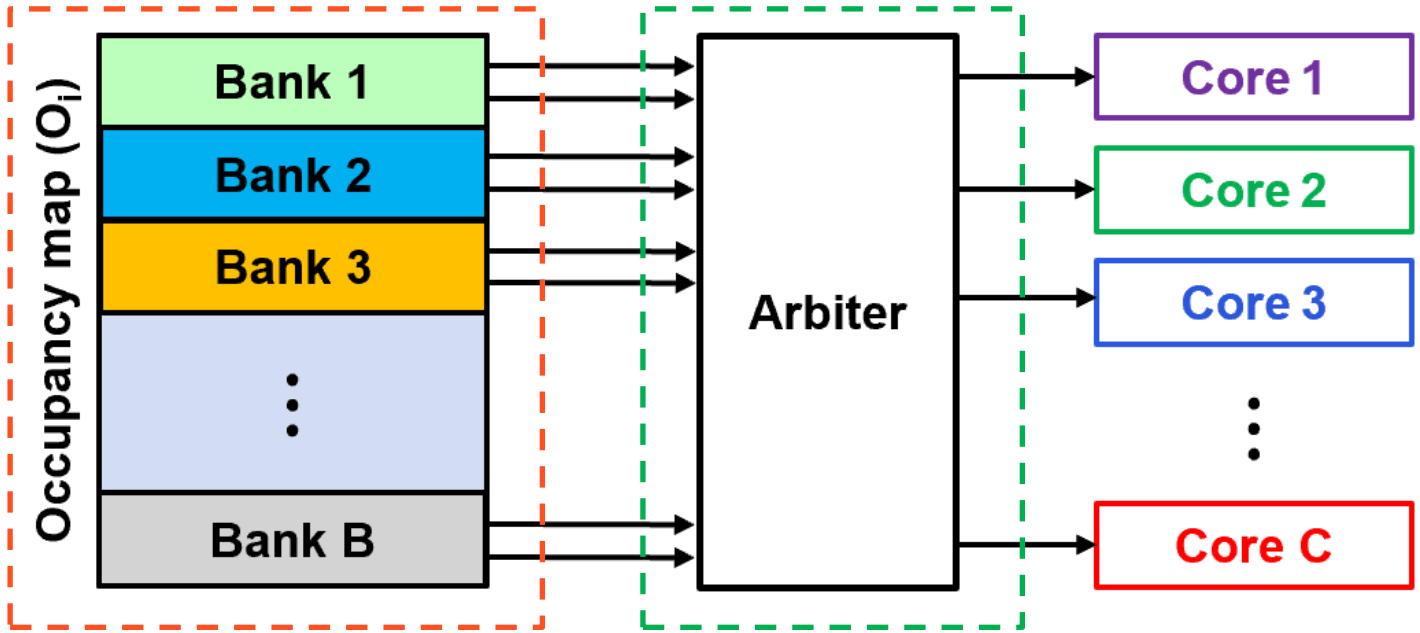
- **Parallelism:** each core computes MI for each single sensor beams by independently accessing the map every cycle



Compute is **limited by the bandwidth of the dual-port SRAMs** that stores the map.

Proposed Accelerator Architecture

- Increasing memory bandwidth (read ports) by partitioning the map storage into multiple banks.



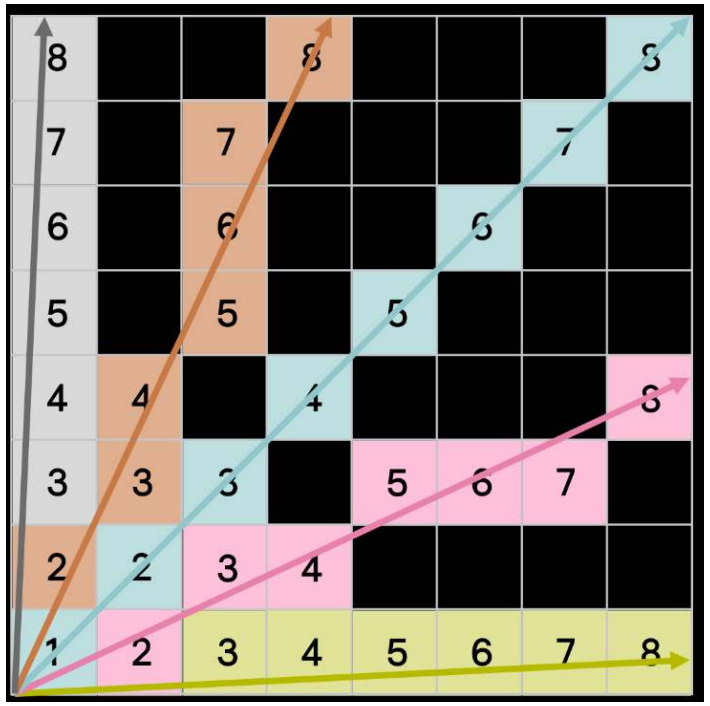
Proposed architecture includes:

