Synthesizing High-Performance Image Processing Applications with Hipacc

Throughput in [MPixel/s]

<table>
<thead>
<tr>
<th>Device</th>
<th>Tegra K1</th>
<th>Tesla K20</th>
<th>Stratix V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>100</td>
<td>101</td>
<td>102</td>
</tr>
</tbody>
</table>

Energy Efficiency in [MPixel/W]

<table>
<thead>
<tr>
<th>Device</th>
<th>Tegra K1</th>
<th>Tesla K20</th>
<th>Kintex 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>150</td>
<td>160</td>
<td>170</td>
</tr>
</tbody>
</table>

A DSL for Image Processing

Hipacc Classes
- Image: input/output buffers
- Accessor: ROI of input image
- BoundaryCondition: boundary handling
- InterpolationRegion: interpolation/filtering
- IterationSpace: iteration domain
- Kernel: compute kernel description
- Mask: convolution mask
- Domain: domain description
- Pyramid: image pyramid description

Compiler Features
- Memory padding/alignment
- Utilization of textures
- Exploit full GPU memory hierarchy
- MPMD code generation
- Loop unrolling
- Constant propagation
- Thread-coarsening
- Multiple pixels per thread
- Vectorization support
- Implicit use of unified CPU/GPU memory

DSL Code Example: Gaussian Blur

DSL Host Code
```c++
// ...,
void kernel() {
    float sum = 0;
    for (int i = 0; i < mask.size(); i++) {
        sum += mask[i] * acc[i];
    }
    output = convert_uchar4(sum);
}
```

DSL Kernel Code
```c++
class GaussianBlur : public Kernel<uchar4> {
    // ...
    void kernel() {
        float sum = convolve(mask, HipaccSUM, [&]() {
            return mask()
                .convert_float4(input(mask));
        });
        output() = convert uchar4(sum + 0.5f);
    }
}
```

Generating the Streaming Pipeline [4]

Use internal representation to transform the execution order of Hipacc kernels into a pipelined and streamed execution:
- Build up dependency graph
- Identify memory reuse
- Prune dead code by traversing in depth-first search starting from output buffers

Loop Coarsening [3]

Vectorization by loop coarsening:
- Replicate only the data path: sublinear increase of resource usage
- Better exploitation of bandwidth: nearly linear speedup

Results

Demonstrator

Hardware
- DE1-SoC with Altera Cyclone-V
- Dell Latitude E6440 Notebook
  - Intel Core i5-4300M CPU
  - AMD-Radeon-HD-8690M GPU

Algorithms
- Preprocessing: Harris Corner, Bilateral, Sobel, Optical Flow
- Postprocessing: Bokeh Effect, Night Filter

Selected Publications


https://hipacc-lang.org/