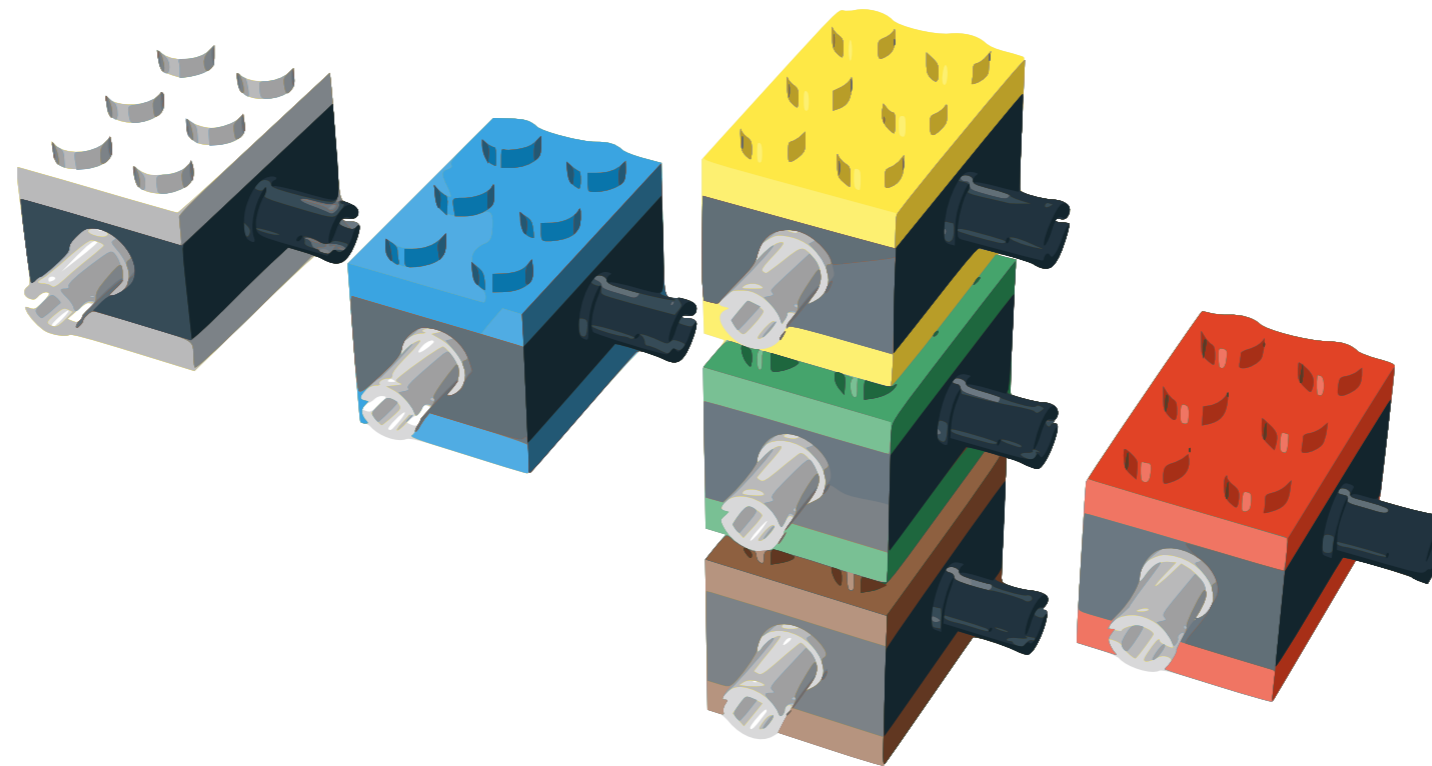


ρ -VEX

A Reconfigurable and Extensible VLIW Processor



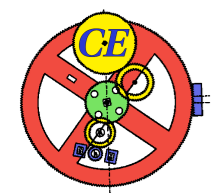
MSc Defense
Thijs van As

CE-MS-2008-12

<http://r-vex.googlecode.com/>

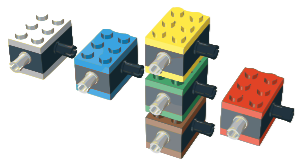
September 19, 2008

Thijs van As



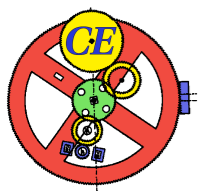
Computer Engineering Laboratory
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

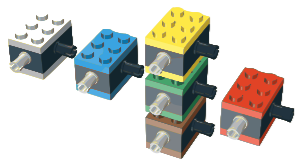
 **TU Delft**



Overview

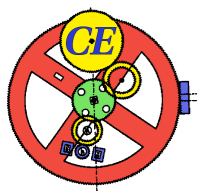
- 1.** Introduction
- 2.** Background
- 3.** Performance Analysis
- 4.** Design & Implementation
- 5.** Development Framework
- 6.** Experimental Results
- 7.** Conclusions

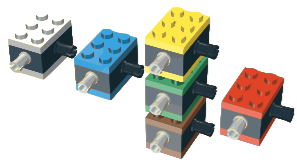




Overview

- 1.** Introduction
- 2.** Background
- 3.** Performance Analysis
- 4.** Design & Implementation
- 5.** Development Framework
- 6.** Experimental Results
- 7.** Conclusions

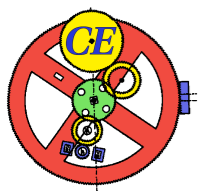


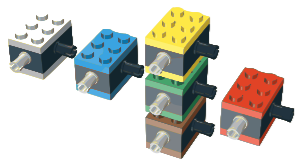


Introduction

Motivation and means

- Advances in reconfigurable hardware
 - MOLEN polymorphic processor
 - Very Long Instruction Word (VLIW)
-
- ➔ VLIW co-Processor for MOLEN
 - VEX ISA
 - Extensible
 - Availability of compiler + simulator
 - Stand-alone: VLIW research

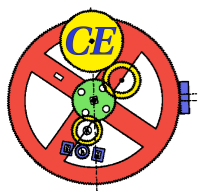


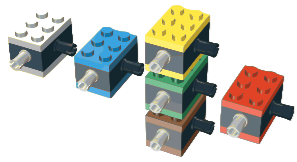


Introduction

Goals

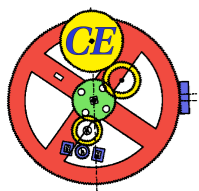
- Preliminary study: performance analysis
- ρ -VEX
 - Extensible ISA
 - Support for custom operations
- ρ -ASM

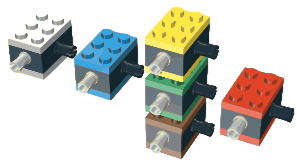




Overview

1. Introduction
2. Background
3. Performance Analysis
4. Design & Implementation
5. Development Framework
6. Experimental Results
7. Conclusions

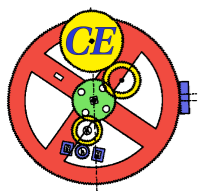


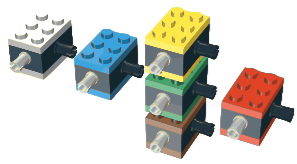


Background

Processor parallelism

- Vector processing
- Multiprocessing
- Multithreading
- Micro-SIMD
- Instruction Level Parallelism (ILP)
 - Multimedia applications expose a lot of ILP





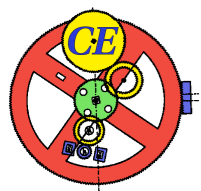
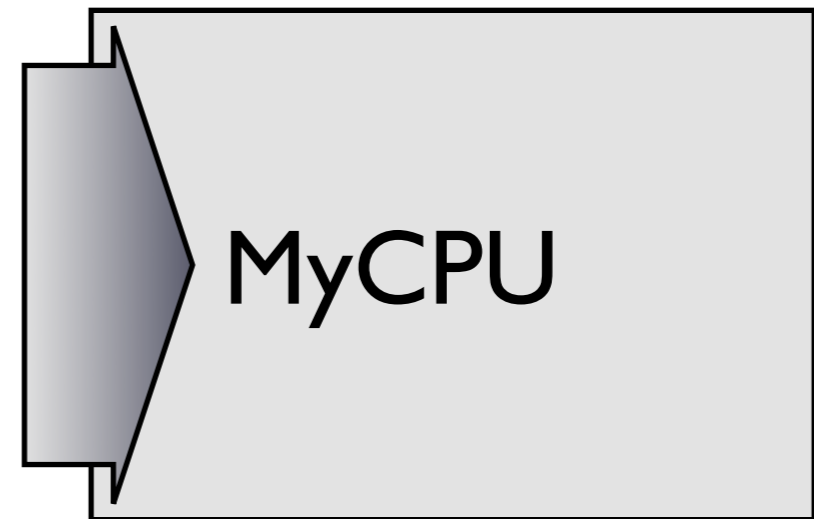
Background

