### Automatic GPU Code Generation for Android

**HIPA®** – [http://hipacc-lang.org](http://hipacc-lang.org)

Oliver Reiche, Richard Membarth, Frank Hannig, and Jürgen Teich

### A DSL for Image Processing

**HIPA®**: The Heterogeneous Image Processing Acceleration Framework

<table>
<thead>
<tr>
<th>DSL code embedded into C++</th>
<th>Compiler Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain knowledge captured by classes</td>
<td>- memory padding / alignment</td>
</tr>
<tr>
<td><strong>HIPA® Classes</strong></td>
<td>- utilization of textures</td>
</tr>
<tr>
<td>Image</td>
<td>- exploit full GPU memory hierarchy</td>
</tr>
<tr>
<td>Accessor</td>
<td>- MPMD code generation</td>
</tr>
<tr>
<td>ROI of input image</td>
<td>- loop unrolling</td>
</tr>
<tr>
<td>IterationSpace</td>
<td>- constant propagation</td>
</tr>
<tr>
<td>ROI of output image</td>
<td>- thread-coarsening (multiple pixels per thread)</td>
</tr>
<tr>
<td>Kernel</td>
<td>- implicit use of unified CPU/GPU memory</td>
</tr>
<tr>
<td>compute kernel description</td>
<td></td>
</tr>
<tr>
<td>Mask</td>
<td></td>
</tr>
<tr>
<td>convolution mask</td>
<td></td>
</tr>
<tr>
<td>Domain</td>
<td></td>
</tr>
<tr>
<td>iteration domain</td>
<td></td>
</tr>
<tr>
<td>Pyramid</td>
<td></td>
</tr>
<tr>
<td>image pyramid description</td>
<td></td>
</tr>
</tbody>
</table>

Optimizations tailored to GPU architectures

#### DSL Code Example: Gaussian Blur

**DSL Host Code**

```cpp
global float4 myGaussian(float4 input, float4 mask) {
  float4 sum = convolve(mask, HipaccSUM, [&]() -> float4 {
    // ...%
  });
  return mask() * convert_float4(input(mask));
}
```

**DSL Kernel Code**

```cpp
class GaussianBlur : public Kernel<uchar4> {
  // ...
  void kernel() {
    GlobalIntMat* filter = new float[4];
    GaussianBlurFilter<uchar4>(filter, acc, mask);
    myGaussian(sum, mask);
  }
};
```

### Seamless Integration into the Android Development Tools

#### Android Software Development Kit (SDK)
- based on Eclipse
- supports C++ via Java Native Interface (JNI)
- automatic compilation and packaging into app
- compilation done by the Android Native Development Kit (NDK)

#### GPU Computing on Android
- RenderScript Compute
  - code mapping to native worker threads
  - targets DSPs, CPUs, and GPUs (since Android 4.2)

- Filterscript
  - stricter limitations
  - relaxed precision
  - no scatter writes
  - pointers are illegal
  - ensures wider compatibility

### HIPA® Integration

**Modular Makefile**

- Used to seamlessly integrate HIPA® into ADT
- called during preprocessing step
- set appropriate target compiler flags
- append generated files to NDK sources

### Demonstration Setup and Results

**Exynos 5250 MPSoC**
- ARM Cortex-A15
- dual core @1.7 GHz
- 64/128-bit SIMD NEON
- ARM Mali T-604 GPU
- 4 cores @533 MHz
- 16 SIMD lanes per core
- 2 GB of 800 MHz DDR3 DRAM

**Live Demo Application**
- five image filters in DSL code
- single description of filters
- target-independent code
- automatic code generation for Renderscript and Filterscript
- up to 4× faster

*Figure: Samsung Exynos 5250 Androide Board*