University
POLITEHNICA
of Bucharest

Faculty of
Automatic Control
and Computers

Computer Science
and Engineering
Department

Compiler for Vector Processing and Frequency Scaling at Instruction Level

Master Project – September 2014

Author

Laura Vasilescu
laura.vasilescu@cti.pub.ro

Scientific Advisor(s)

ș.l. dr. ing. Lucian Petrică (ETTI)
ș.l. dr. ing. Răzvan Deaconescu (ACS)

biology

Computer Vision

biology

+

mathematics

Computer Vision

biology

+

mathematics

+

algorithms design

Computer Vision

biology

+

mathematics

+

algorithms design

+

hardware design

Computer Vision

biology

+

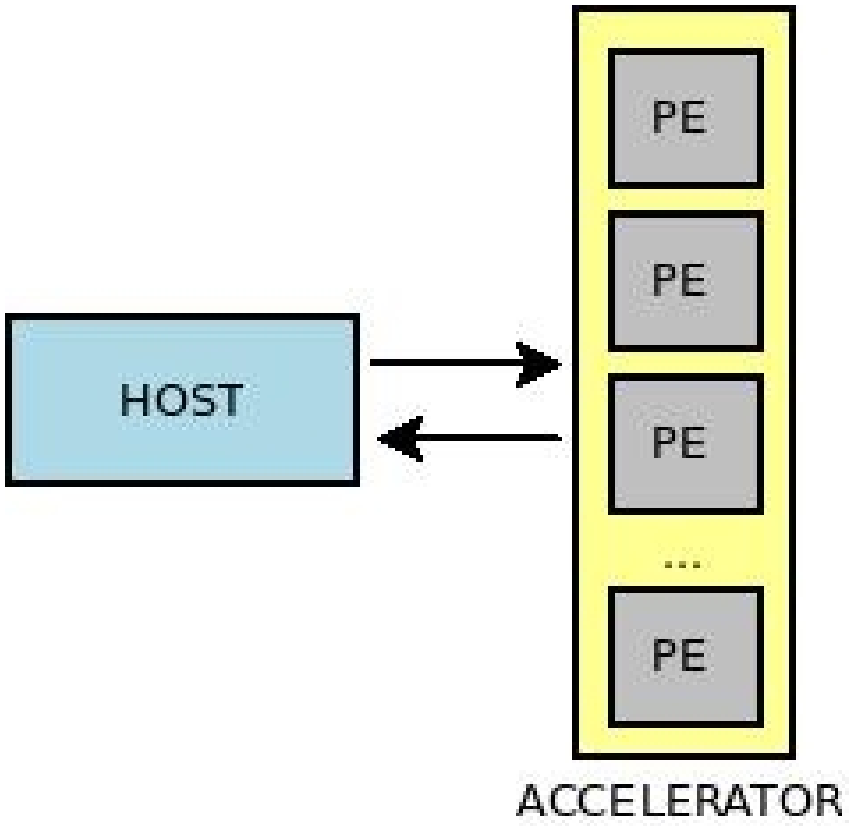
mathematics

+

algorithms design

+

hardware design



Starting Point

ConnexArray – architecture designed for computer vision applications

Starting Point

ConnexArray – architecture designed for computer vision applications

- development process: **heavy**

Starting Point

ConnexArray – architecture designed for computer vision applications

- development process: **heavy**
  - specific assembly

Starting Point

ConnexArray – architecture designed for computer vision applications

- development process: **heavy**
  - specific assembly
  - rewrite any application

Objectives

- run computer vision applications

Objectives

- run computer vision applications
- write applications in an architecture independent programming language

Objectives

- run computer vision applications
- write applications in an architecture independent programming language
- optimize the running process