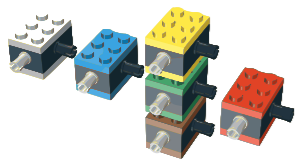
Processor parallelism

Instruction Level Parallelism (ILP)

MyProgram

3	d = y + z
2	c = w + x
1	b = u + v
0	a = s + t





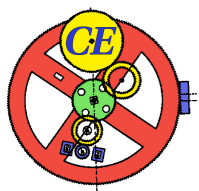
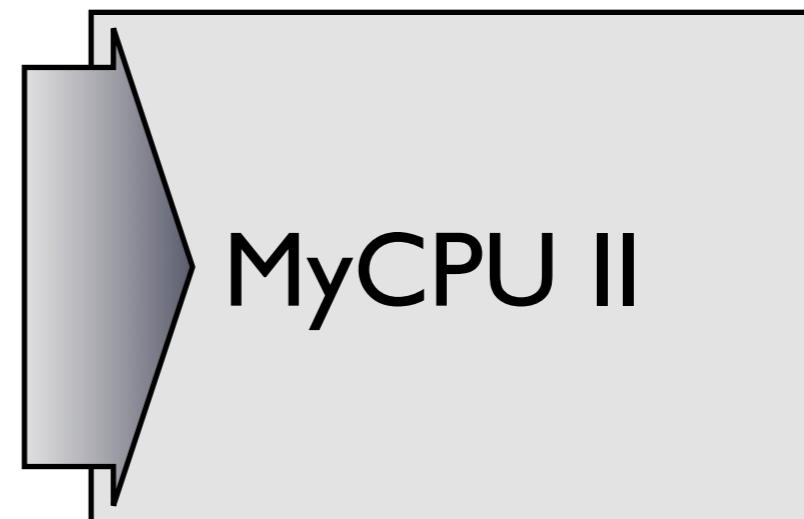
Background

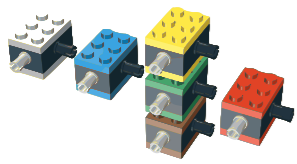
Processor parallelism

Instruction Level Parallelism (ILP)

MyProgram

3	d = y + z
2	c = w + x
1	b = u + v
0	a = s + t





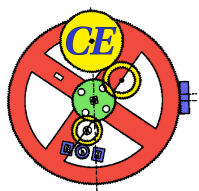
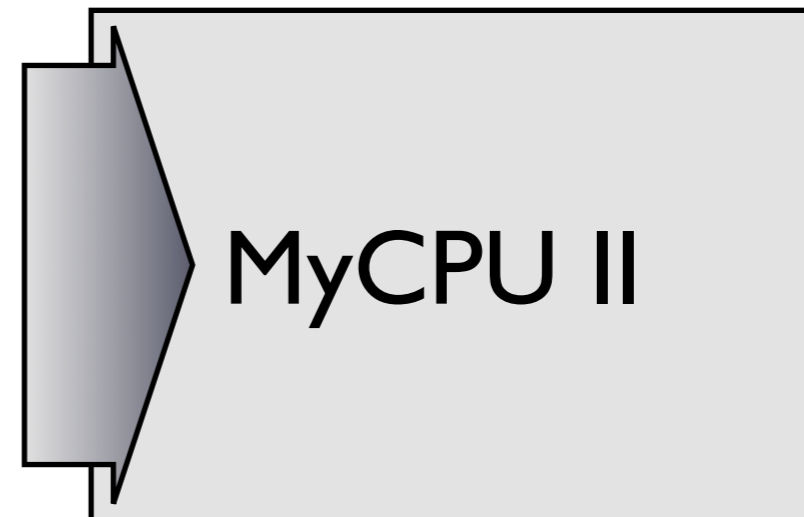
Background

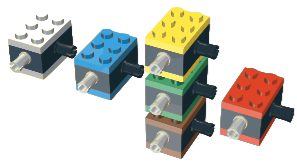
Processor parallelism

Instruction Level Parallelism (ILP)

MyProgram2

3	d = c + b
2	c = w + a
1	b = u + v
0	a = s + t



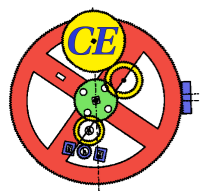


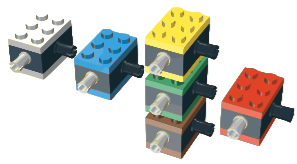
Background

Processor parallelism

Instruction Level Parallelism (ILP)

- Execute multiple different operations at once
- Two architectural solutions:
 - Very Long Instruction Word (VLIW)
 - ◆ Operation scheduling by compiler
 - Superscalar
 - ◆ Dynamic operation scheduling

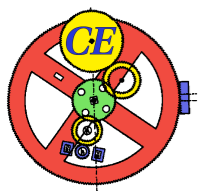


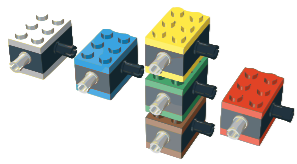


Background

Very Long Instruction Word (VLIW)

- Multi-operation instruction stream
- Compiler schedules operations
- Processor design transparent to compiler
- Industrial examples:
 - Intel Itanium (IA-64/EPIC)
 - Philips/NXP TriMedia (TMA)
 - Transmeta Crusoe (IA-32)

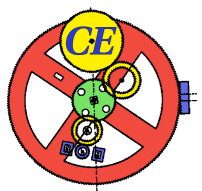


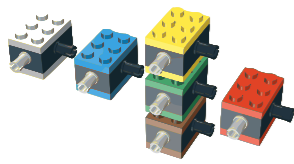


Background

The VEX ISA

- Lx (HP/ST) descendant
- Multi-cluster support
- Extensible ISA
- Custom operations
- Compiler + simulator toolchain by HP

