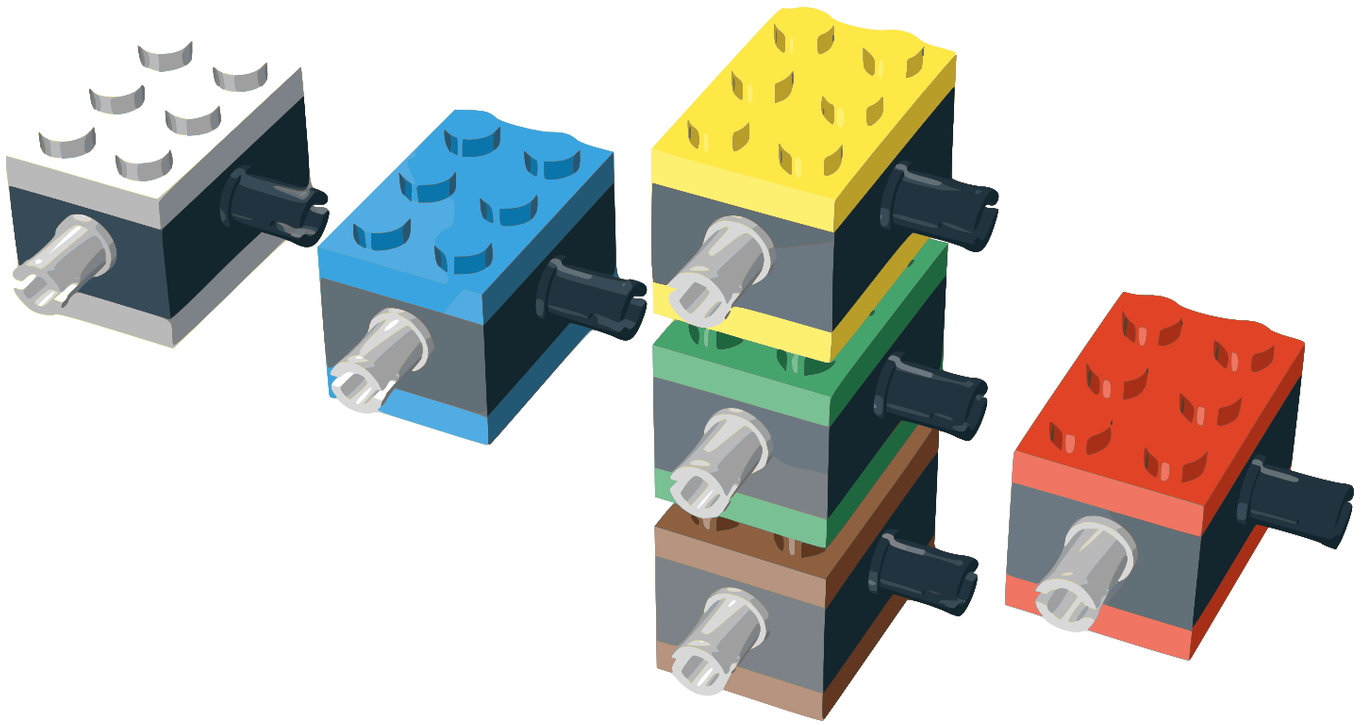
ρ-VEX

A Reconfigurable and Extensible VLIW Processor



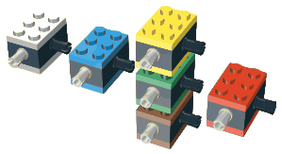
MSc Defense

Thijs van As

CE-MS-2008-12

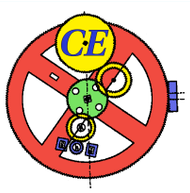
http://r-vex.googlecode.com/

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

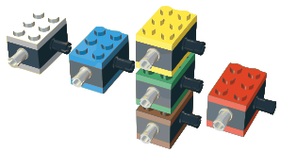


Overview

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

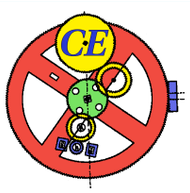


TU Delft

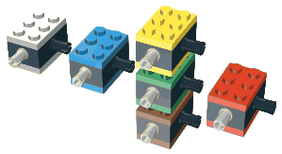


Overview

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



TU Delft



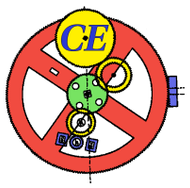
Introduction

Motivation and means

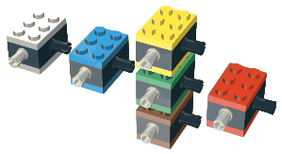
- Advances in reconfigurable hardware
- MOLEN polymorphic processor
- Very Long Instruction Word (VLIW)

➡ VLIW co-Processor for MOLEN

- VEX ISA
  - Extensible
  - Availability of compiler + simulator
- Stand-alone: VLIW research



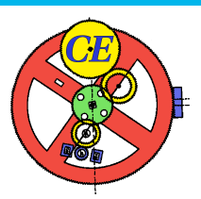
TU Delft



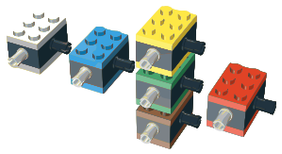
Introduction

Goals

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

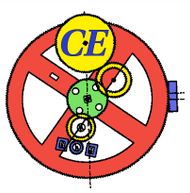


TU Delft

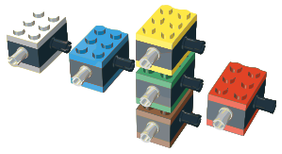


Overview

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



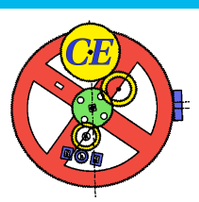
TU Delft



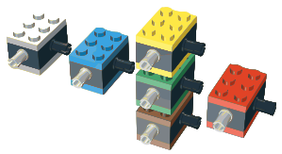
Background

Processor parallelism

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



TU Delft



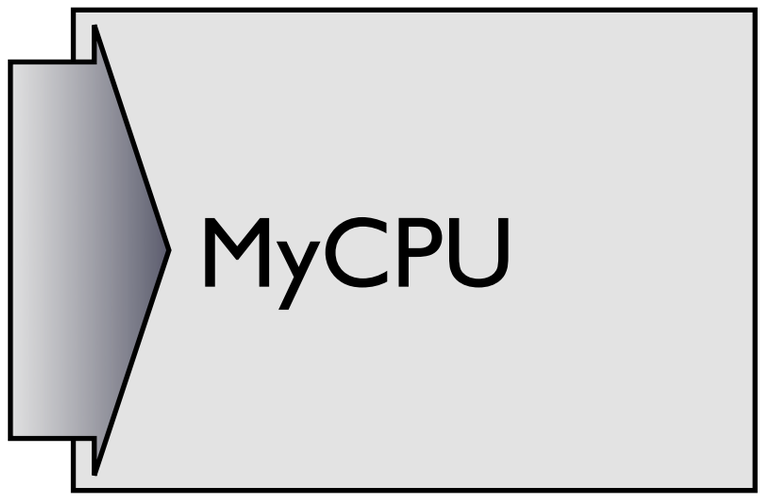
Background

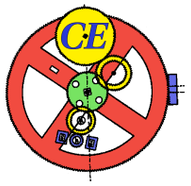
Processor parallelism

Instruction Level Parallelism (ILP)

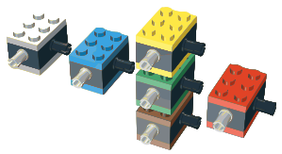
MyProgram

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





TU Delft



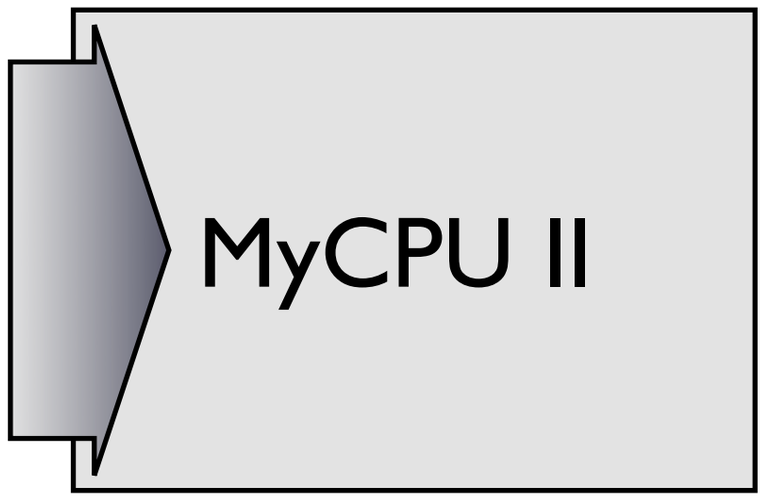
Background

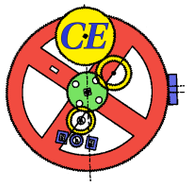
Processor parallelism

Instruction Level Parallelism (ILP)