Solution

- **Compiler for ConnexArray**

Solution

- **Compiler for ConnexArray**
  - modularity

Solution

- **Compiler for ConnexArray**
  - modularity
  - easy to extend

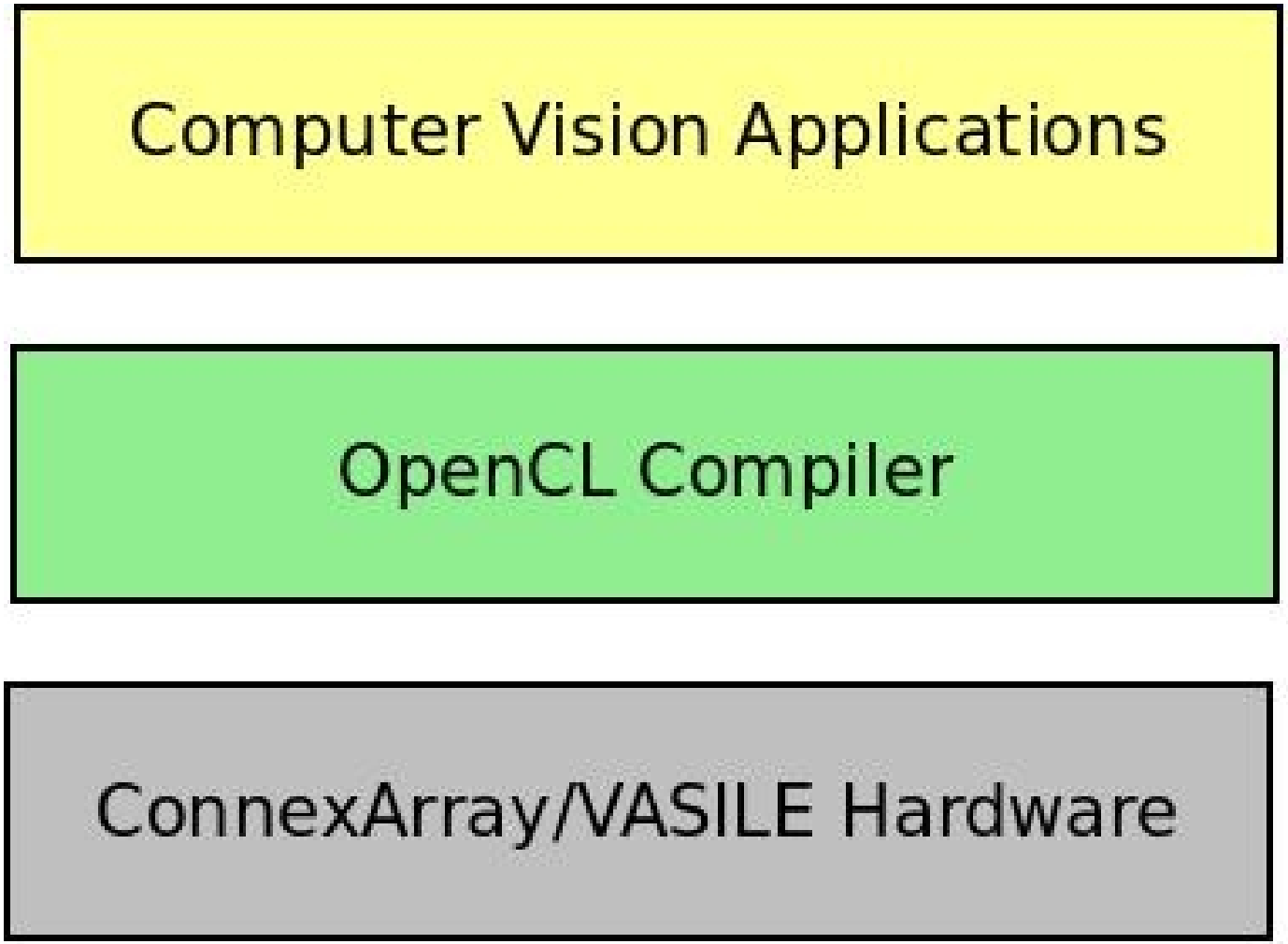
Solution

- **Compiler for ConnexArray**
  - modularity
  - easy to extend
  - no hard-constraints for a programming language