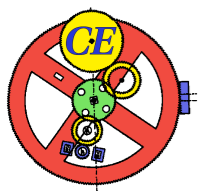


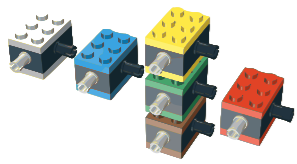


Background

The MOLEN polymorphic processor

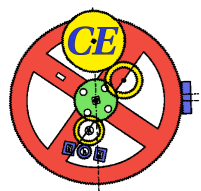
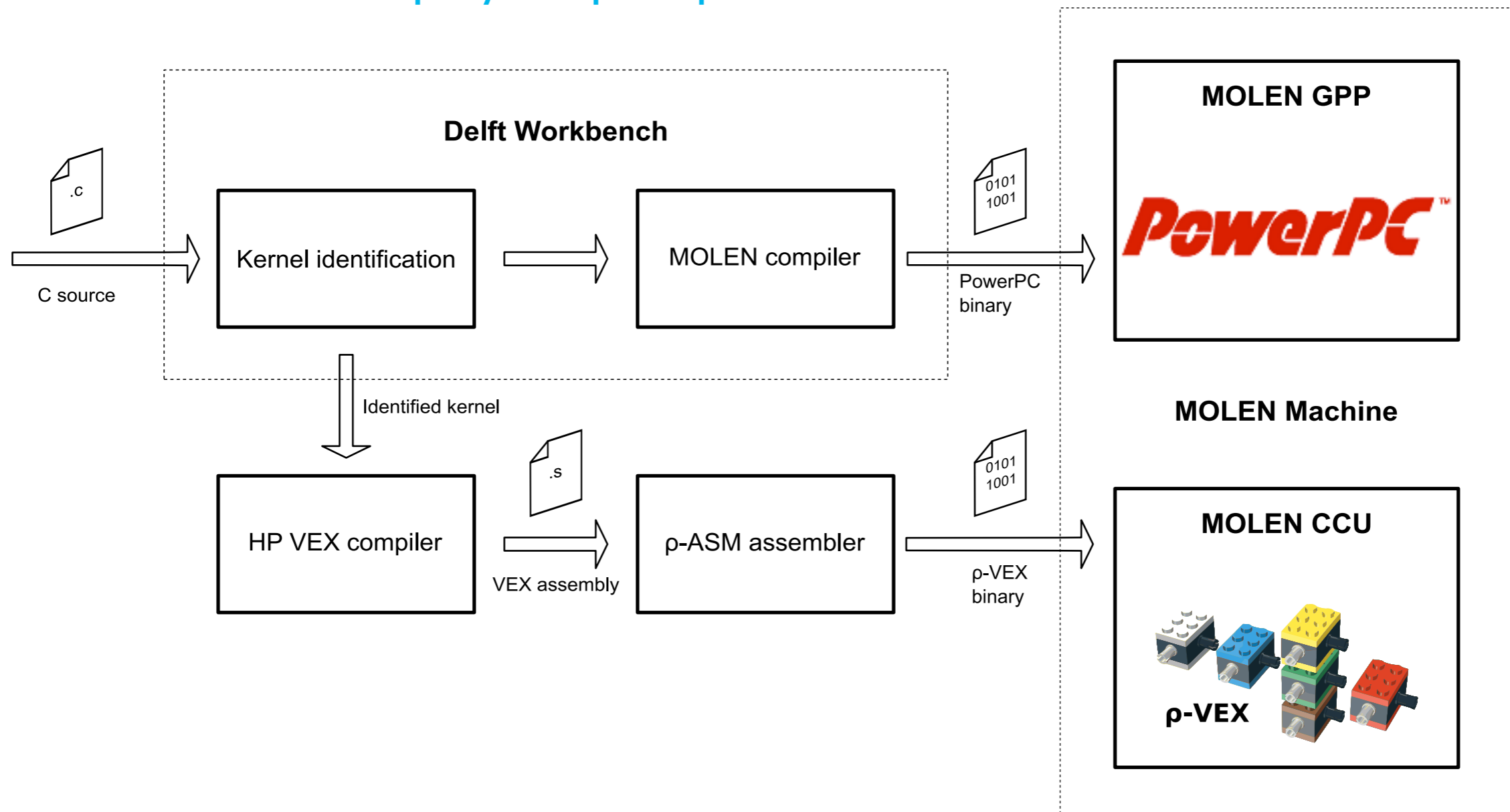
- Fixed GPP with reconfigurable processor
- CCUs as small application-specific computing elements
- Configuration by reconfigurable microcode ($\rho\mu$ -code)
- Only a few extra instructions
- Architecture-independent
- Delft Workbench workflow

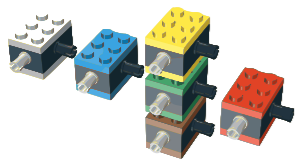




Background

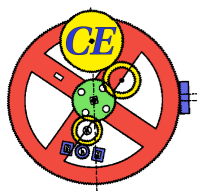
The MOLEN polymorphic processor

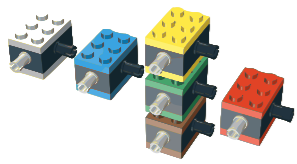




Overview

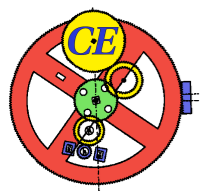
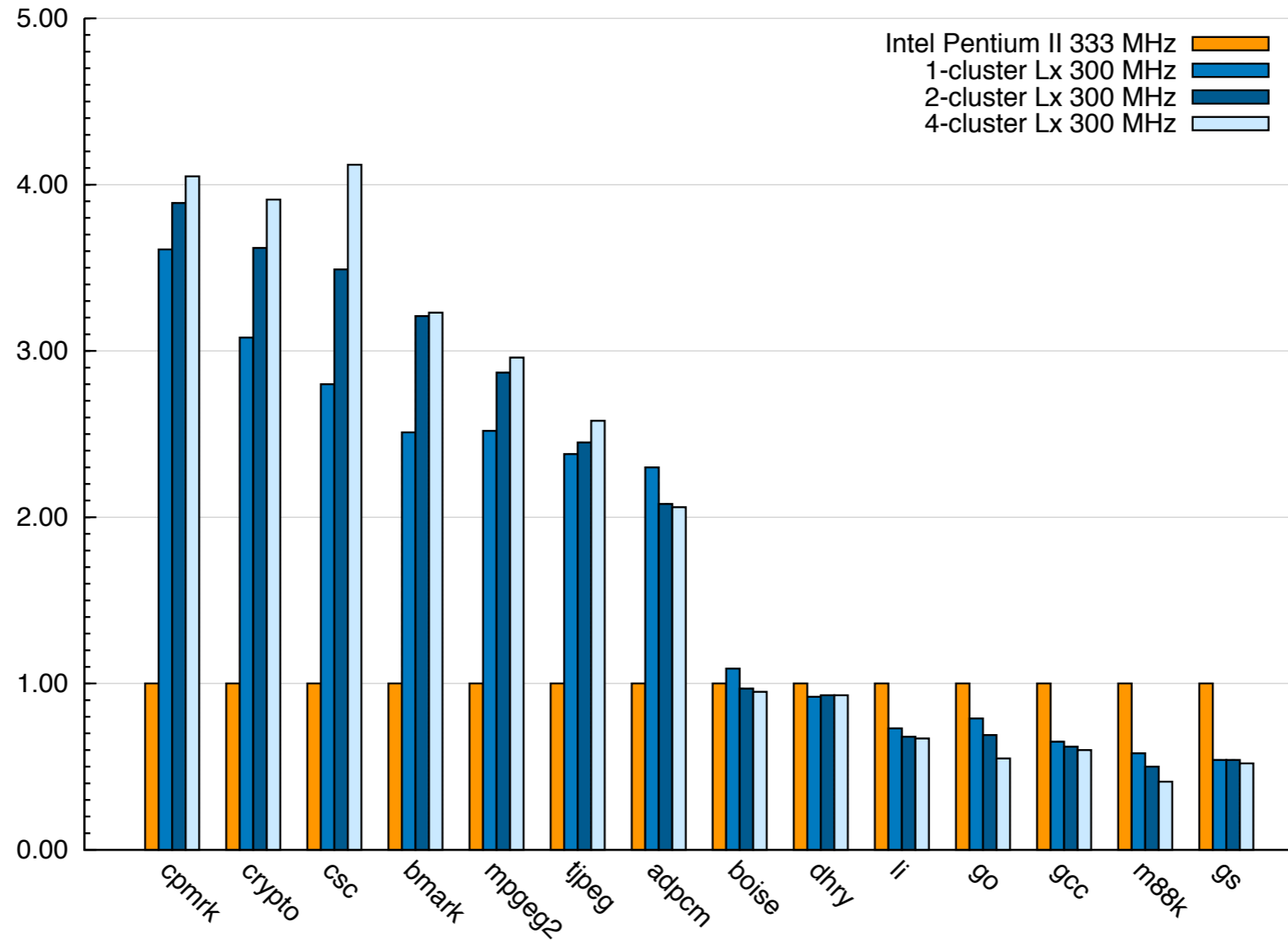
1. Introduction
2. Background
3. Performance Analysis
4. Design & Implementation
5. Development Framework
6. Experimental Results
7. Conclusions