1) **Memory banking pattern** that minimizes memory read conflicts among all cores.
2) **Efficient arbiter** that fairly and quickly manages the memory access requests of the cores.

Naïve Banking Implementation

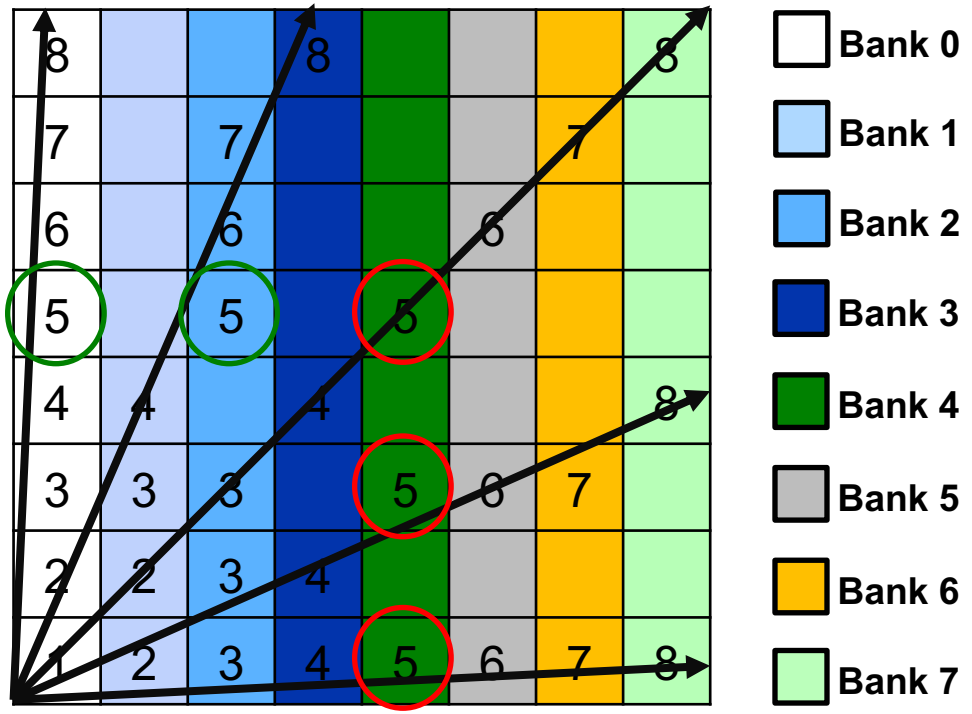
- **Challenge:** memory access pattern is scan location and sensor angle dependent.

Memory access pattern at every cycle



Cores read the map at the **same row or column every cycle**.