Solution

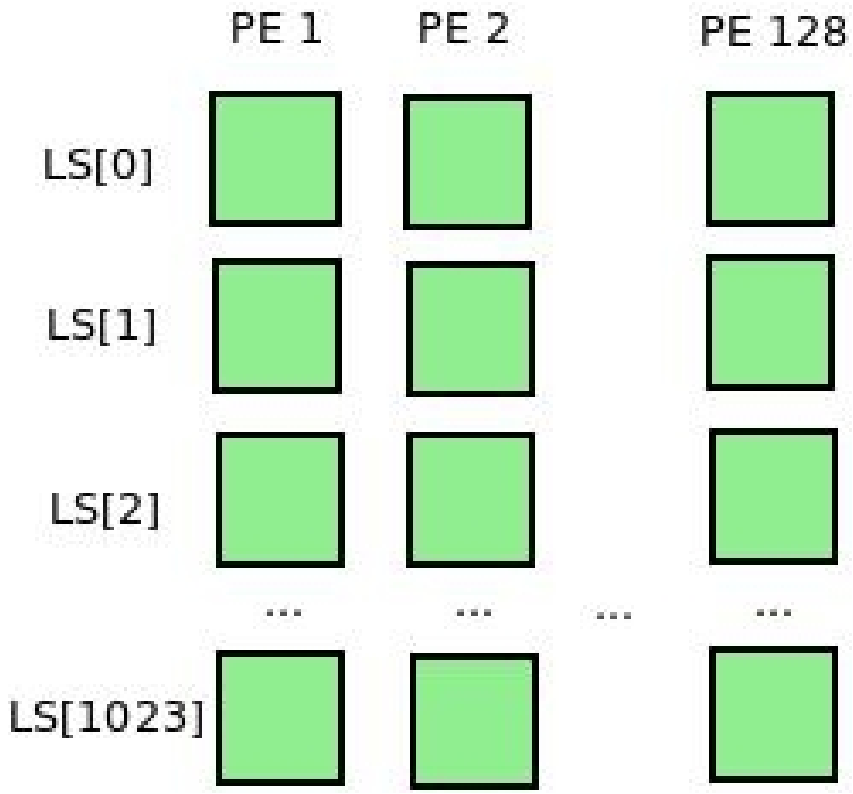
- **Compiler for ConnexArray**
  - modularity
  - easy to extend
  - no hard-constraints for a programming language
  - frequency scaling at instruction level (***VASILE***)