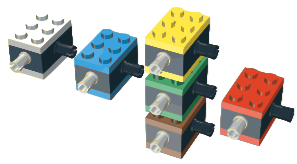




Performance Analysis

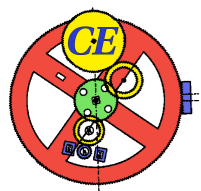
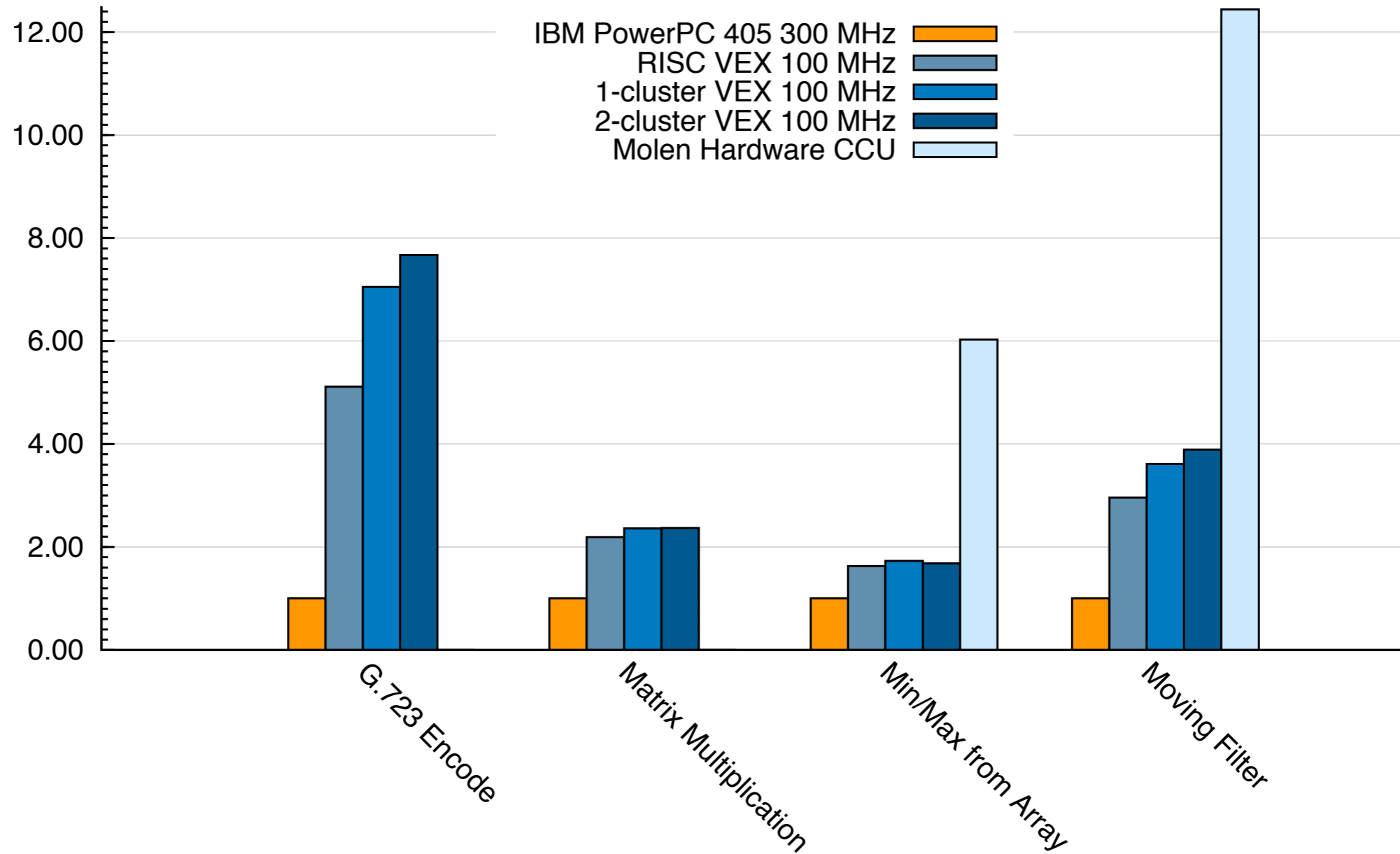
Lx analysis

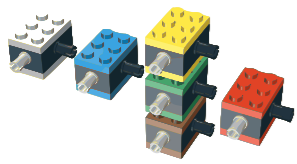




Performance Analysis

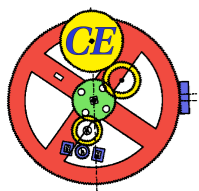
VEX analysis

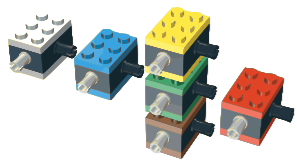




Overview

1. Introduction
2. Background
3. Performance Analysis
4. Design & Implementation
5. Development Framework
6. Experimental Results
7. Conclusions