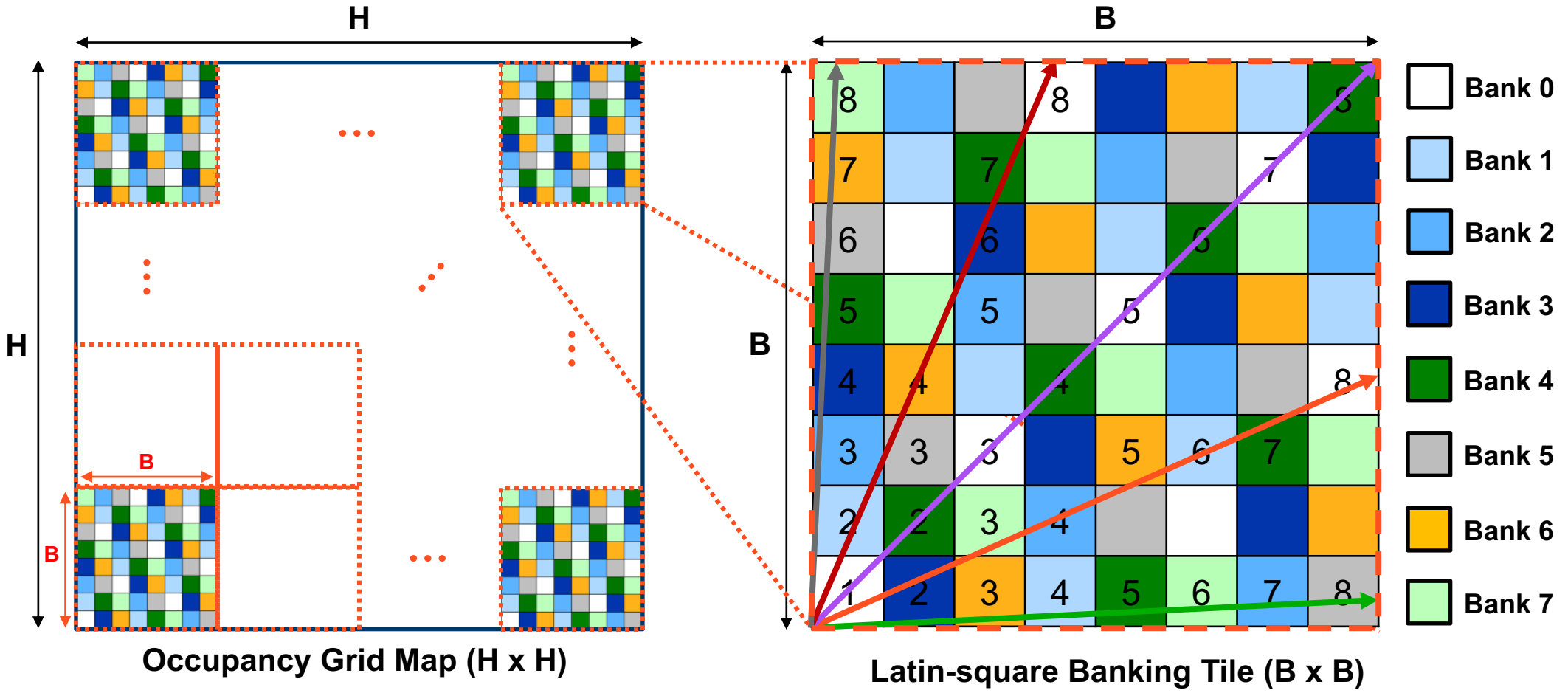
Naïve implementation



Naïve vertical banking pattern **always creates read conflicts for selected beams**

Proposed Memory Banking Pattern

- **Latin-square banking tile:** cells in each column and row is assigned to different banks



We **rigorously proved** that Latin-square tiles usage minimizes read conflict among cores.

Operation of the Priority Arbiter

- **Greedy & fair:** Services up to 2 cores with the highest idle time / priority per bank

Core Index	Bank	Address	Priority
1	5	10	6
2	6	14	5
3	5	13	6
4	5	7	4
5	6	15	7
6	3	20	5
7	2	11	6
8	5	2	4