Overview


Computer Vision Applications
OpenCL Compiler
ConnexArray/VASILE Hardware

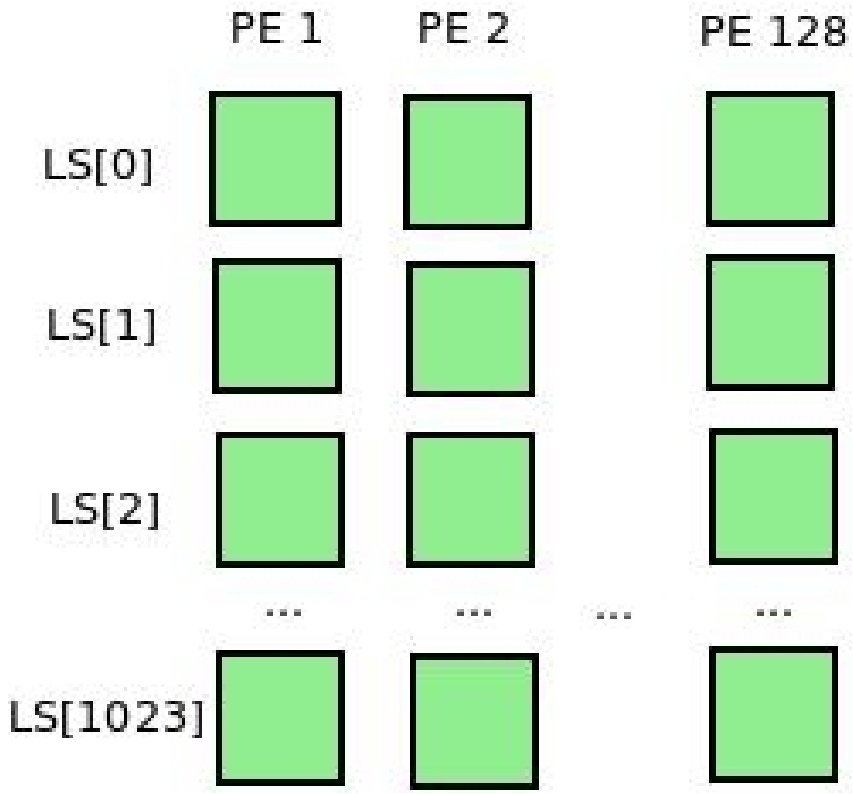
ConnexArray Architecture

- SIMD architecture



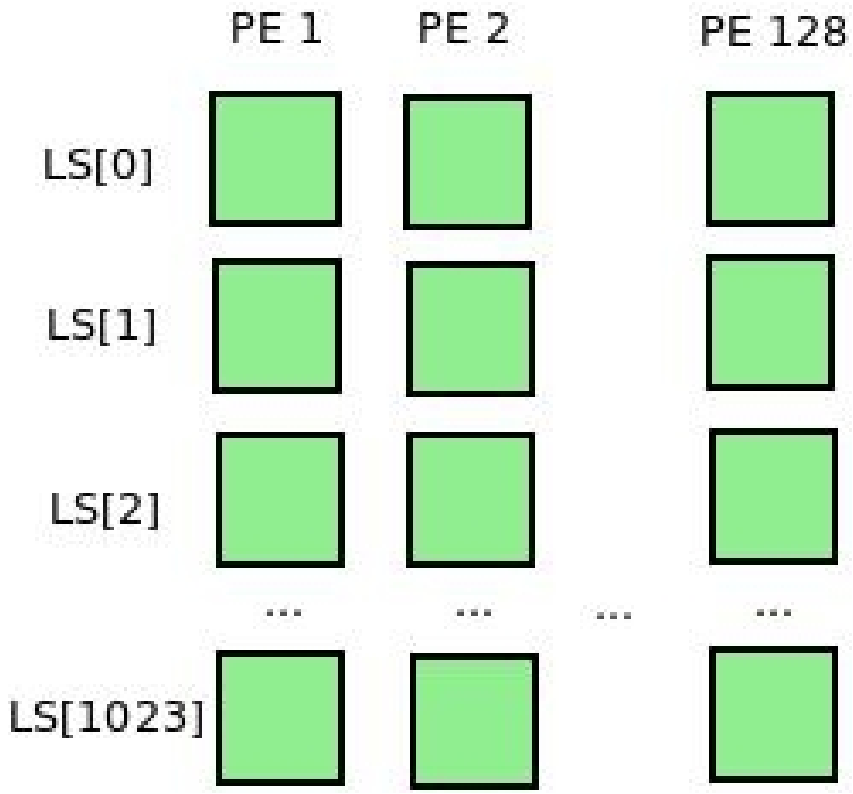
ConnexArray Architecture

- SIMD architecture
- 128 execution units



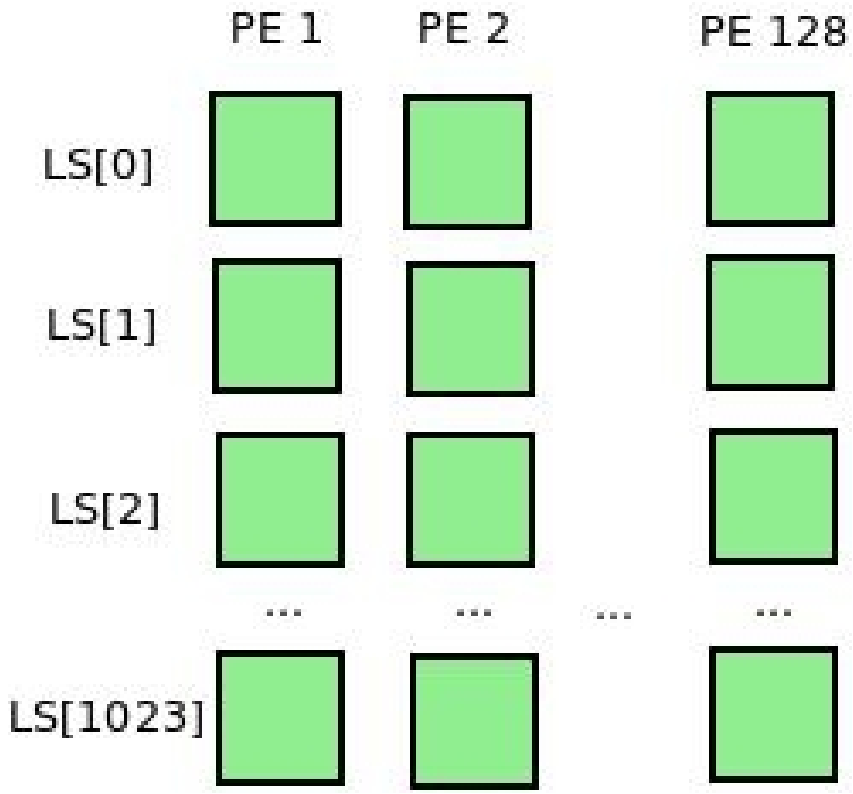
ConnexArray Architecture

- SIMD architecture