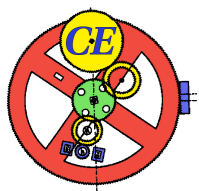




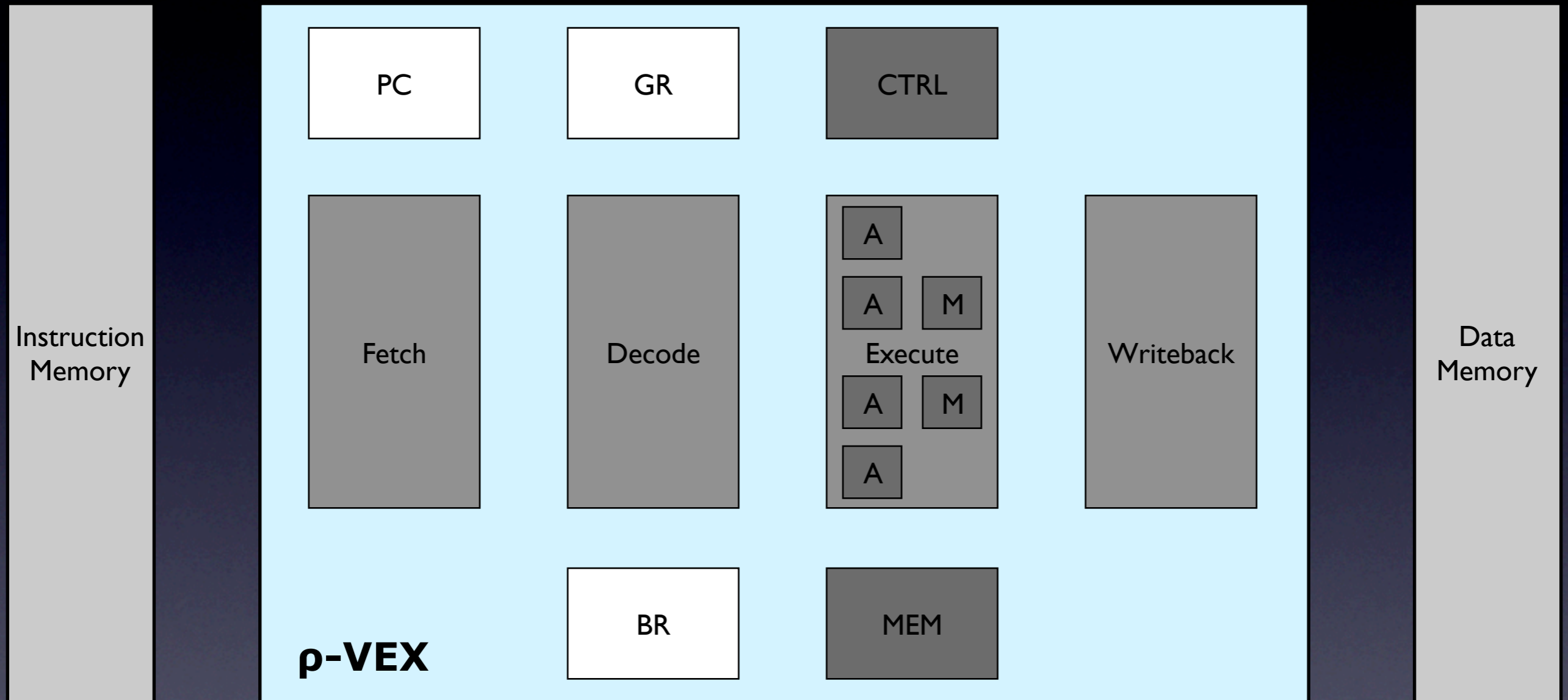
Design & Implementation

General approach

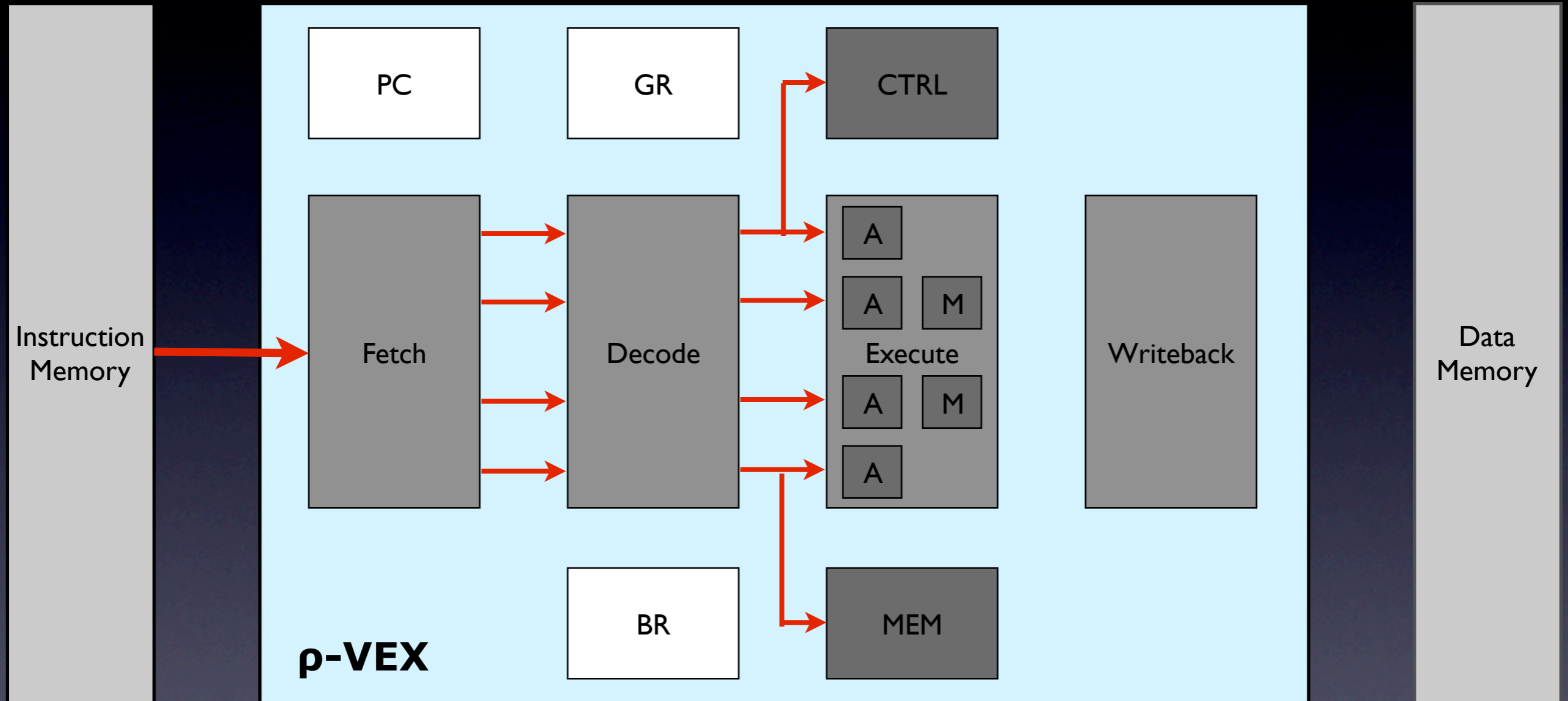
- Harvard architecture
- Designed custom instruction/syllable layout
- 73 standard VEX operations
- All ALU/MUL operations support immediate operands
- 32 bit syllables (128 bit instructions for 4-issue)
- First 1-issue, then 4-issue (standard cluster ISA)