Stage 1:
Extract up to 2 requests with the highest priorities for each bank

Bank	Address	Priority
5	10	6
6	14	5
5	13	6
5	7	4
6	15	7
3	20	5
2	11	6
5	2	4

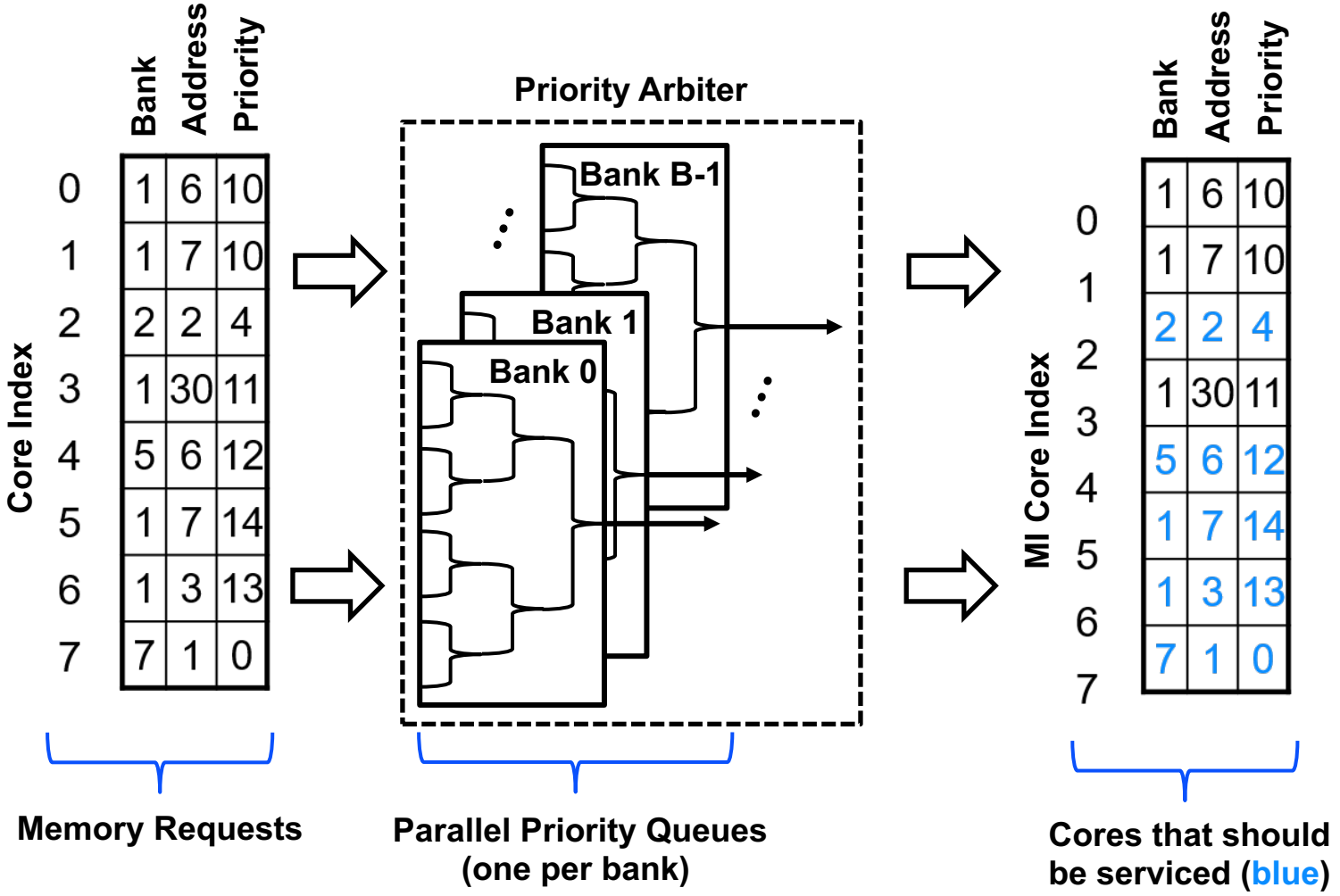
Stage 2:
Grant these extracted requests (grey)

Bank	Address	Priority
5	19	5
7	13	4
1	12	5
5	7	4
7	12	6
4	2	4
7	7	5
5	2	4

Stage 3:
Decrease the priorities of cores whose requests are granted (red) and accept next requests from these cores (blue)

