



Compiling Image Processing Applications for Many-Core Accelerators

Pierre Guillou

► **To cite this version:**

Pierre Guillou. Compiling Image Processing Applications for Many-Core Accelerators. Journées de seconde année de l'Ecole Doctorale, Jun 2015, Paris, France. hal-01178938

HAL Id: hal-01178938

<https://hal-mines-paristech.archives-ouvertes.fr/hal-01178938>

Submitted on 21 Jul 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

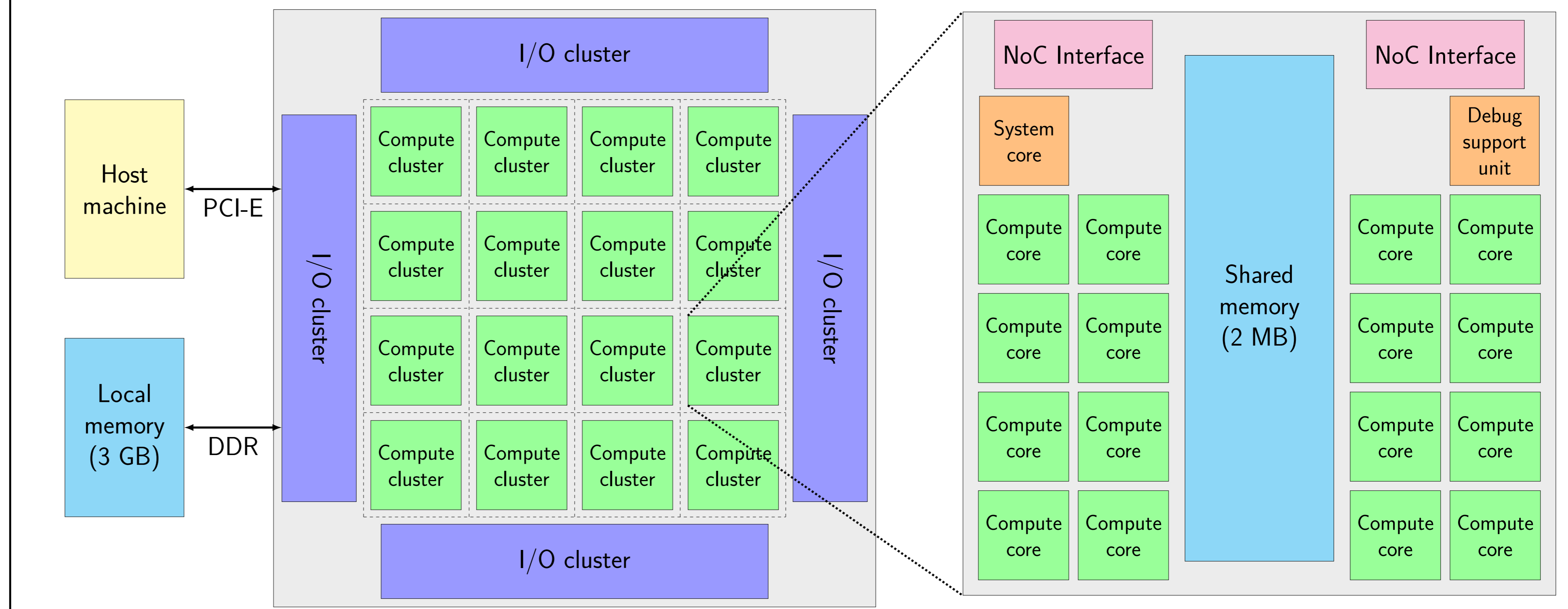
Image Processing

image analysis: detect geometrical structures in an image
 mathematical morphology: image analysis theory and technique based on lattices theory

Mathematical Morphology Base Operators

- arithmetic operators
 - unary (pixel \otimes parameter, 1 input image)
 - binary (pixel \otimes pixel, 2 input images)
 - $+ - \times \div \min \max = \& | \sim$
 - morphological operators
 - stencils
 - neighbor selection $+ \min/\max/\text{avg}$
 - reduction operators
 - global $\max/\min/\text{sum}$
 - other operators
 - threshold, mask, \log_2, \dots
- ⇒ Sigma-C agent library