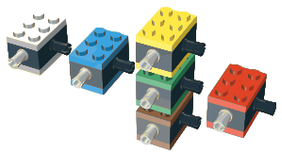
MyProgram

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





TU Delft



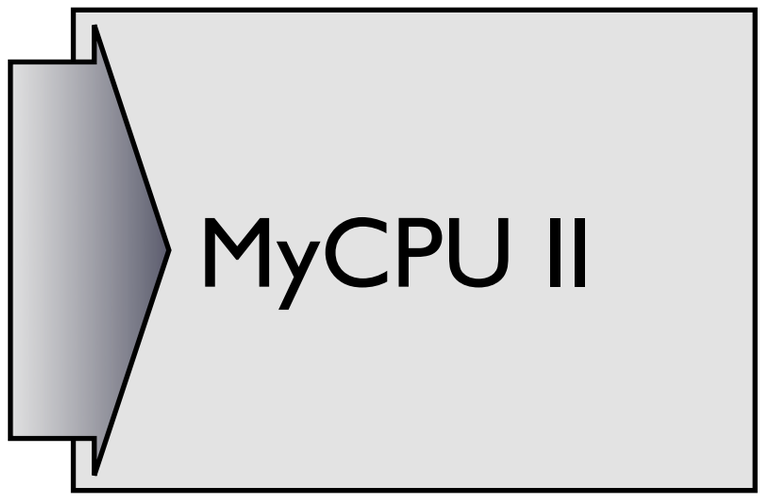
Background

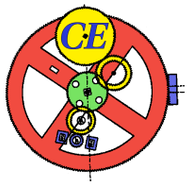
Processor parallelism

Instruction Level Parallelism (ILP)

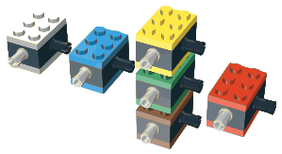
MyProgram2

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





TU Delft

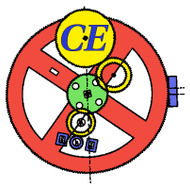


Background

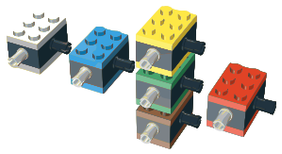
Processor parallelism

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



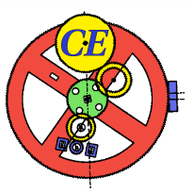
TU Delft



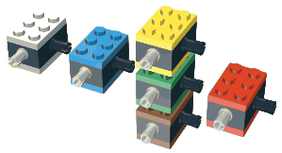
Background

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)



