Accelerating OpenVX Applications on Embedded Many-Core Accelerators

Giuseppe Tagliavini, DEI (University of Bologna)
Germain Haugou, IIS-ETHZ
Andrea Marongiu, DEI (University of Bologna) & IIS-ETHZ
Luca Benini, DEI (University of Bologna) & IIS-ETHZ
Outline

- Introduction
- OpenVX acceleration
- Work in progress
OpenVX overview

- Foundational API for vision acceleration
  - Focus on mobile and embedded systems
- Stand-alone or complementary to other libraries
- Enable efficient implementations on different devices
  - CPUs, GPUs, DSPs, many-core accelerators
Accelerator template

Multi-core processor

Cluster-based design

Cluster memory (optional)

Host

Cluster controller (optional)

Low latency shared TCDM memory

DMA engine (L1 ↔ L3)

MPMD Processing Elements

Many-core accelerator

Cluster

... Cluster

L2

L3

DDR3 memory

SoC design (optional)

HW synchronizer
PULP
Parallel Ultra-Low-Power platform
OpenVX programming model

- The OpenVX model is based on a **directed acyclic graph** of nodes (**kernels**), with data (**images**) as linkage

```c
vx_image imgs[] = {
  vxCreateImage(ctx, width, height, VX_DF_IMAGE_RGB),
  vxCreateVirtualImage(graph, 0, VX_DF_IMAGE_U8),
  ...
};
```

**Virtual images are not required to actually reside in main memory**
- ✓ They define a data dependency between kernels, but they cannot be read/written

**An OpenVX program must be verified to guarantee some mandatory properties:**
- ✓ Inputs and outputs compliant to the node interface
- ✓ No cycles in the graph
- ✓ Only a single writer node to any data object is allowed
- ✓ Writes have higher priorities than reads.
- ✓ **Virtual image must be resolved into concrete types**

```c
vxVerifyGraph(graph);
vxProcessGraph(graph);
```
Outline

- Introduction
- OpenVX acceleration
- Work in progress
A first solution: using OpenCL to accelerate OpenVX kernels

- OpenCL is a commonly used programming model for many-core accelerators

- First solution: **OpenVX kernel == OpenCL kernel**
  - When a node is selected for execution, the related OpenCL kernel is enqueued on the device

- Main limiting factor: **memory bandwidth**
OpenCL bandwidth

- Experiments performed on the STHORM evaluation board

![Bar chart showing OpenCL bandwidth for various tasks](chart.png)
Our solution

- We realized an OpenVX framework for many-core accelerators coupling a tiling approach with algorithms for graph partition and scheduling.

- Main goals:
  - Reducing the memory bandwidth
  - Maximizing the accelerator efficiency

- Several steps are required:
  - Tile size propagation
  - Graph partitioning
  - Node scheduling
  - Buffer allocation
  - Buffer sizing
Common access patterns for image processing kernels

(A) POINT OPERATORS
Compute the value of each output point from the corresponding input point
Support: Basic tiling

(B) LOCAL NEIGHBOR OPERATORS
Compute the value of a point in the output image that corresponds to the input window
Support: Tile overlapping

(C) RECURSIVE NEIGHBOR OPERATORS
Like the previous ones, but also consider the previously computed values in the output window
Support: Persistent buffer

(D) GLOBAL OPERATORS
Compute the value of a point in the output image using the whole input image
Support: Host exec / Graph partitioning

(E) GEOMETRIC OPERATORS
Compute the value of a point in the output image using a non-rectangular input window
Support: Host exec / Graph partitioning

(F) STATISTICAL OPERATORS
Compute any statistical functions of the image points
Support: Graph partitioning
Example (1)

- MEMORY DOMAINS
  - L3
- NESTED GRAPH
- ACCELERATOR SUB-GRAPH
- ADAPTIVE TILING
- HOST NODE
Example (2)

PEs

CC

DMAin

DMAout

<table>
<thead>
<tr>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
</table>

L1 memory buffers

B0 and B5 use double buffering

L3 access
Bandwidth reduction

![Bar chart showing bandwidth reduction for various tasks and technologies. The chart compares OVX and OpenCL against available bandwidth.](chart.png)
Speed-up w.r.t. OpenCL

- Random graph: 6.73
- Edge detector: 3.86
- Object detection: 3.46
- Super resolution: 3.50
- FAST9: 2.81
- Disparity: 5.64
- Pyramid: 2.92
- Optical: 1.00
- Canny: 3.12
- Retina preproc.: 5.04
- Disparity S4: 9.61
Outline

- Introduction
- OpenVX acceleration
- Work in progress
OpenVX + Virtual Platform

ADRENALINE

Virtual Platform

TestApplications

Platform configuration

Run-time support

Run-time policies

Application mapping

http://www-micrel.deis.unibo.it/adrenaline/
THANKS!!!

Work supported by EU-funded research projects