Hardware Architecture for the Priority Arbiter

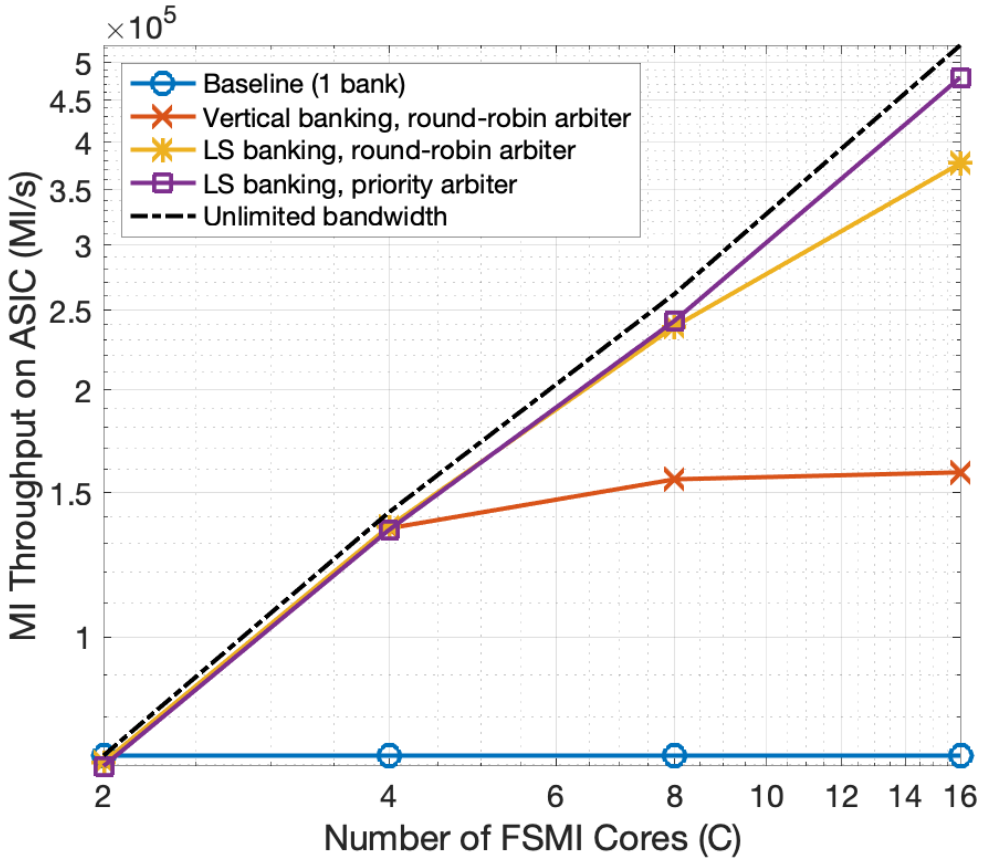
- **Fast:** Critical path scales with O(log(C)), where C denotes the number of cores



Results

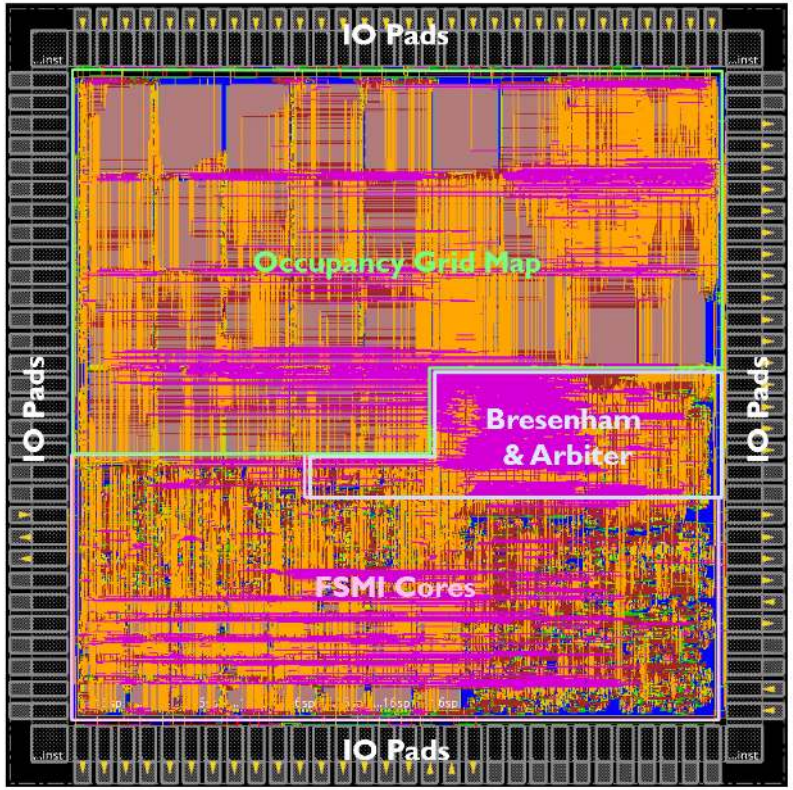
System Performance

System throughput (purple) at **91%** of theoretical limit (dotted black line)