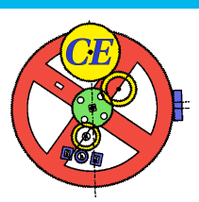
TU Delft



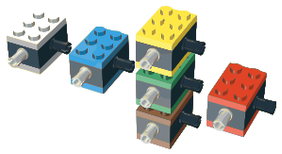
Background

The VEX ISA

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



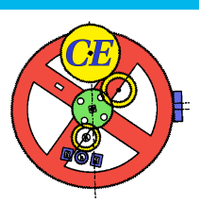
TU Delft



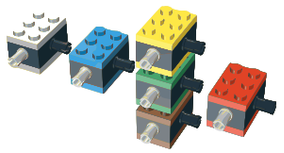
Background

The MOLEN polymorphic processor

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

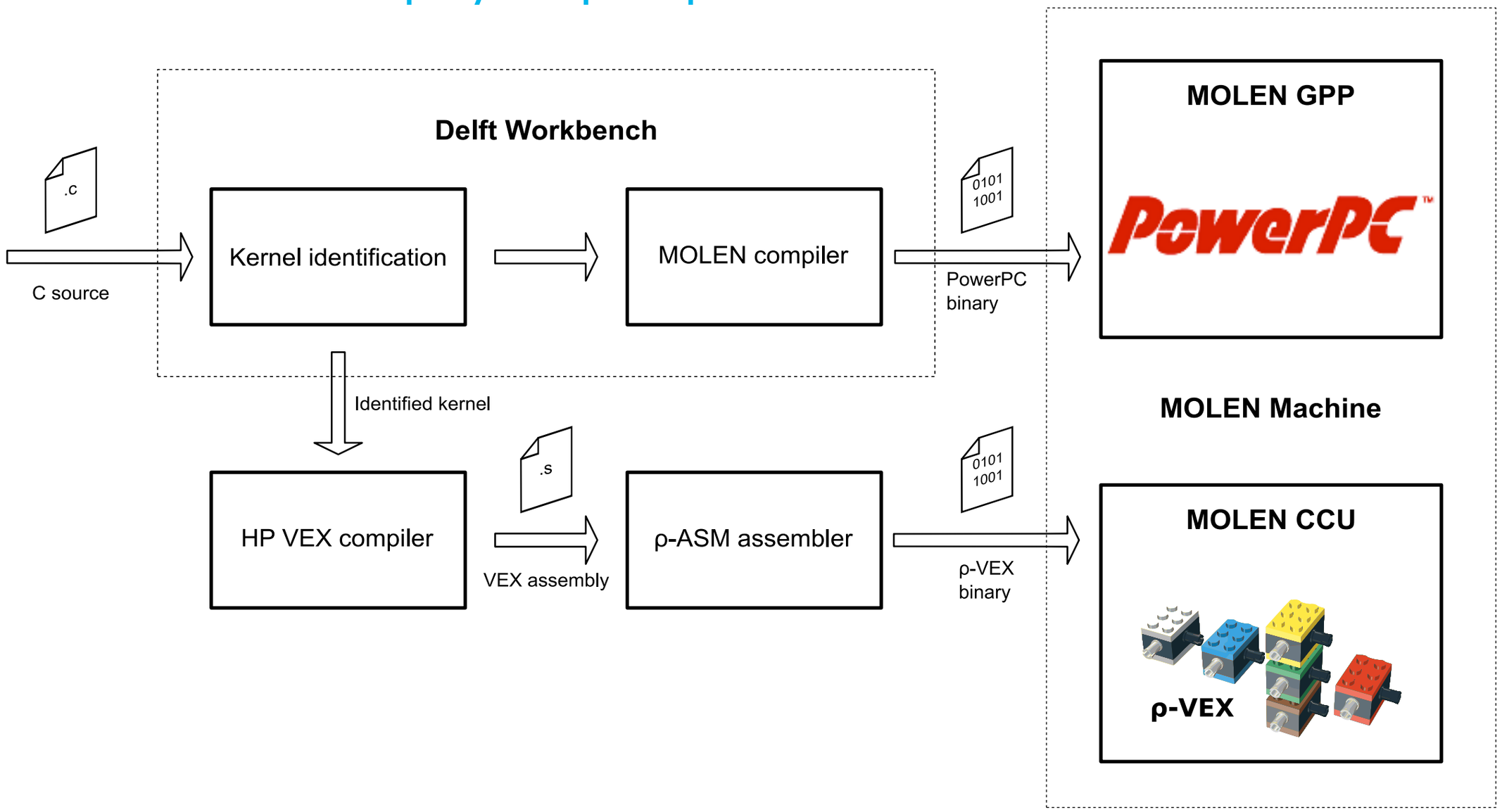


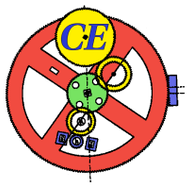
TU Delft



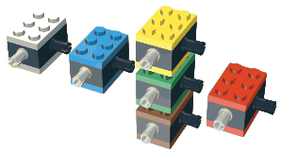
Background

The MOLEN polymorphic processor



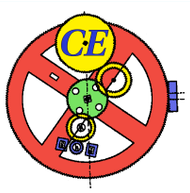


TU Delft

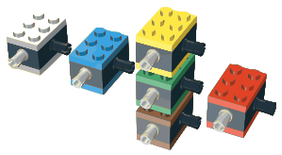


Overview

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