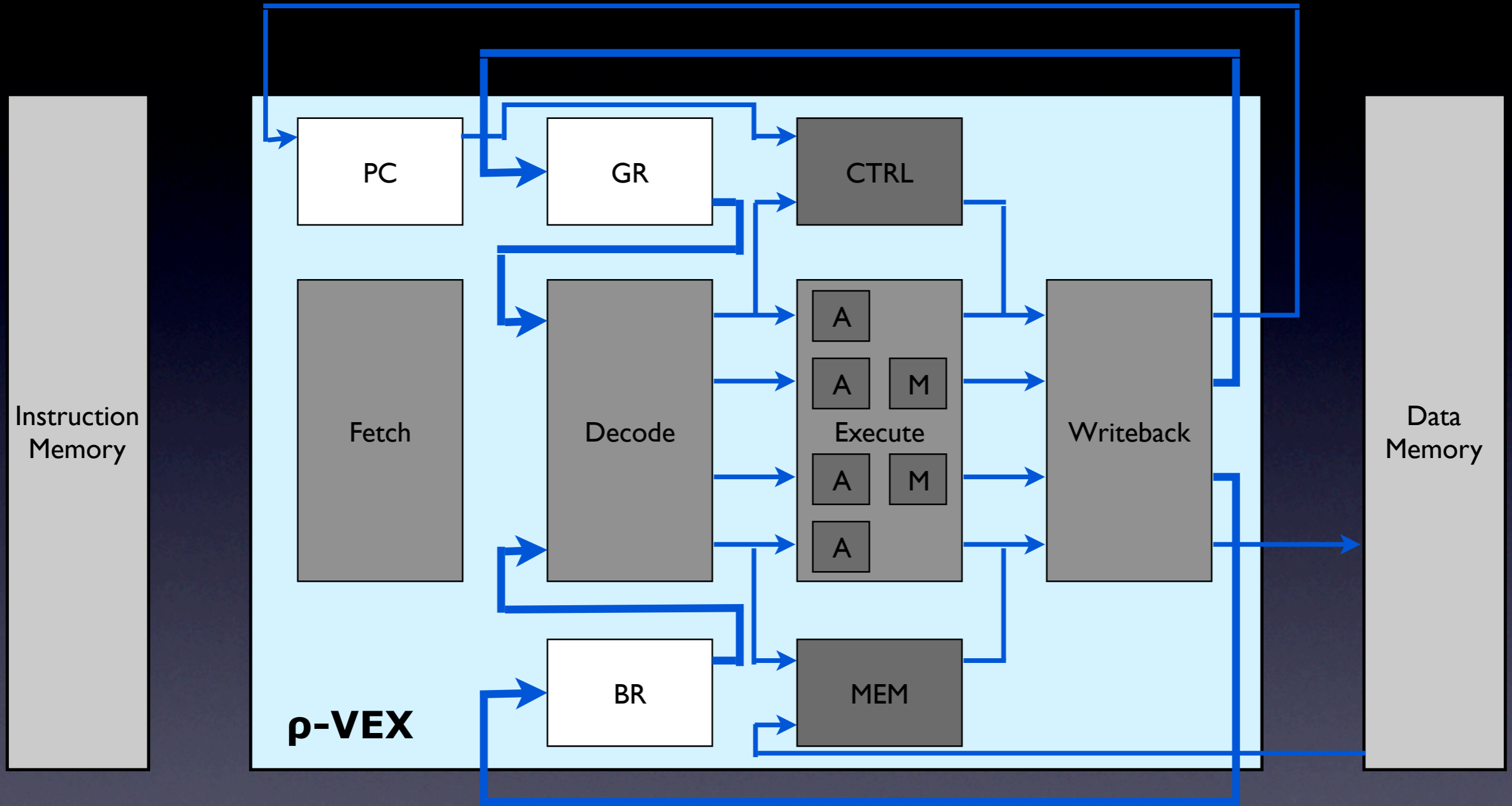
Organization



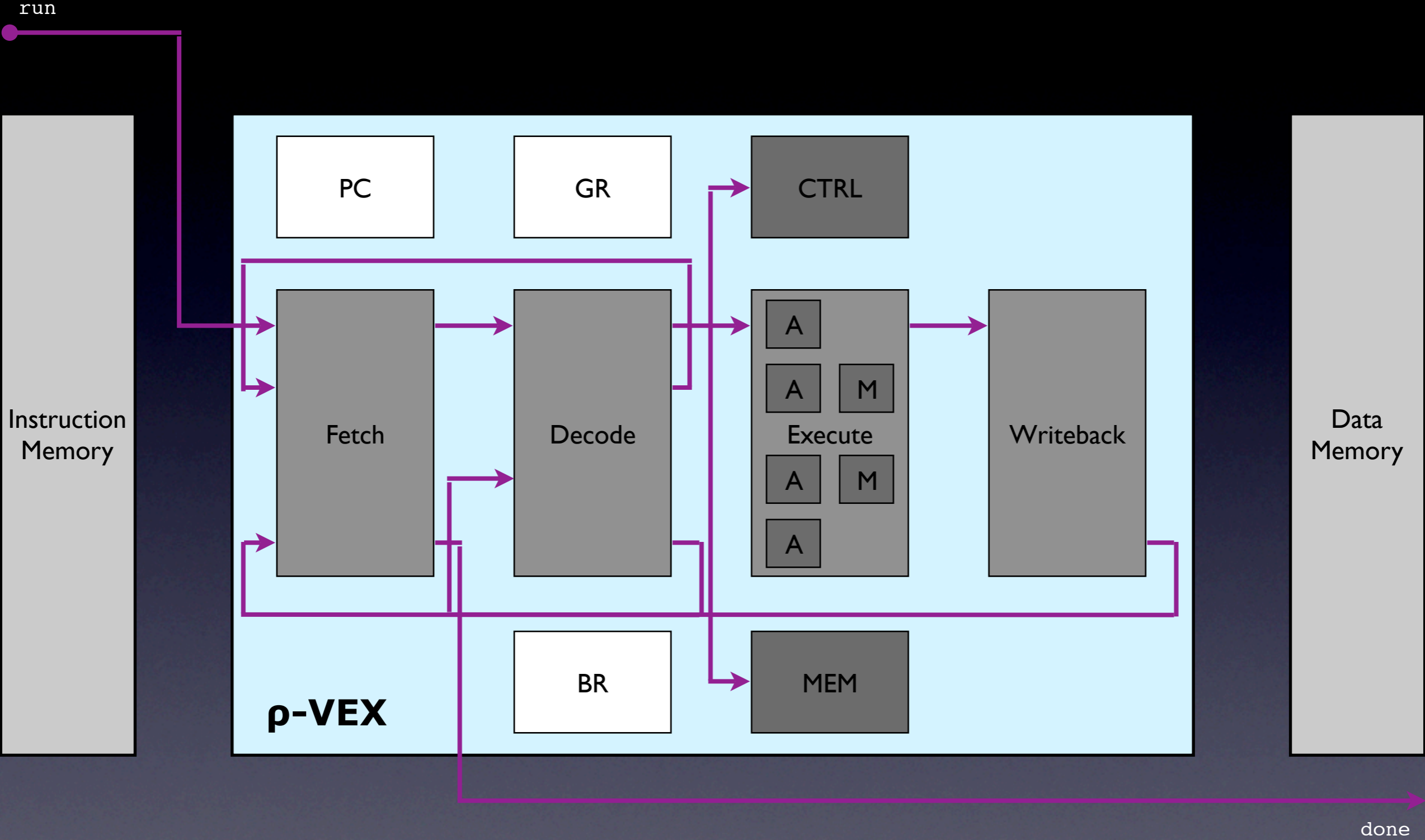
Instruction path



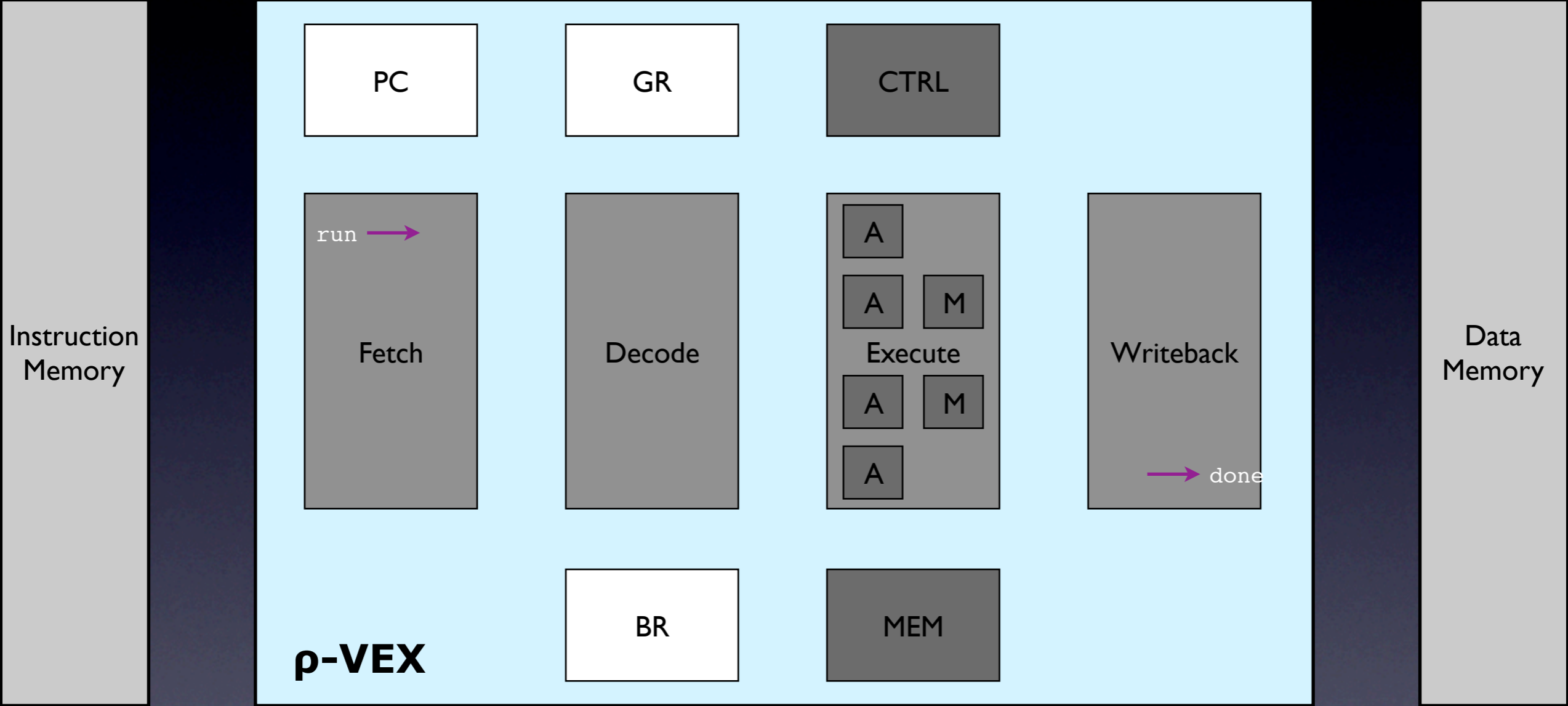
Data path

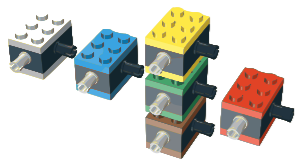


Control path



Control path



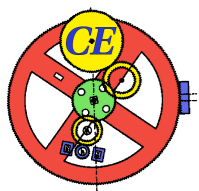


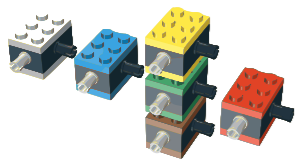
Design & Implementation

Extensibility

ρ -OPS

- Reconfigurable operations
- 24 available opcodes
- Easily extendable
- Not just combinatorial operators allowed



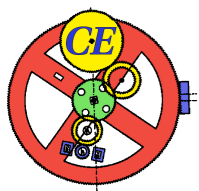


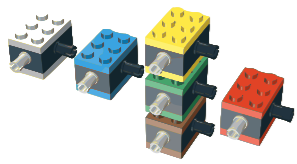
Design & Implementation

Extensibility

VEX machine models

- Issue-width
- Number of ALU units
- Number of MUL units
- Number of GR/BR registers



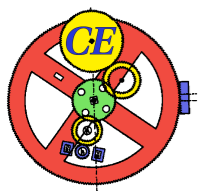


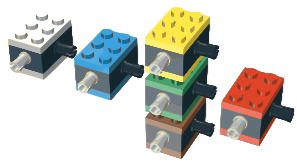
Design & Implementation

Verification

Unified Verification Methodology (UVM)

- System-level design
- Subsystem verification
- System integration
- System verification