ASIC Layout using 65nm Technology

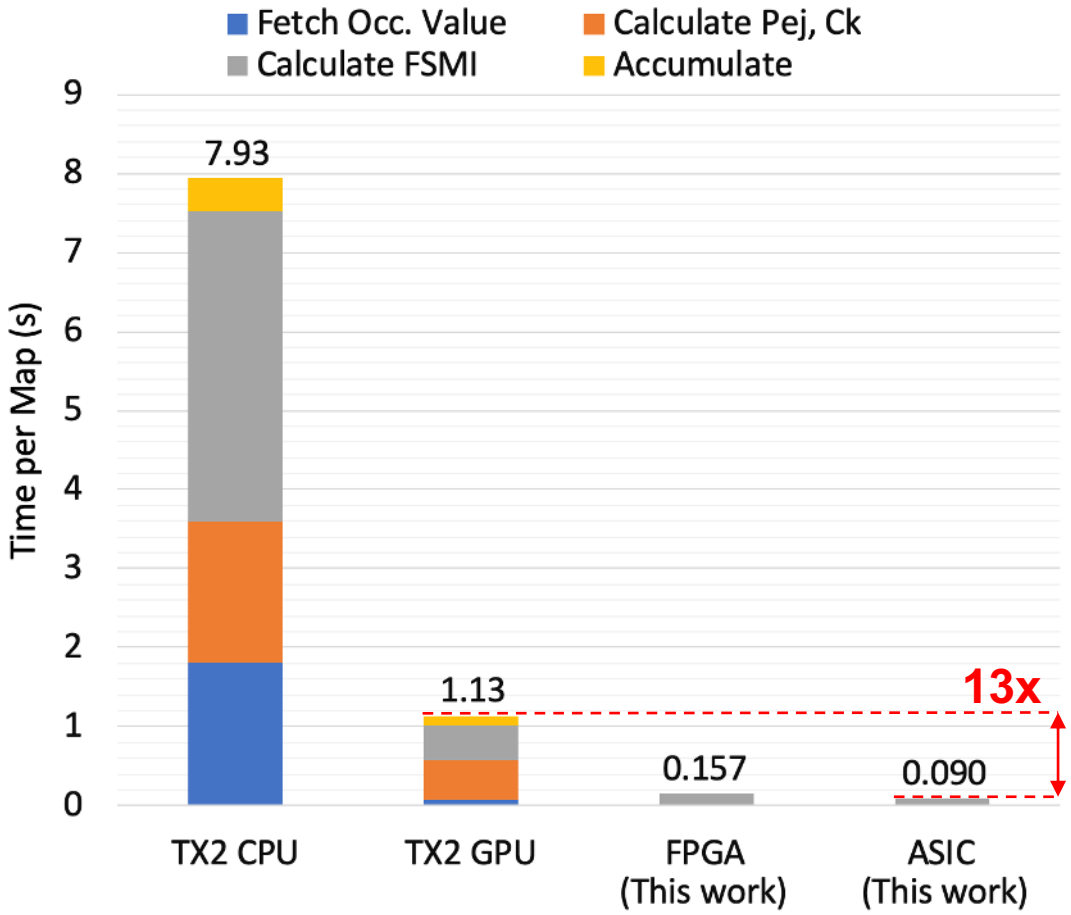
Chip area: 2550umx2550um, **Gates:** 3 million
Core clock: 116.5MHz, **Power:** 164mW