- 128 execution units
- implemented in FPGA



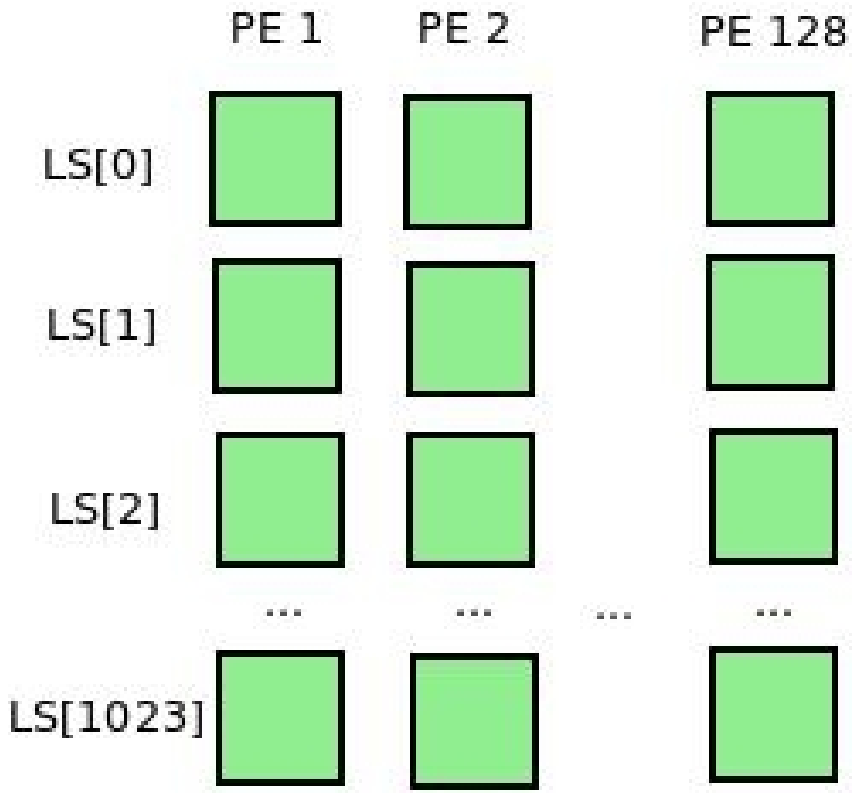
ConnexArray Architecture

- SIMD architecture
- 128 execution units
- implemented in FPGA
- custom ISA



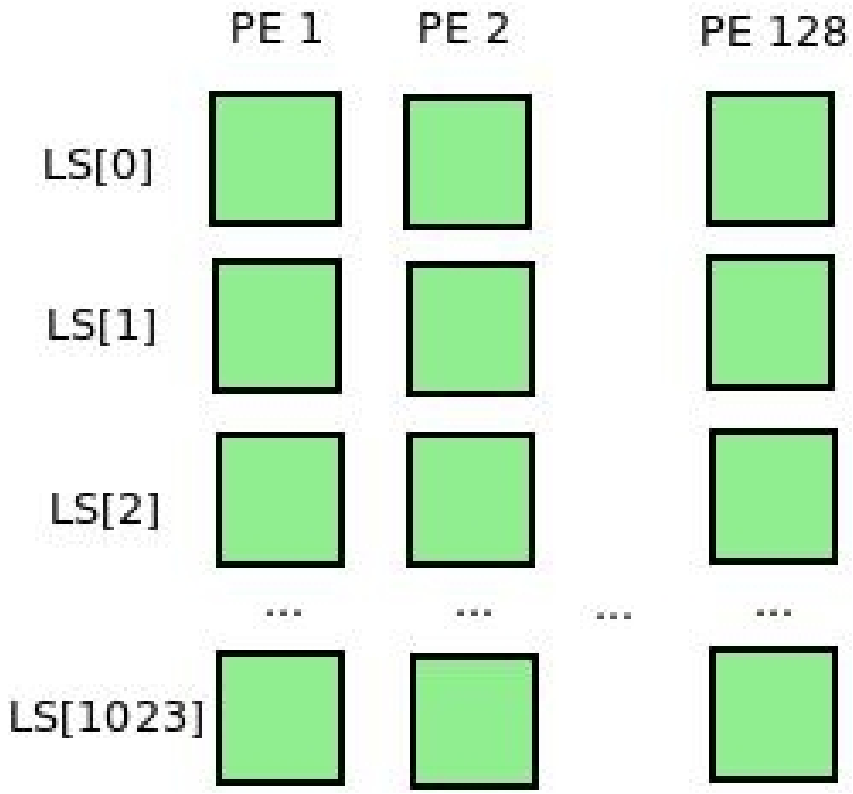
ConnexArray Architecture

- SIMD architecture
- 128 execution units
- implemented in FPGA
- custom ISA
- no control instructions