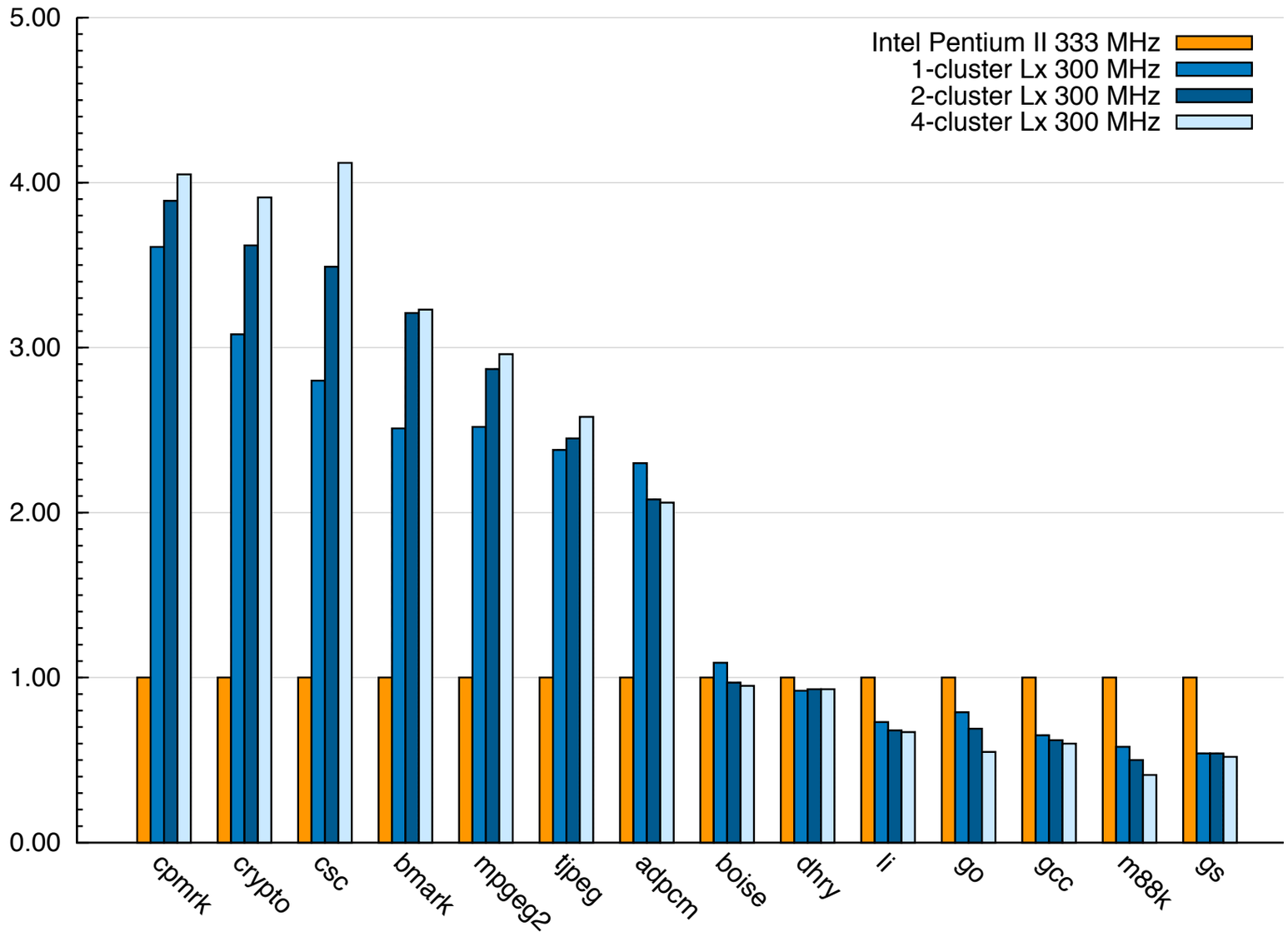


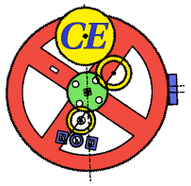
TU Delft



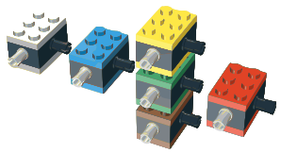
Performance Analysis

Lx analysis



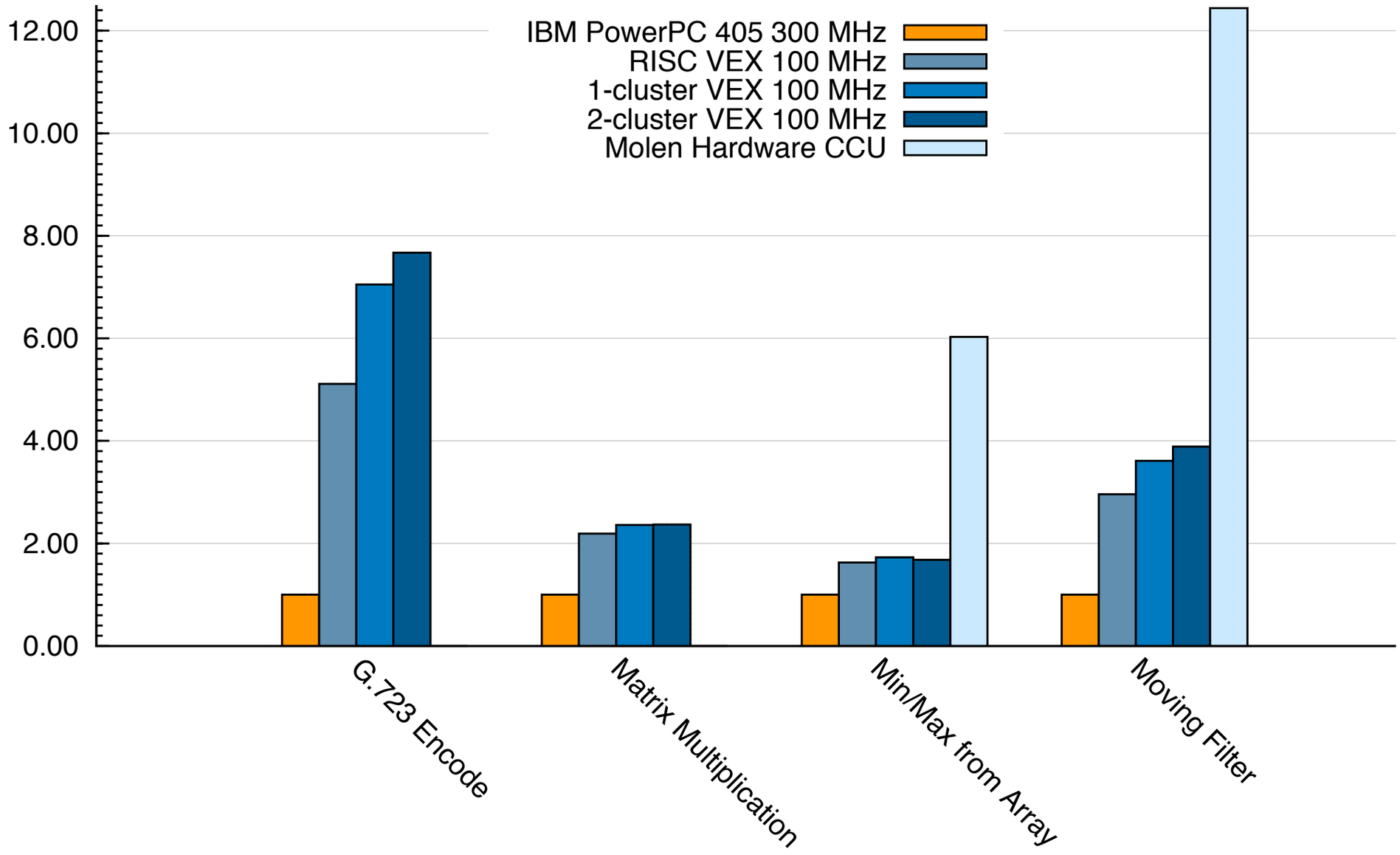


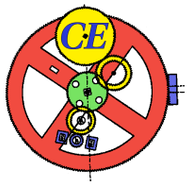
TU Delft



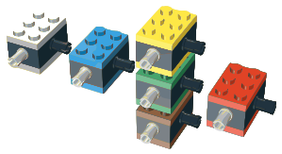
Performance Analysis

VEX analysis



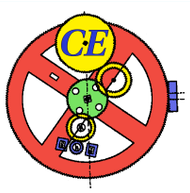


TU Delft

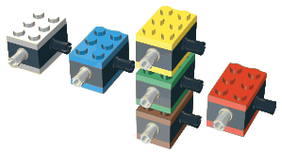


Overview

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



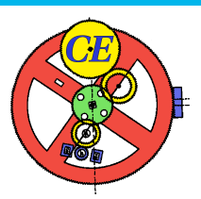
TU Delft



Design & Implementation

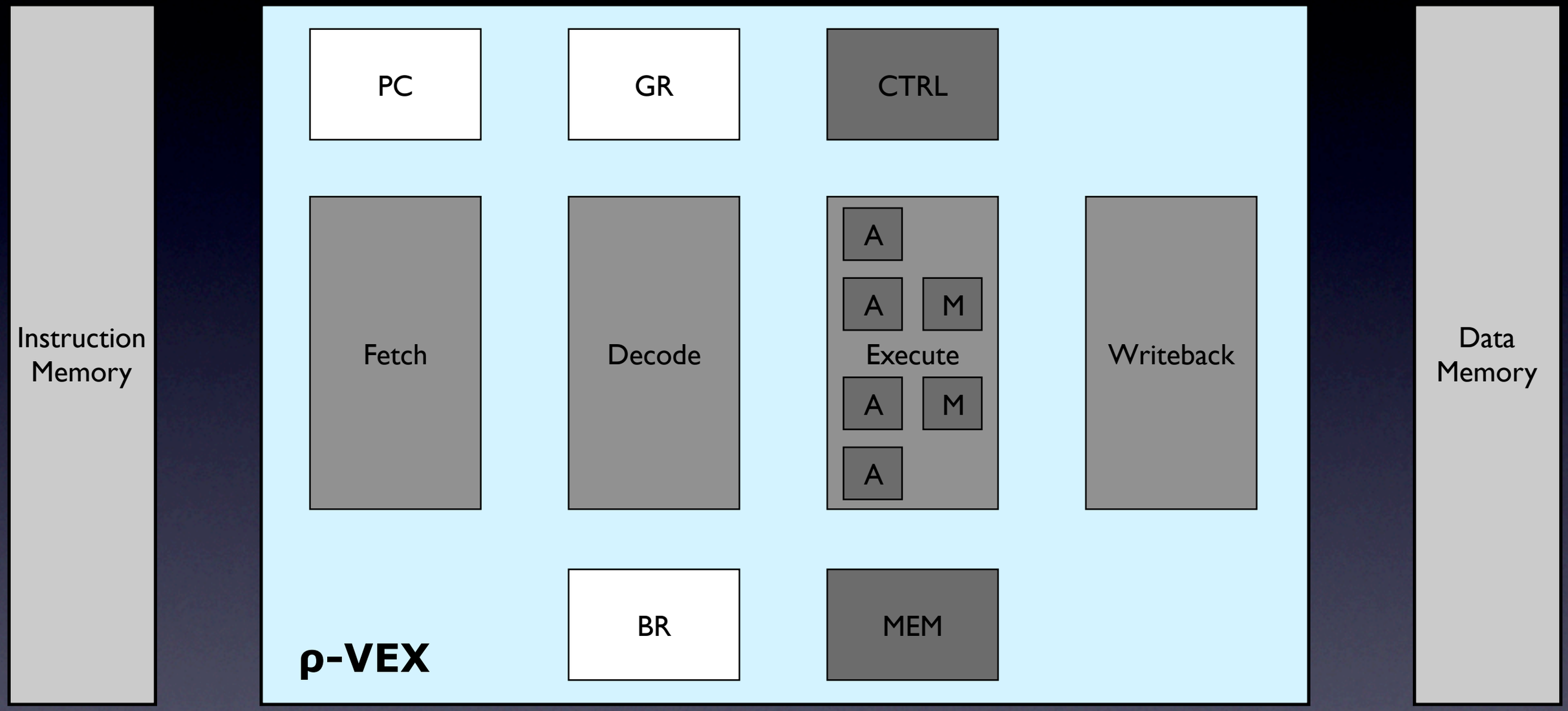
General approach

- 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)