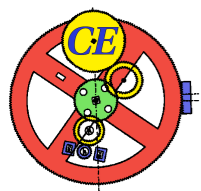
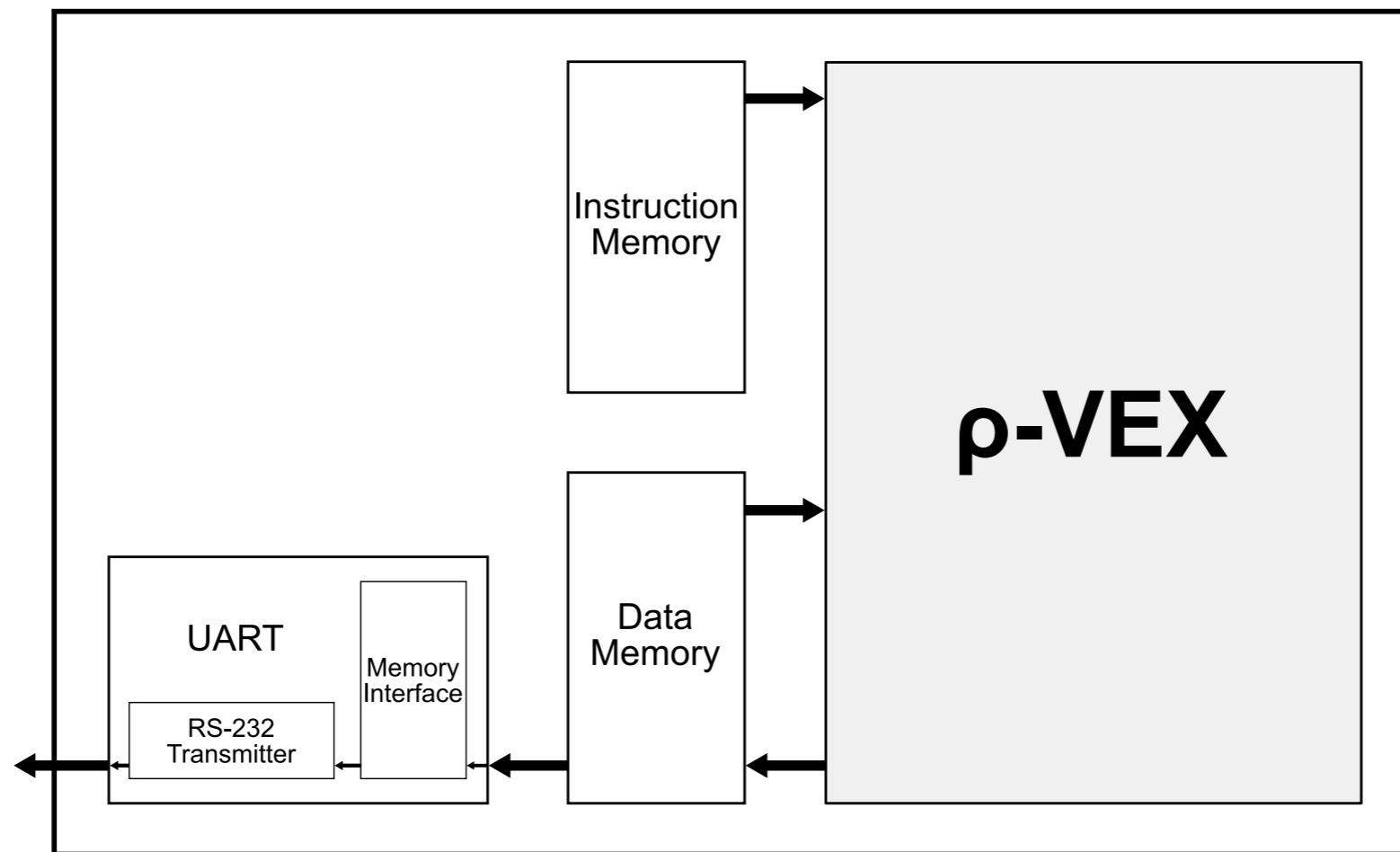


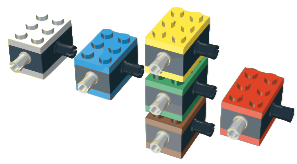


Design & Implementation

Verification

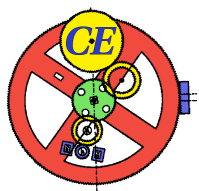
System wrapper

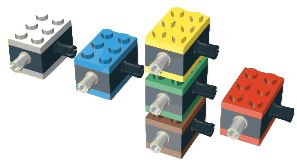




Overview

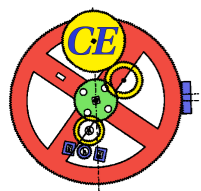
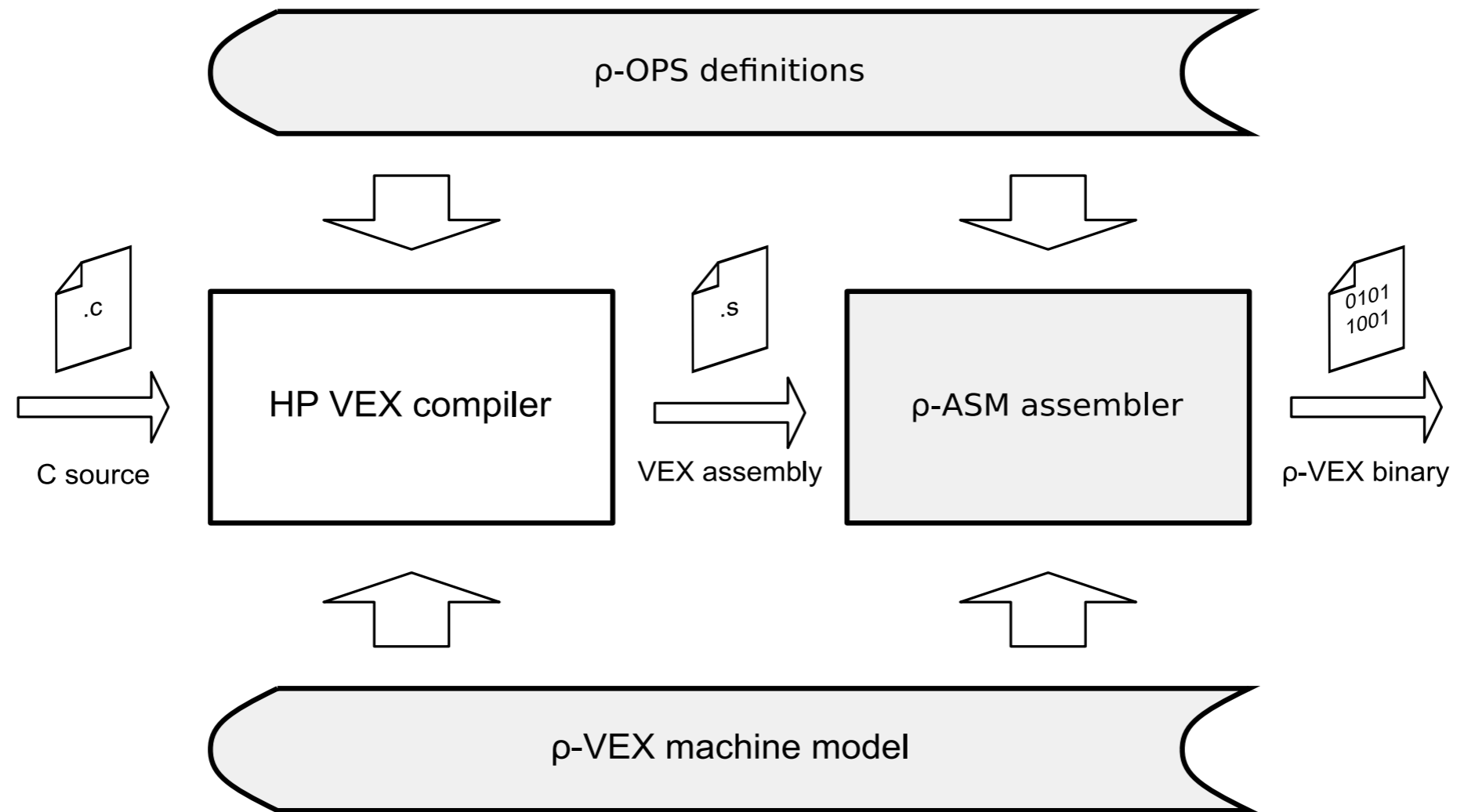
1. Introduction
2. Background
3. Performance Analysis
4. Design & Implementation
5. Development Framework
6. Experimental Results
7. Conclusions

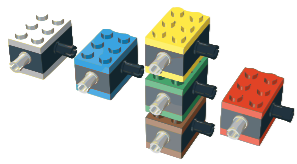




Development Framework

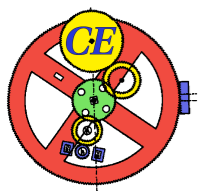
Development framework

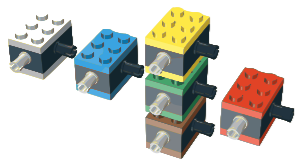




Overview

1. Introduction
2. Background
3. Performance Analysis
4. Design & Implementation
5. Development Framework
6. Experimental Results
7. Conclusions

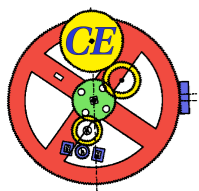


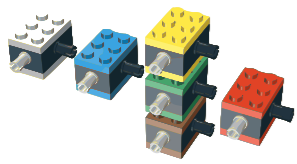


Experimental Results

Fibonacci's Sequence

- Calculates 45th Fibonacci Number
- Hand-coded VEX assembly
- 1-, 2- and 4-issue versions
- ρ -OPS exploiting version

