Compiling Image Processing Applications for Many-Core Accelerators
Pierre Guillou

To cite this version:
Pierre Guillou. Compiling Image Processing Applications for Many-Core Accelerators. ACACES Summer School: Eleventh International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems, Jul 2015, Fiuggi, Italy. hal-01254412

HAL Id: hal-01254412
https://hal-mines-paristech.archives-ouvertes.fr/hal-01254412
Submitted on 12 Jan 2016

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.
Compiling Image Processing Applications for Many-Core Accelerators

Pierre Guillou – CRI MINES ParisTech, PSL Research University

Image Processing

- Mathematical Morphology Base Operators
  - arithmetic operators
    - threshold
    - erosion
    - dilation
    - neighbor
  - reduction operators
    - global max/min/sum
    - neighbor selection + min/max/avg
  - other operators
    - user-defined
  - unrolling of loops
  - mathematical morphology: image analysis theory and technique based on lattices theory

The MPPA-256 Chip

- Host machine
- PCI-E
- Local memory (3 GB)
- DDR
- I/O cluster
- Compute cluster
- NoC Interface
- Call graph optimisations
- Operator library
- Compute clusters
- NoC Interface
- Debug support unit
- System core
- Compute core
- Compute core
- Compute core
- Compute core
- Compute core
- Shared memory (2 MB)
- Data-parallelism to take better advantage of the current architecture

Example: Licence Plate Extraction

Input
Output

Sigma-C, a Dataflow Programming Language

```
agent foo() { ...
  // describe agent interface
  interface (
    input0 inp0 [2], output outp [3]);
  // describe subgraph interface
  subgraph bar() {
    // describe subgraph interface
    interface { /* ... */ }
  }
  // loop over the state
  void run(agent) exchange (input inp0 [2], output outp [3]) {
    inp0 [1] = inp0 [0];
    inp1 = inp1 ;
    outp [2] = inp0 [0];
  }
}
```

Compilation Chain

- Host binary
- GCC
- Unix pipes
- Compiler
- Runtime Environment
- Accelerator runtime
- Compute binaries
- Compute clusters
- NoC Interface
- Control code
- Accelerator runtime
- Compute clusters
- Data-parallelism to take better advantage of the current architecture
- unrolling of loops
- arithmetic operators aggregation
- generation of hardware-specific customizations
- data parallelization for compute-intensive operators

Optimisations

- unrolling of converging loops
- arithmetic operators aggregation
- generation of hardware-specific customizations
- data parallelization for compute-intensive operators

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

Future Work

- Other programming models
- Improvements to 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 Irigoin.
Automatic Compilation of Image Processing Applications.
The 27th International Workshop on Languages and Compilers for Parallel Computing (LCPC), 2014.