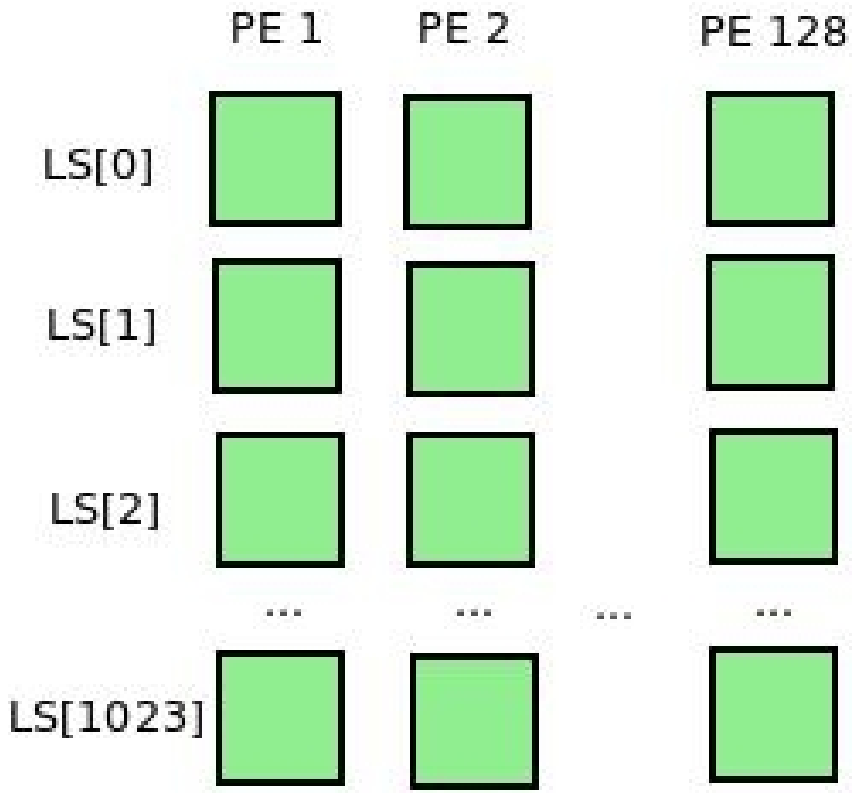
ConnexArray Architecture

- SIMD architecture
- 128 execution units
- implemented in FPGA
- custom ISA
- no control instructions
- Local Store: 1024 cells

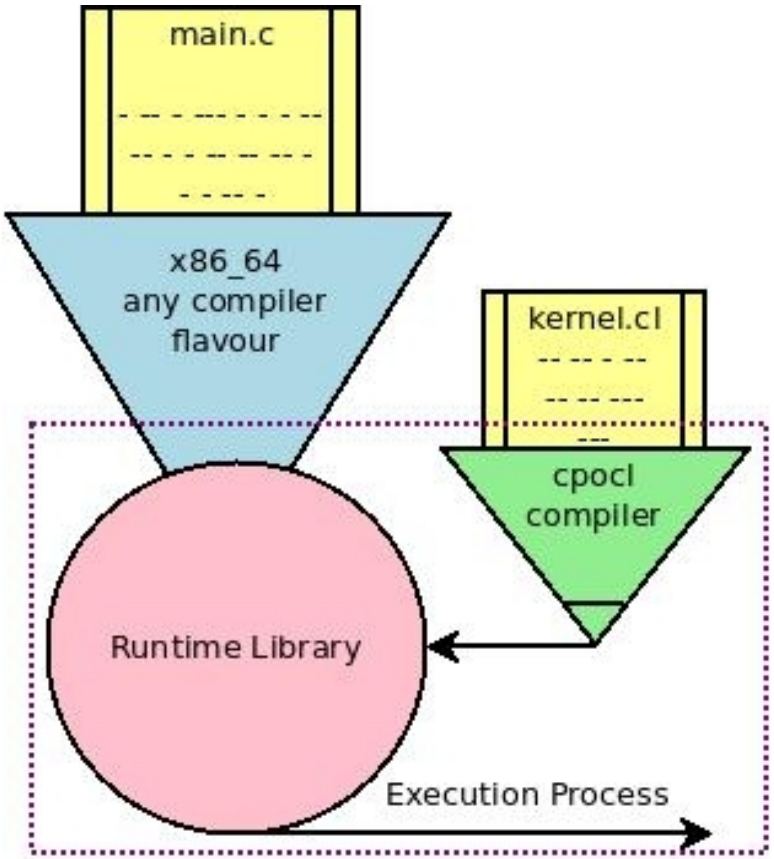


ConnexArray Architecture

- SIMD architecture
- 128 execution units
- implemented in FPGA
- custom ISA
- no control instructions
- Local Store: 1024 cells
- memory:
  128 x 1024 x 2