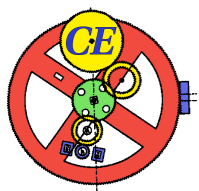


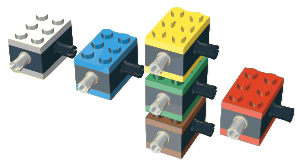


Experimental Results

Fibonacci's Sequence

ρ -VEX	Cycles
1-issue	1906
2-issue	1080
4-issue	537
4-issue (ρ -OPS)	141

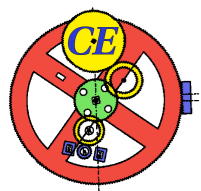


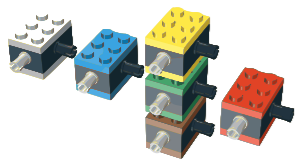


Experimental Results

Resource utilization on XC2VP30

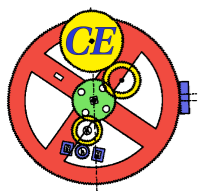
ρ -VEX	Freq. (MHz)	Slices	Slices GR
1-issue	89.44	1895 (13%)	1 (0%)
2-issue	89.44	5105 (37%)	3370 (24%)
4-issue	89.44	10433 (76%)	3927 (28%)

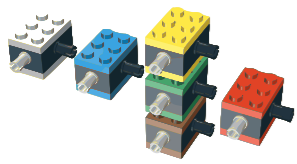




Overview

1. Introduction
2. Background
3. Performance Analysis
4. Design & Implementation
5. Development Framework
6. Experimental Results
7. Conclusions

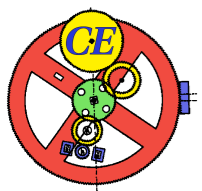


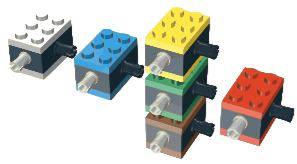


Conclusions

Performed work (chronologically)

- Performance/pay-off analysis for MOLEN
- Designed custom instruction layout for ρ -VEX
- 1-issue ρ -VEX hardware implementation
- n-issue ρ -VEX hardware implementation
- Paper (accepted on ICFPT 2008 in Taiwan)
- ρ -ASM VEX assembler/instruction ROM generator
- Published project as open source on Google Code

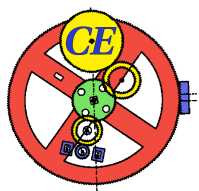


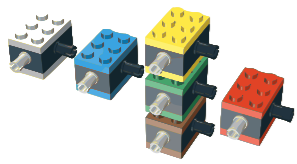


Conclusions

Main contributions

- ρ -VEX
 - Extensible and scalable ISA
 - Support for reconfigurable operations (ρ -OPS)
- ρ -ASM
- Application development framework
- Performance/pay-off analysis for MOLEN

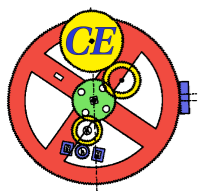


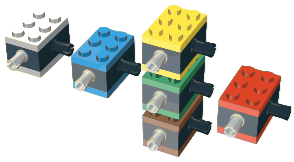


Conclusions

Future work

- ρ -VEX
 - Pipelining the implementation
 - Wishbone bus connectivity
- ρ -ASM
 - BRAM initialization
- Application development framework
 - Boot loader functionality
- MOLEN
 - Resolve CCU timing constraints





Questions

Thank you for your attendance!

r-vex - Google Code

http://code.google.com/p/r-vex/

t.vanas@gmail.com | What's new? | Profile | Settings | Help | Sign out

Google Code **r-vex**
p-VEX: A Reconfigurable and Extensible VLIW Processor

Project Home Downloads Wiki Issues Source Administer

p-VEX

p-VEX: A Reconfigurable and Extensible VLIW Processor

This is the project website for p-VEX, an open source VLIW processor with an accompanied development framework. The project started as the MSc project of Thijs van As.

About p-VEX

p-VEX is an open source reconfigurable and extensible Very-Long Instruction Word (VLIW) processor, accompanied by a development framework consisting of a VEX assembler, p-ASM. The processor architecture is based on the VEX ISA, as introduced by [J.A. Fisher et al.](#). The VEX ISA offers a scalable technology platform for embedded VLIW processors, that allows variation in many aspects, including instruction issue-width, organization of functional units, and instruction set. The p-VEX source code is described in VHDL. p-ASM is written in C.

A software development compiler toolchain for VEX is made publicly available by [Hewlett-Packard](#). The reasons VEX was chosen as the ISA are merely its extensibility and the quality of the available compiler. The design provides mechanisms that allow parametric extensibility of p-VEX. Both reconfigurable operations, as well as the versatility of VEX machine models are supported by p-VEX. The processor and framework are targeted at VLIW prototyping research and embedded processor design.

Code License: [GNU General Public License v3](#)

Labels: [r-VEX](#), [p-VEX](#), [rho-VEX](#), [VEX](#), [FPGA](#), [VHDL](#), [VLIW](#), [microprocessor](#), [processor](#), [softcore](#), [assembler](#), [r-ASM](#), [p-ASM](#), [rho-ASM](#), [p-VEX](#)

Featured Downloads: [r-vex_r38_MSc.tgz](#), [thesis_tvanas.pdf](#)

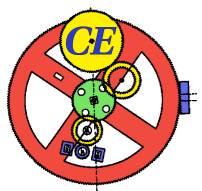
Featured Wiki Pages: [OperationsSemantics](#), [QuickstartGuide](#)

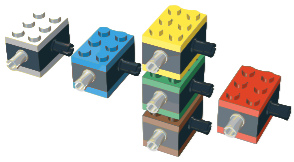
Links: [VEX Toolchain by HP](#), [VEX Website & Forum](#), [TU Delft | Computer Engineering Laboratory](#), [pretopia.net](#)

Blogs: [blog.pretopia.net](#)

Project owners: [t.vanas](#)

<http://r-vex.googlecode.com/>





Questions

Thank you for your attendance!

r-vex - Google Code

http://code.google.com/p/r-vex/

t.vanas@gmail.com | What's new? | Profile | Settings | Help | Sign out

Google Code **r-vex**
p-VEX: A Reconfigurable and Extensible VLIW Processor

Project Home Downloads Wiki Issues Source Administer

p-VEX

p-VEX: A Reconfigurable and Extensible VLIW Processor

This is the project website for p-VEX, an open source VLIW processor with an accompanied development framework. The project started as the MSc project of Thijs van As.

About p-VEX

p-VEX is an open source reconfigurable and extensible Very-Long Instruction Word (VLIW) processor, accompanied by a development framework consisting of a VEX assembler, p-ASM. The processor architecture is based on the VEX ISA, as introduced by [J.A. Fisher et al.](#). The VEX ISA offers a scalable technology platform for embedded VLIW processors, that allows variation in many aspects, including instruction issue-width, organization of functional units, and instruction set. The p-VEX source code is described in VHDL. p-ASM is written in C.

A software development compiler toolchain for VEX is made publicly available by [Hewlett-Packard](#). The reasons VEX was chosen as the ISA are merely its extensibility and the quality of the available compiler. The design provides mechanisms that allow parametric extensibility of p-VEX. Both reconfigurable operations, as well as the versatility of VEX machine models are supported by p-VEX. The processor and framework are targeted at VLIW prototyping research and embedded processor design.

Code License: [GNU General Public License v3](#)

Labels: [r-VEX](#), [p-VEX](#), [rho-VEX](#), [VEX](#), [FPGA](#), [VHDL](#), [VLIW](#), [microprocessor](#), [processor](#), [softcore](#), [assembler](#), [r-ASM](#), [p-ASM](#), [rho-ASM](#), [p-VEX](#)

Featured Downloads: [r-vex_r38_MSc.tgz](#), [thesis_tvanas.pdf](#)

Featured Wiki Pages: [OperationsSemantics](#), [QuickstartGuide](#)

Links: [VEX Toolchain by HP](#), [VEX Website & Forum](#), [TU Delft | Computer Engineering Laboratory](#), [pretopia.net](#)

Blogs: [blog.pretopia.net](#)

Project owners: [t.vanas](#)

<http://r-vex.googlecode.com/>