ASIC Throughput & Power

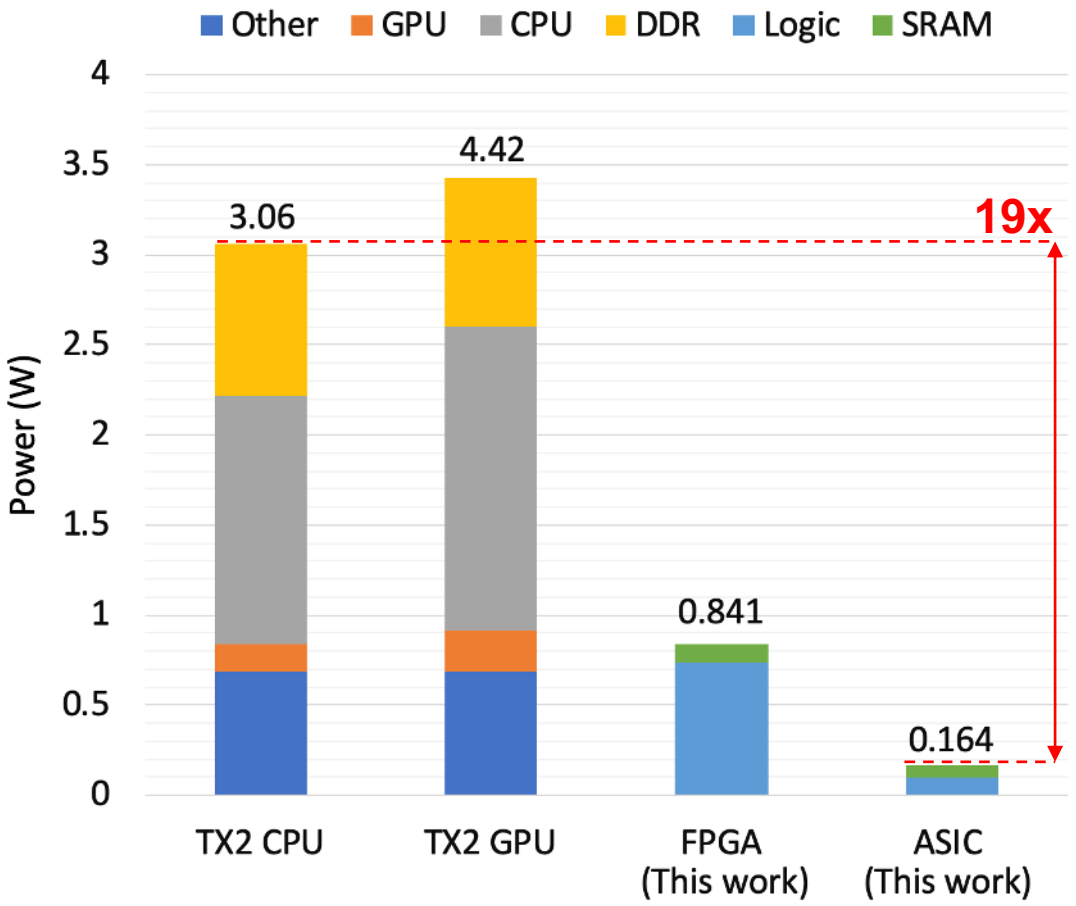
Compute Time per MI Map

13x faster for computing MI map of size 201x201 on ASIC vs. NVIDIA Pascal GPU on TX2



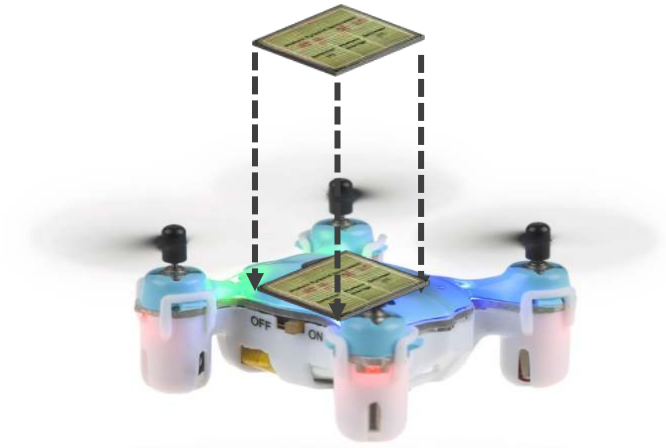
System Power

19x lower power on ASIC vs. ARM Cortex-A57 CPU on TX2



Summary

- Memory bandwidth limitation for MI computation is resolved by:
  1. Provably optimal class of Latin-square banking patterns
  2. Greedy and fair priority arbiter
- Computes the MI for the entire map of size 201x201 for the **first time at 90ms** (< 7s on CPU) on an ASIC while consuming only **164mW**.
- Real-time MI map computation allows the robot to choose the **most optimal exploration trajectory in a theoretically proven exploration pipeline** and **reduce exploration time.**



P. Z. X. Li*, Z. Zhang*, S. Karaman, V. Sze, "High-throughput Computation of Shannon Mutual Information on Chip," *Robotics: Science and Systems (RSS)*, June 2019.

Website: https://lean.mit.edu/highlights/mutual-information