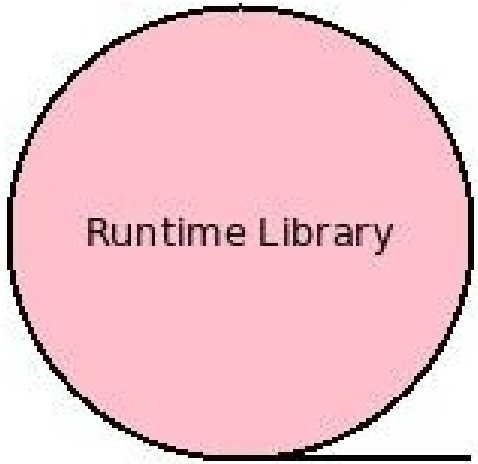


OpenCL Compiler Infrastructure



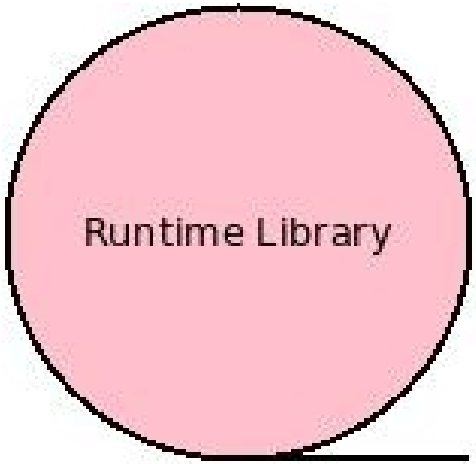
Runtime Library

- implements OpenCL standard API



Runtime Library

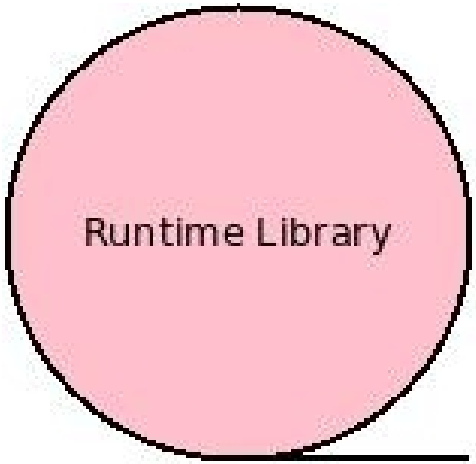
- implements OpenCL standard API
- memory management



Runtime Library

- implements OpenCL standard API

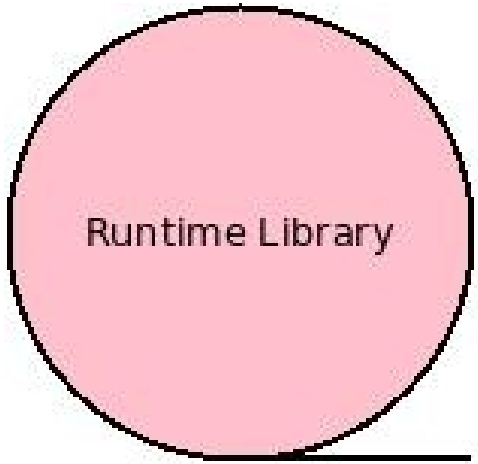
- memory management
- data transfers



Runtime Library

- implements OpenCL standard API

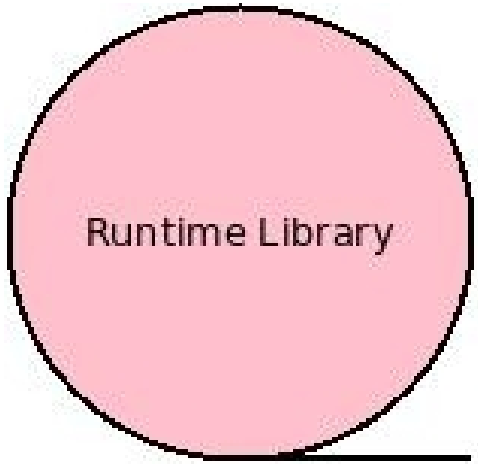
- memory management
- data transfers
- handles runtime compilation