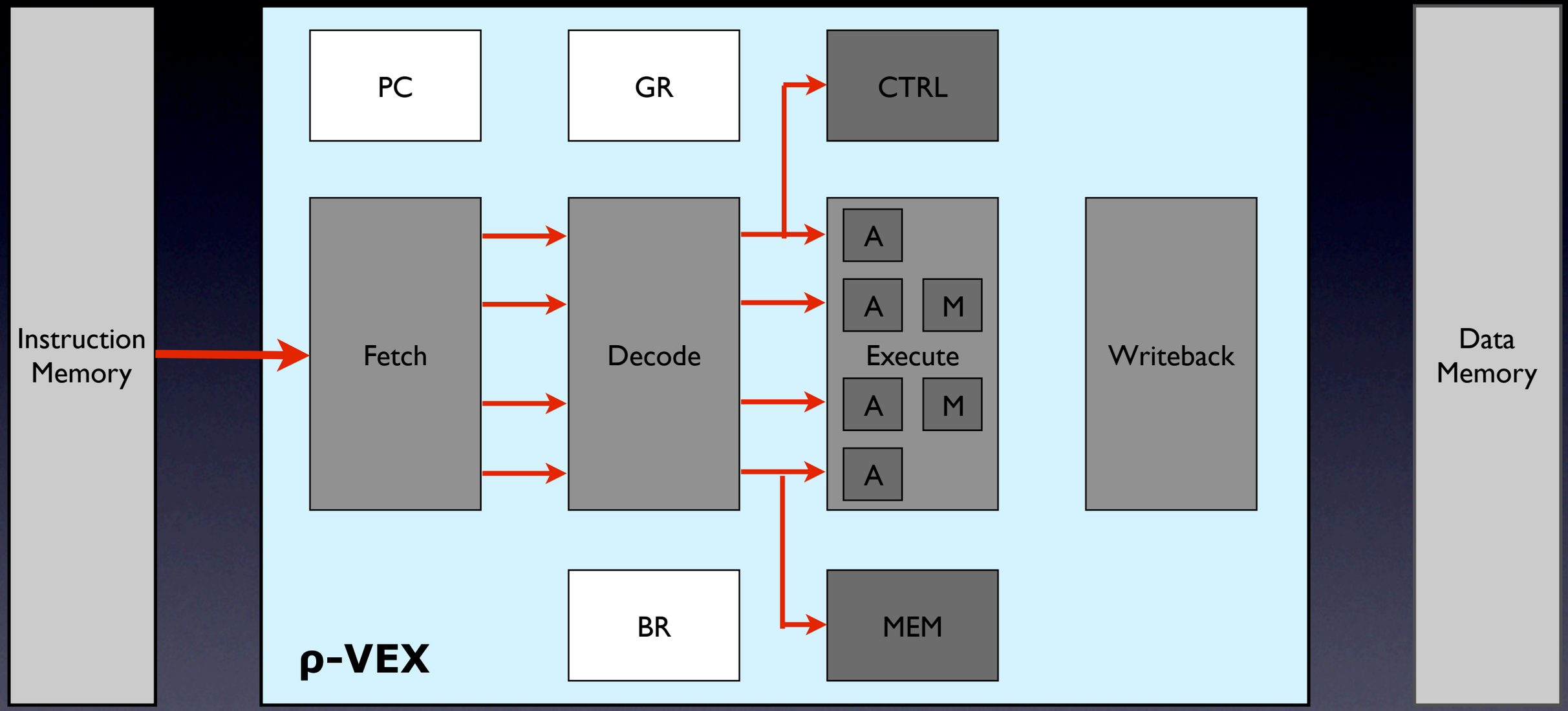


TU Delft

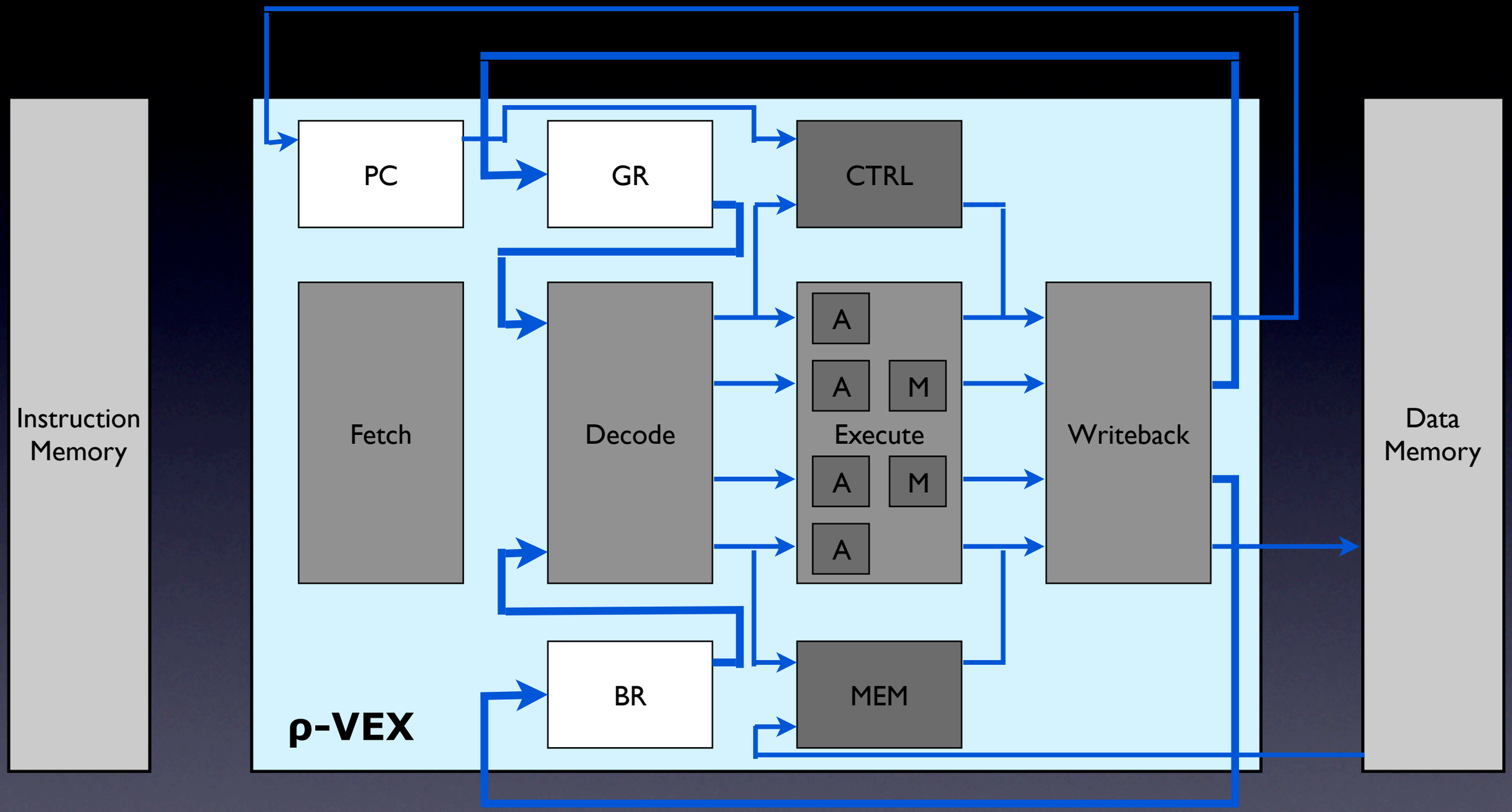
Organization



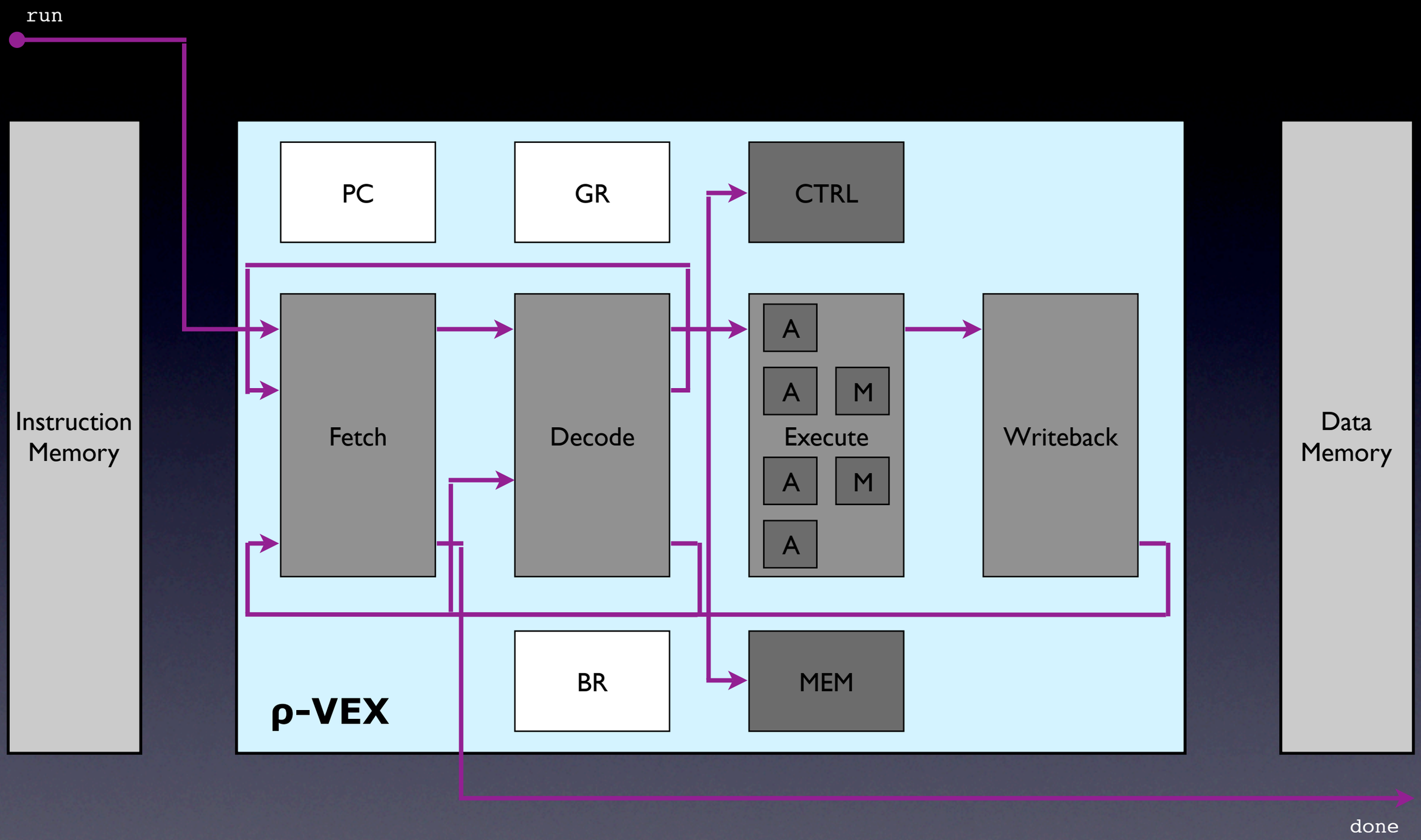
Instruction path



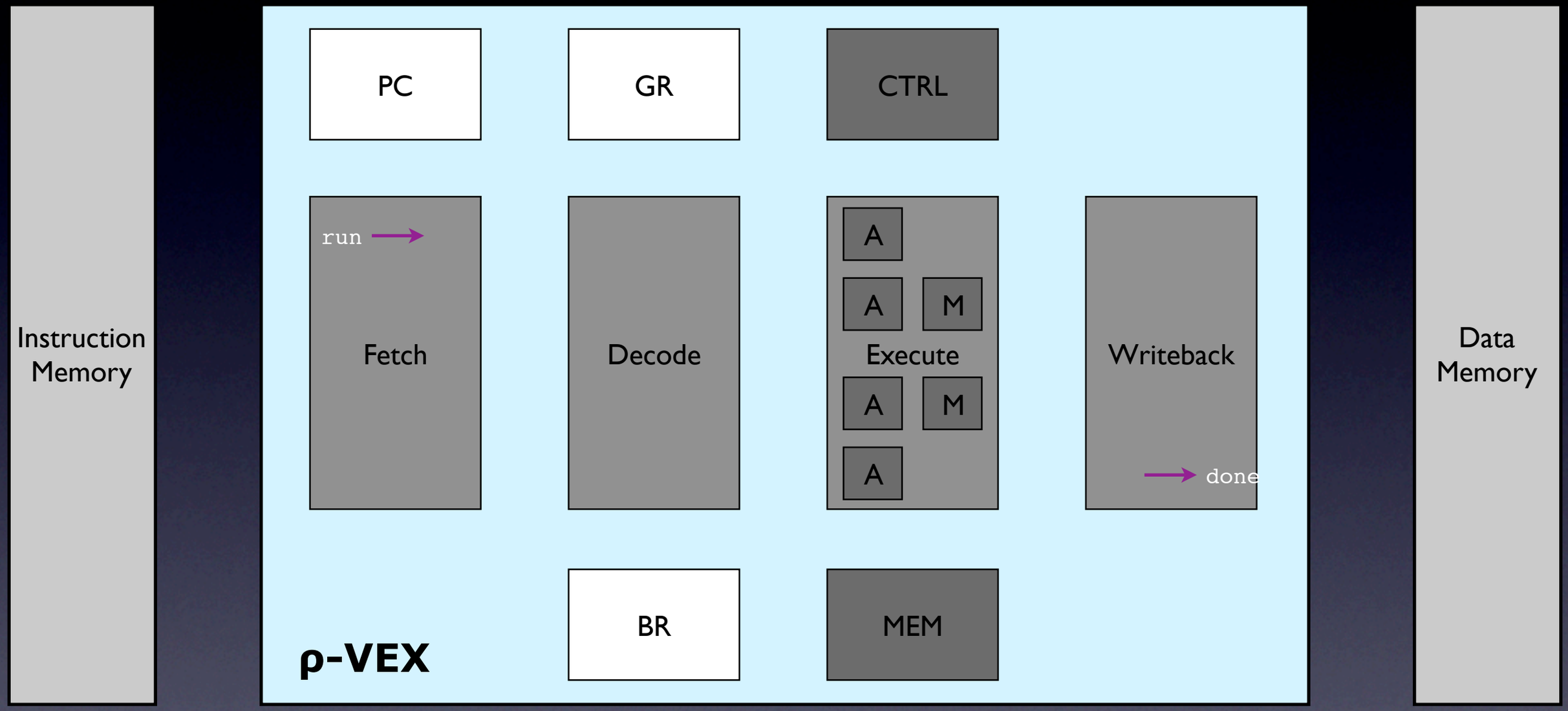
Data path

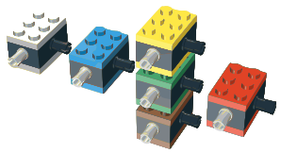

Instruction Memory
PC
GR
CTRL
Fetch
Decode
A
A
M
Execute
A
M
A
Writeback
Data Memory
BR
MEM
ρ-VEX

Control path



Control path



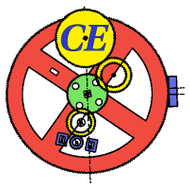


Design & Implementation

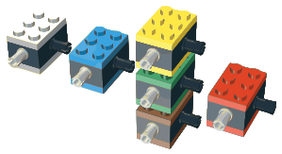
Extensibility

ρ-OPS

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



TU Delft

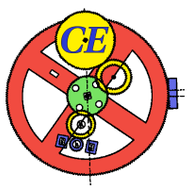


Design & Implementation

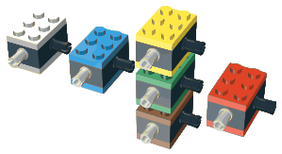
Extensibility

VEX machine models

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



TU Delft

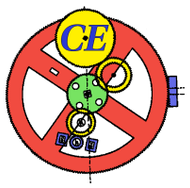


Design & Implementation

Verification

Unified Verification Methodology (UVM)