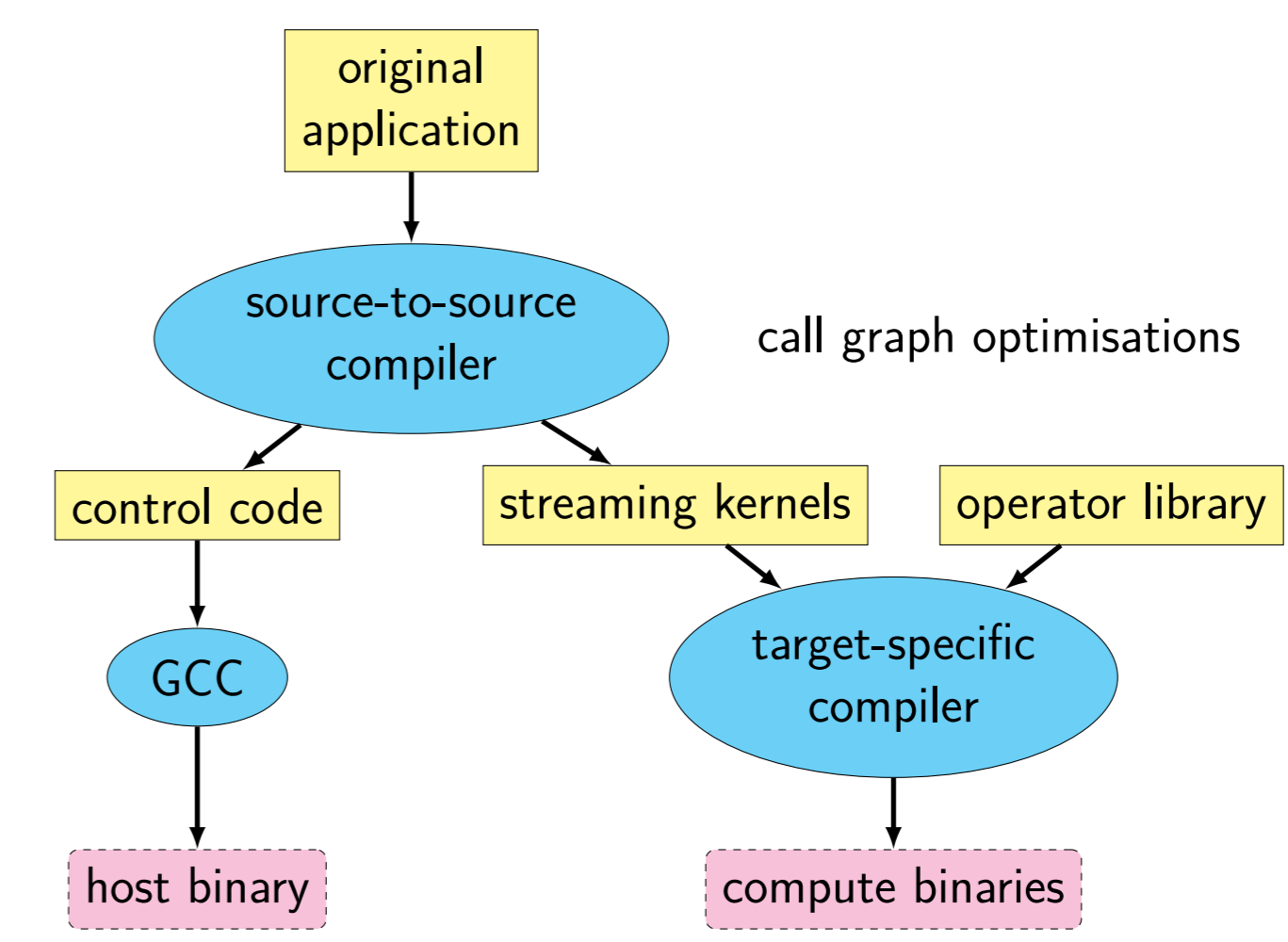
The MPPA-256 Chip



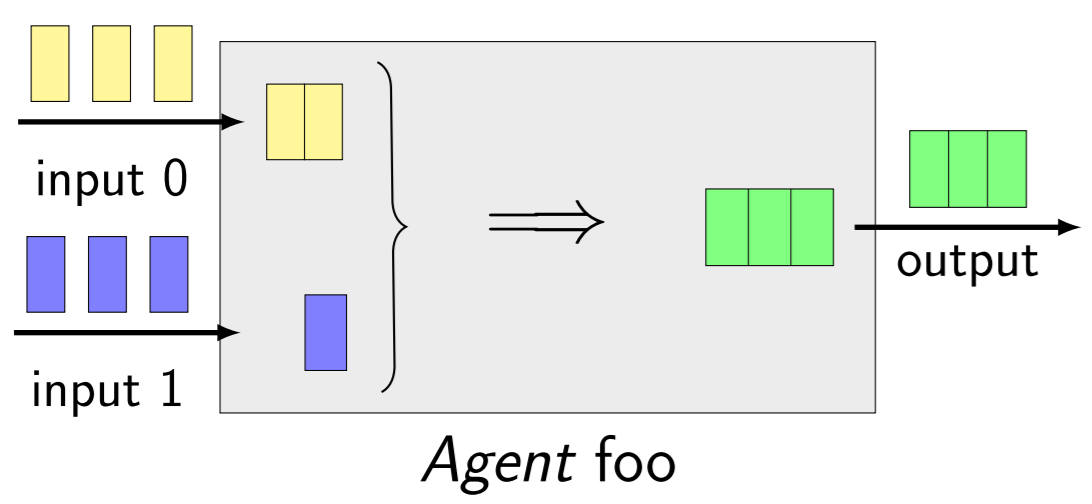
Example: Licence Plate Extraction



Compilation Chain

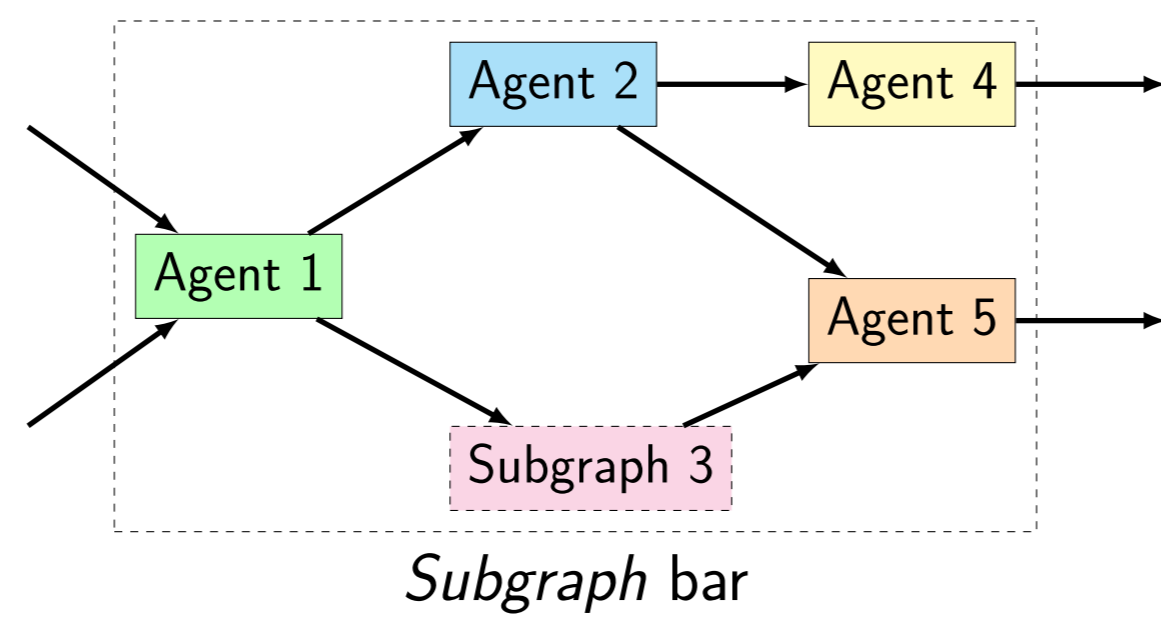


Sigma-C, a Dataflow Programming Language

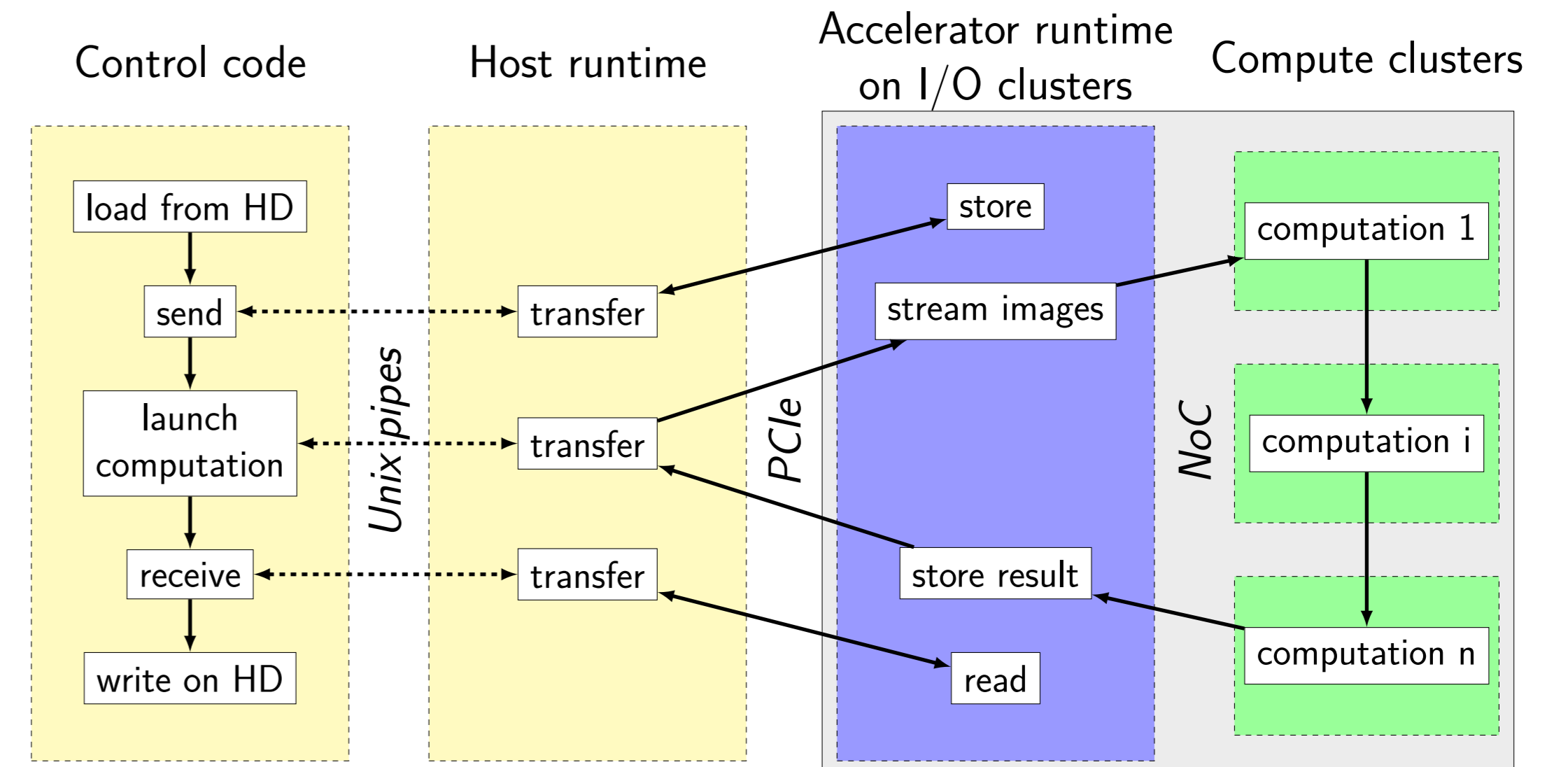


```
agent foo() {
  // describe agent interface
  interface {
    in<int> input0, input1;
    out<int> output;
    // declare the state machine
    spec{input0[2], input1, output[3]};
  }
  // loop over the state
  void start() exchange (input0 inp0[2], input1 inp1,
    output outp[3]) {
    outp[0] = inp0[0];
    outp[1] = inp1;
    outp[2] = inp0[1];
  }
}
```

```
subgraph bar() {
  // describe subgraph interface
  interface { /* ... */ }
  map {
    // instantiate agents
    agent a1 = new Agent1();
    agent a3 = new Subgraph3(); // ...
    // connect agents to subgraph interfaces
    connect (input0, a1.input0);
    connect (a5.output, output1); // ...
    // connect agents
    connect (a1.output0, a2.input);
    connect (a3.output, a5.input1); // ...
  }
}
```