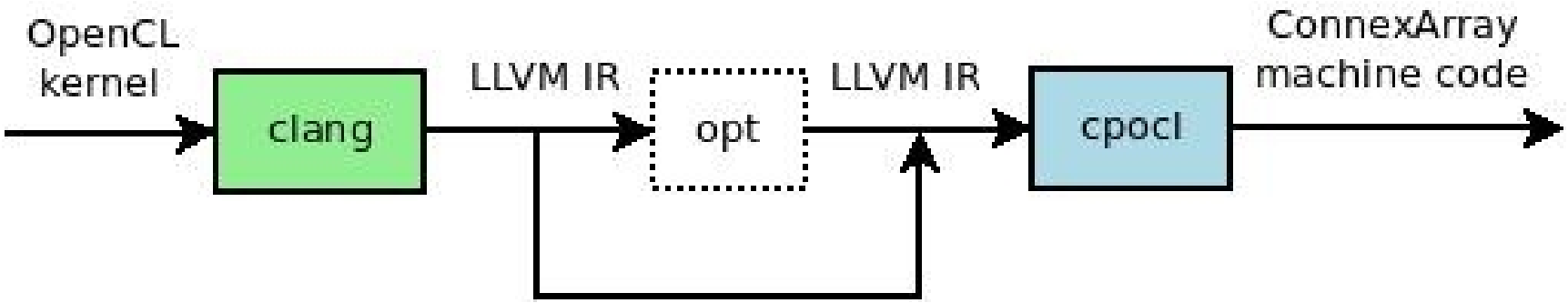
Runtime Library

- implements OpenCL standard API

- memory management
- data transfers
- handles runtime compilation
- handles execution flow



cpocl – ConnexArray CodeGen



VASILE optimization

- **V**ector **A**rchitecture for frequency **S**caling at **I**nstruction **LE**vel
  - ConnexArray enhancement

VASILE optimization

- **V**ector **A**rchitecture for frequency **S**caling at **I**nstruction **LE**vel
  - ConnexArray enhancement

Steps:

- hardware profiling

VASILE optimization

- **V**ector **A**rchitecture for frequency **S**caling at **I**nstruction **LE**vel
  - ConnexArray enhancement

Steps:

- hardware profiling
- determine max. frequencies

VASILE optimization

- **V**ector **A**rchitecture for frequency **S**caling at **I**nstruction **LE**vel
  - ConnexArray enhancement

Steps:

- hardware profiling
- determine max. frequencies
- compiler configuration – loop tiling

Hardware Profiling

SSD algorithm: $\sum(a_i - b_i)^2$

Hardware Profiling

SSD algorithm: $\sum(a_i - b_i)^2$

Instructions	Max. Freq. (Mhz)	Coverage
Addition/Substraction	160	54 %
Multiplication/Division	117	34 %
Memory Access	160	12 %

VASILE optimization - results

SSD algorithm: $\sum(a_i - b_i)^2$

VASILE optimization - results

SSD algorithm: $\sum(a_i - b_i)^2$

	Time (s)
ConnexArray	1.397
VASILE	1.186

VASILE optimization - results

SSD algorithm: $\sum(a_i - b_i)^2$

	Time (s)
ConnexArray	1.397
VASILE	1.186

speedup: 1.178

Summary

- fully functional ConnexArray compiler

Keywords: computer vision, ConnexArray, compiler, frequency scaling

Summary

- fully functional ConnexArray compiler
  - modular

Keywords: computer vision, ConnexArray, compiler, frequency scaling

Summary

- fully functional ConnexArray compiler
  - modular
  - easy to extend

Keywords: computer vision, ConnexArray, compiler, frequency scaling

Summary

- fully functional ConnexArray compiler
  - modular
  - easy to extend
  - without hard-constrains for a programming language

Keywords: computer vision, ConnexArray, compiler, frequency scaling

Summary

- fully functional ConnexArray compiler
  - modular
  - easy to extend
  - without hard-constrains for a programming language

- freqency scaling optimization (VASILE):

Keywords: computer vision, ConnexArray, compiler, frequency scaling

Summary

- fully functional ConnexArray compiler
  - modular
  - easy to extend
  - without hard-constrains for a programming language

- freqency scaling optimization (VASILE):
  - Speedup: 1.178

Keywords: computer vision, ConnexArray, compiler, frequency scaling