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

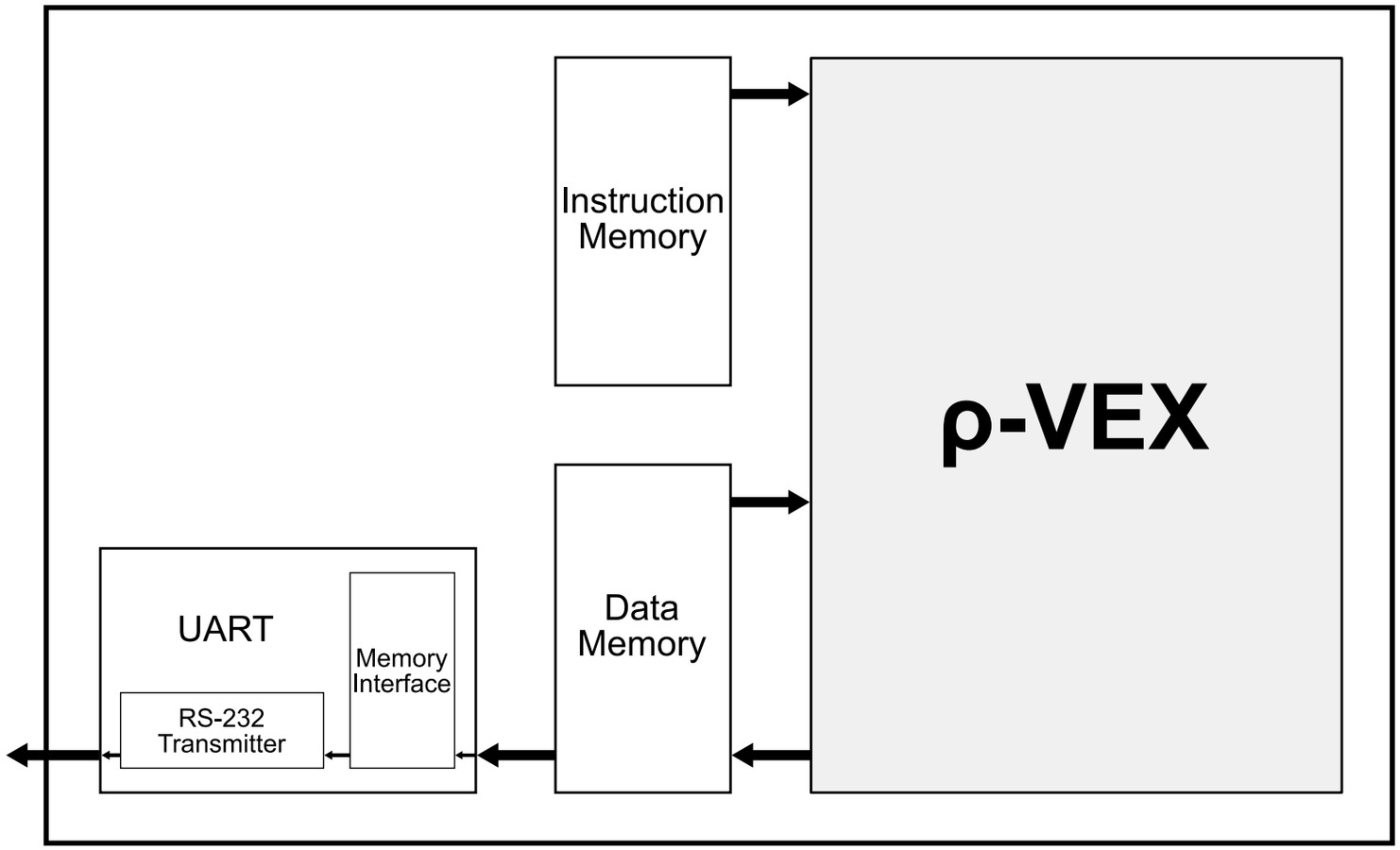


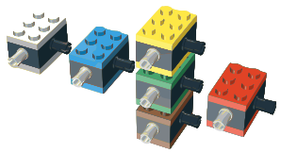
TU Delft

Design & Implementation

Verification

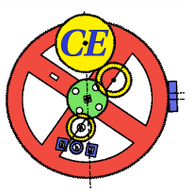
System wrapper



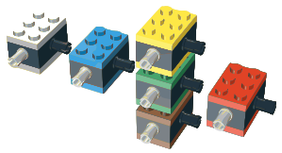


Overview

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

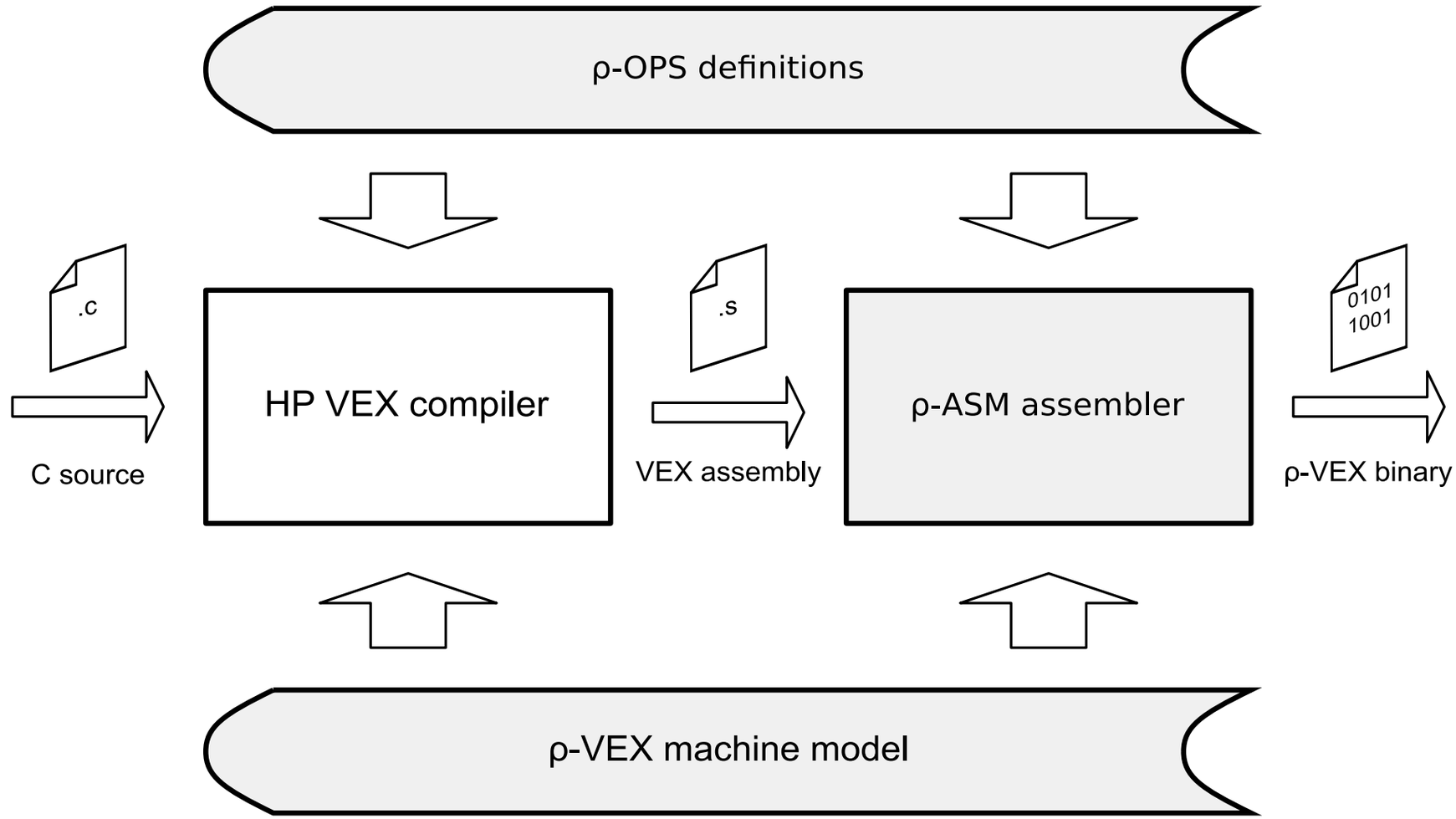


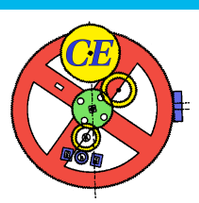
TU Delft



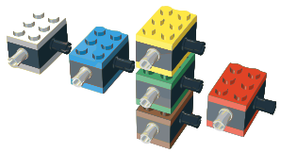
Development Framework

Development framework



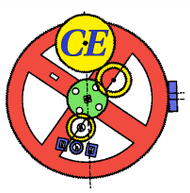


TU Delft

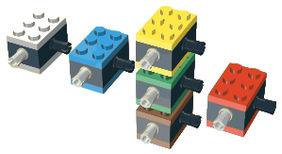


Overview

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



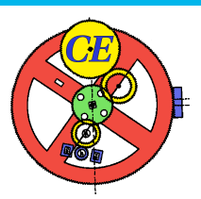
TU Delft



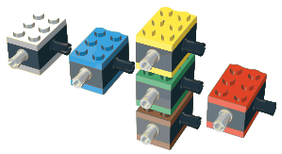
Experimental Results

Fibonacci's Sequence

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



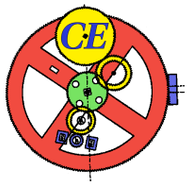
TU Delft



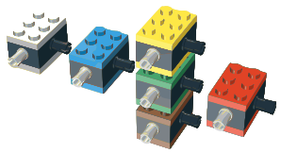
Experimental Results

Fibonacci's Sequence

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



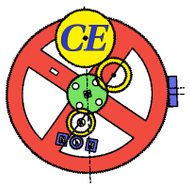
TU Delft



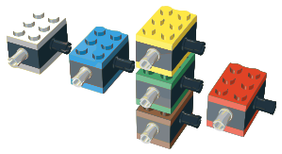
Experimental Results

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%)

