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.
Mathematical Morphology Base Operators

- Arithmetic operators
  - Unary: (pixel = parameter, input image)
  - Binary: (pixel = pixel, 2 input images)
- Morphological operators
- Stencil
- Neighborhood: max, min, avg
- Reduction operators
- Global max/min/sum
- Other operators
  - Threshold, mask, log2, ...
- Mathematical morphology: Image analysis theory and technique based
  - Image analysis: Detect geometrical structures in an image
- Other operators
  - Reduction operators
  - Morphological operators
  - Arithmetic operators

Implement more complex algorithms: watershed, arrow, labelling, minima, ...

Improve data-parallelism to take better advantage of the current architecture.

Other programming models:
- Relative execution time
- Threshold, mask, log2, ...
- Global max/min/sum
- Neighbor selection: min/max/avg
- + − × ÷ binary (pixel, 2 input images)
- ⊗ ⊗ & min max
- | ∼ pixel, 2 input images

Image Processing

- Image analysis: Detect geometrical structures in an image
- Mathematical morphology: Image analysis theory and technique based on lattice theory

The MPPA-256 Chip

- NoC Interface
- Compute clusters
- Debbug support unit
- Shared memory (2 MB)

Compilation Chain

- Host machine
- Local memory (3 GB)
- I/O cluster
- PCI-E

Sigma-C, a Dataflow Programming Language

```
agent foo () {
  // describe agent interface
  interface ( /* ... */ );
  // describe agent body
  void start () exchange ( input0 inp0 [2] , input1 inp1 , output outp [3] ) {
    // loop over the state
    for ( int i = 0 ; i < inp0 [2] ; i++ ) {
      outp [i] = inp1 ;
      outp [2] = inp0 [i];
    }
  }
}
```

Sigma-C, a Dataflow Programming Language

```
agent foo () {
  // describe agent interface
  interface ( /* ... */ );
  // describe agent body
  void start () exchange ( input0 inp0 [2] , input1 inp1 , output outp [3] ) {
    // loop over the state
    for ( int i = 0 ; i < inp0 [2] ; i++ ) {
      outp [i] = inp1 ;
      outp [2] = inp0 [i];
    }
  }
}
```

Example: Licence Plate Extraction

```
Input
Output
```

Compilation Chain

- Source-to-source compiler
- Call graph optimisations
- Instrumentation
- Target-specific compiler
- Compute kernel
- Compute binary

Runtime Environment

- Control code
- Load from HD
- Compute binaries
- Compute kernels
- Compute framework
- Compute libraries
- Compute memory
- Compute runtime
- Compute environment
- Compute cores
- Compute clusters
- Compute I/O clusters
- Compute accelerators
- Compute system

Optimisations

- Unrolling of converging loops
- Arithmetic operators aggregation
- Generation of host-specific convolutions
- Data parallelization for compute-intensive operators

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

```
<table>
<thead>
<tr>
<th>Application</th>
<th>MPPA-256</th>
<th>Tesla C 2050 (OpenCL/240 W)</th>
<th>Quadro 600 (OpenCL/40 W)</th>
<th>AMD 4-core (OpenCL/60 W)</th>
<th>SPoC (FPGA/25 W)</th>
<th>MPPA-256 (Sigma-C/10 W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>anr999</td>
<td>10^9</td>
<td>10^7</td>
<td>10^8</td>
<td>10^9</td>
<td>10^10</td>
<td>10^2</td>
</tr>
<tr>
<td>antibiotic</td>
<td>10^4</td>
<td>10^2</td>
<td>10^3</td>
<td>10^4</td>
<td>10^5</td>
<td>10^0</td>
</tr>
<tr>
<td>burner</td>
<td>10^6</td>
<td>10^4</td>
<td>10^5</td>
<td>10^6</td>
<td>10^7</td>
<td>10^1</td>
</tr>
<tr>
<td>deblinking</td>
<td>10^7</td>
<td>10^5</td>
<td>10^6</td>
<td>10^7</td>
<td>10^8</td>
<td>10^2</td>
</tr>
<tr>
<td>dp</td>
<td>10^8</td>
<td>10^6</td>
<td>10^7</td>
<td>10^8</td>
<td>10^9</td>
<td>10^3</td>
</tr>
<tr>
<td>h264</td>
<td>10^9</td>
<td>10^7</td>
<td>10^8</td>
<td>10^9</td>
<td>10^10</td>
<td>10^4</td>
</tr>
<tr>
<td>retina</td>
<td>10^10</td>
<td>10^8</td>
<td>10^9</td>
<td>10^10</td>
<td>10^11</td>
<td>10^5</td>
</tr>
<tr>
<td>toggle</td>
<td>10^11</td>
<td>10^9</td>
<td>10^10</td>
<td>10^11</td>
<td>10^12</td>
<td>10^6</td>
</tr>
<tr>
<td>GMEAN</td>
<td>10^12</td>
<td>10^10</td>
<td>10^11</td>
<td>10^12</td>
<td>10^13</td>
<td>10^7</td>
</tr>
</tbody>
</table>
```

Future Work

- Other programming models:
  - Pilot56s/Spin6P 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 Coudoix, and François Irigoin.
Automatic Streamlining of Image Processing Applications.
The 21st International Workshop on Languages and Compilers for Parallel Computing (LCPC), 2014.