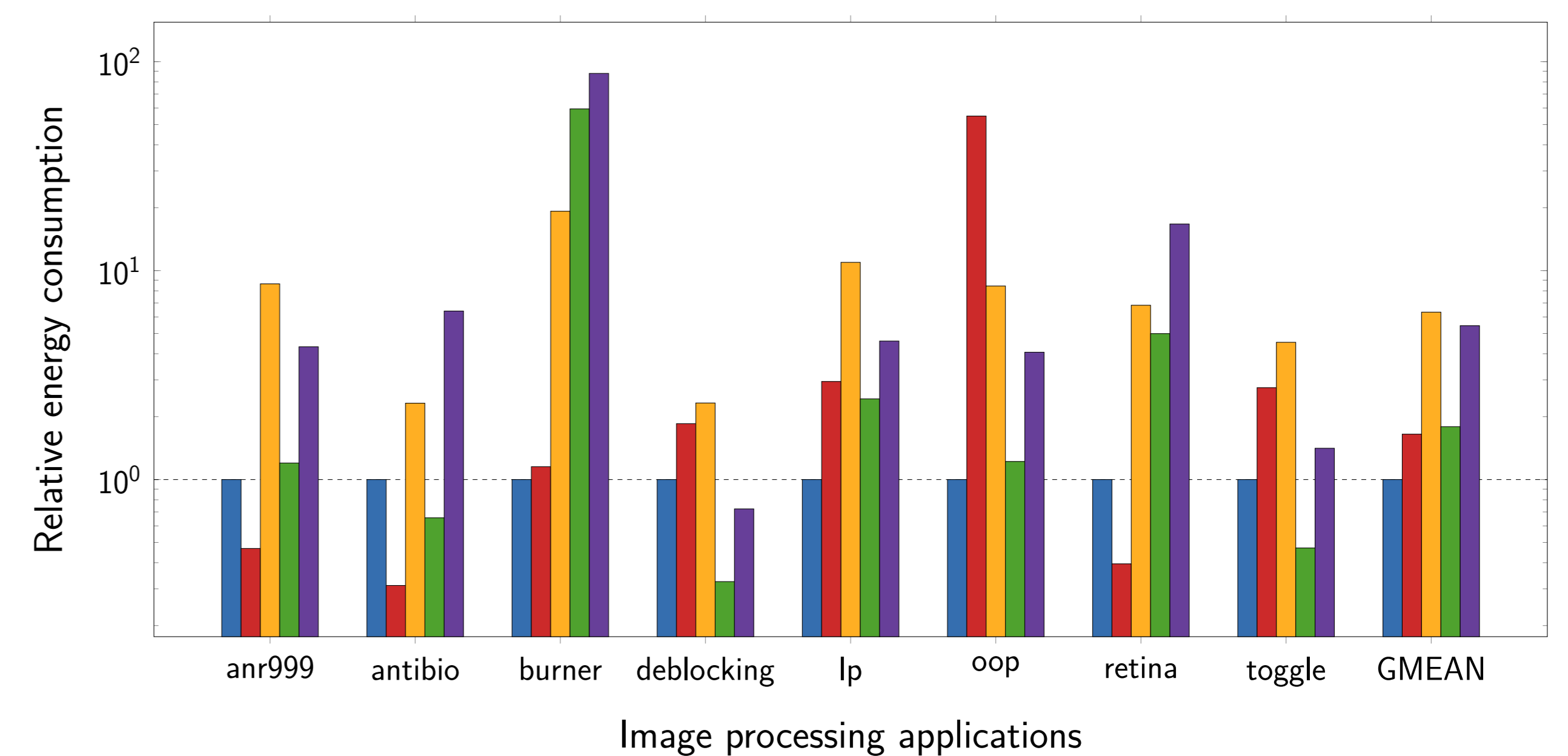
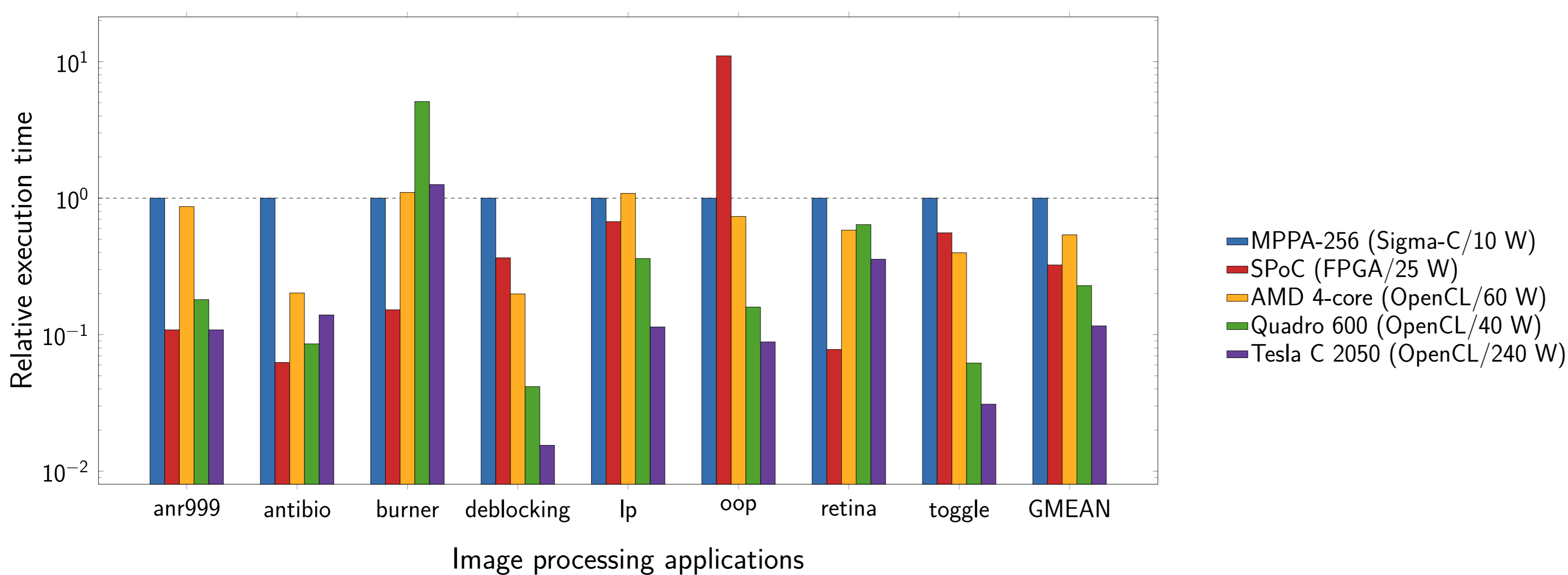
Runtime Environment



Optimisations

- unrolling of converging loops
- arithmetic operators aggregation
- generation of kernel-specific convolutions
- data parallelization for compute-intensive operators

Results: Execution Times and Energy Consumption (MPPA-256 = 1, lower is better)



Future Work

- Other programming models:
 - Pthreads/OpenMP on compute clusters, communication library between clusters
 - OpenCL via local memory pagination
- Improve data-parallelism to take better advantage of the current architecture
- Implement more complex algorithms: watershed, arrow, labelling, minima, ...

References

Pierre Guillou, Fabien Coelho, and François Irigoien.
 Automatic Streamization of Image Processing Applications.
 The 27th International Workshop on Languages and Compilers for Parallel Computing (LCPC), 2014.
 Available at <http://www.cri.enscm.fr/classement/doc/A-570.pdf>.