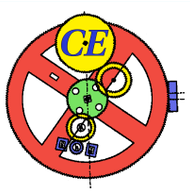


TU Delft

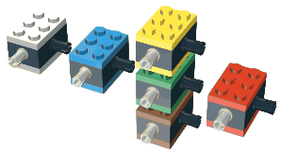


Overview

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



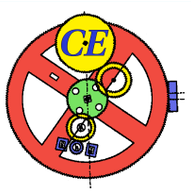
TU Delft



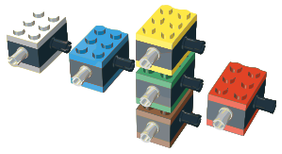
Conclusions

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



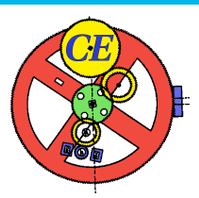
TU Delft



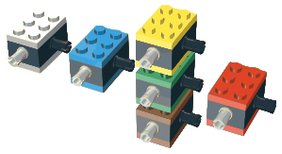
Conclusions

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



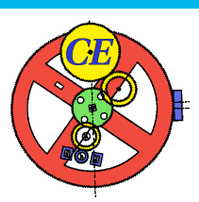
TU Delft



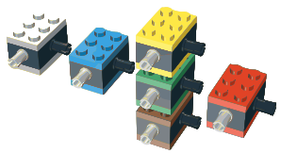
Conclusions

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

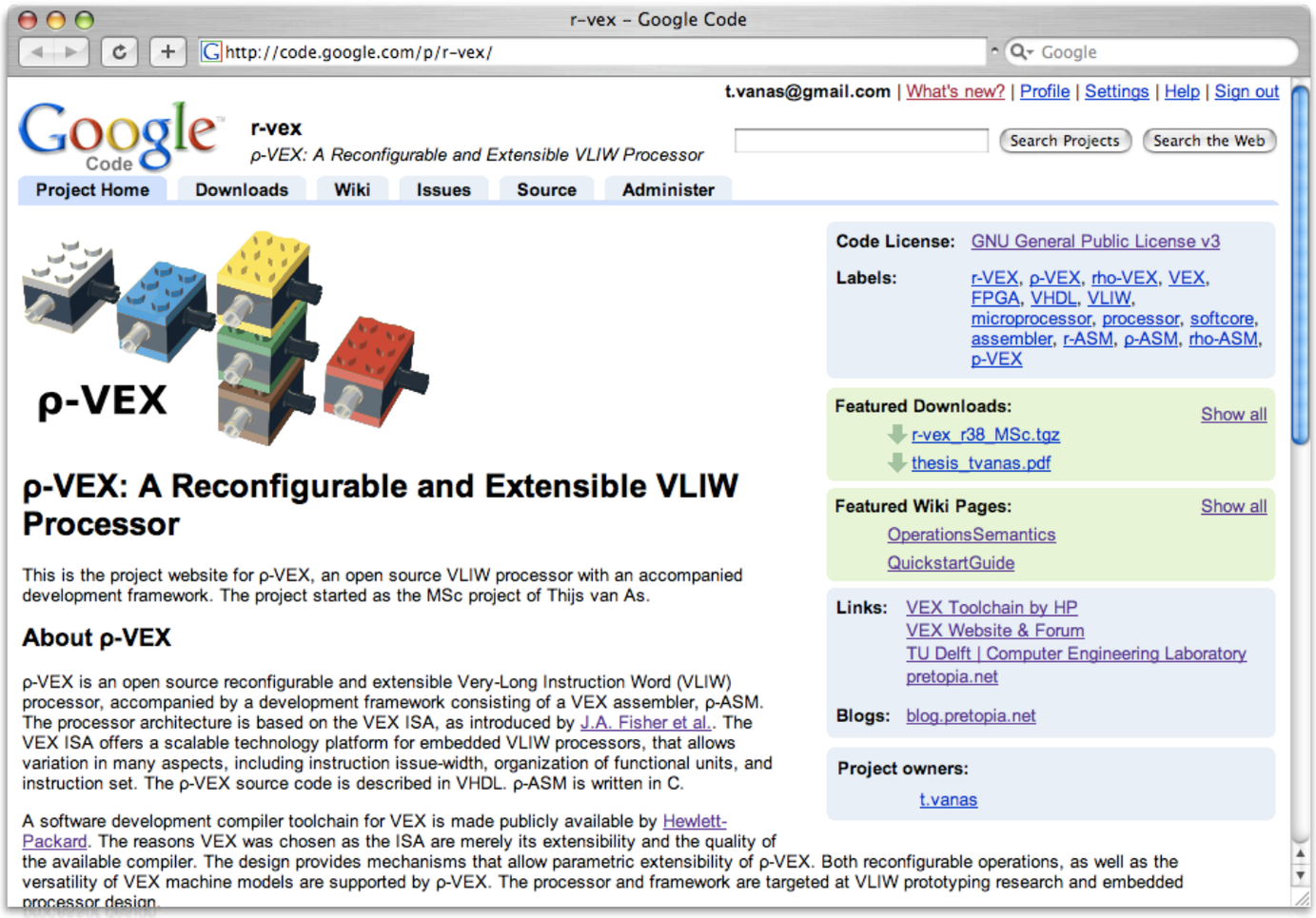


TU Delft

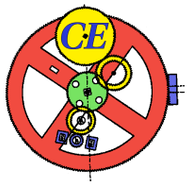


Questions

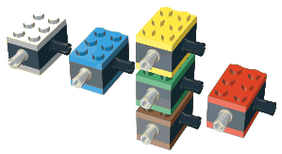
Thank you for your attendance!



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

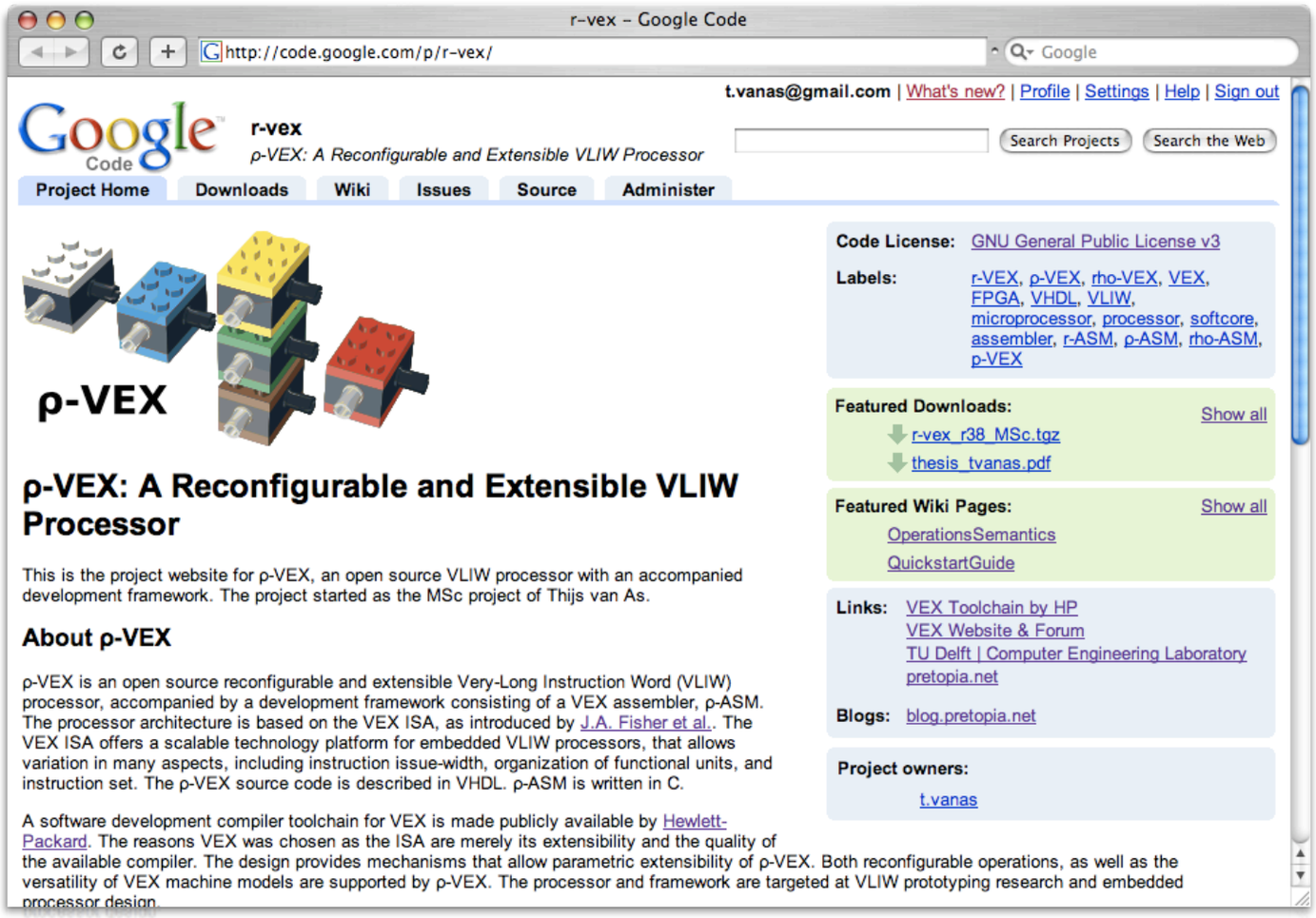


TU Delft

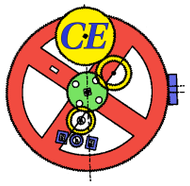


Questions

Thank you for your attendance!



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



TU Delft