DATE 25
DESIGN, AUTOMATION AND TEST IN EUROPE
THE EUROPEAN EVENT FOR ELECTRONIC SYSTEM DESIGN & TEST
31 MARCH – 2 APRIL 2025
LYON, FRANCE
CENTRE DE CONGRES DE LYON

A RISC-V ISA Extension for Chaining in Scalar Processors

Luca Colagrande, Jayanth Jonnalagadda, Prof. Dr. Luca Benini

ETH zürich
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
PULP Parallel Ultra Low Power

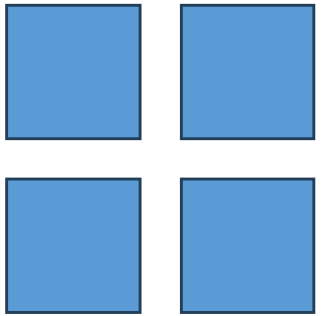
Context

General-purpose accelerators

Context

General-purpose accelerators

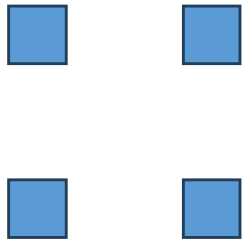
Built from arrays of *slimmed-down area-* and *energy-efficient* processors



Context

General-purpose accelerators

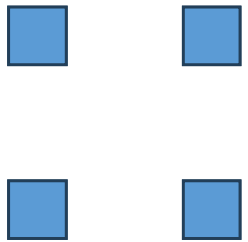
Built from arrays of *slimmed-down area-* and *energy-efficient* processors



Context

General-purpose accelerators

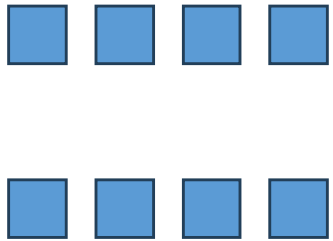
Built from arrays of *slimmed-down area-* and *energy-efficient* processors



Context

General-purpose accelerators

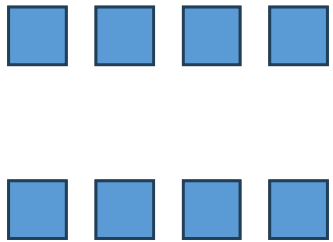
Built from arrays of *slimmed-down area-* and *energy-efficient* processors



Context

General-purpose accelerators

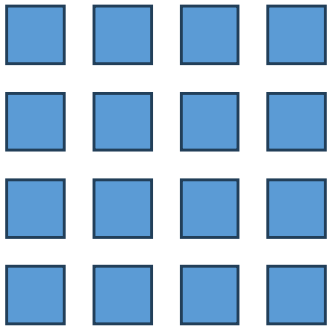
Built from arrays of *slimmed-down area-* and *energy-efficient* processors



Context

General-purpose accelerators

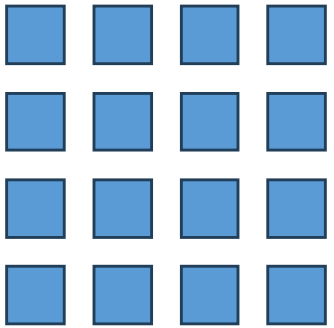
Built from arrays of *slimmed-down area-* and *energy-efficient* processors



Context

General-purpose accelerators

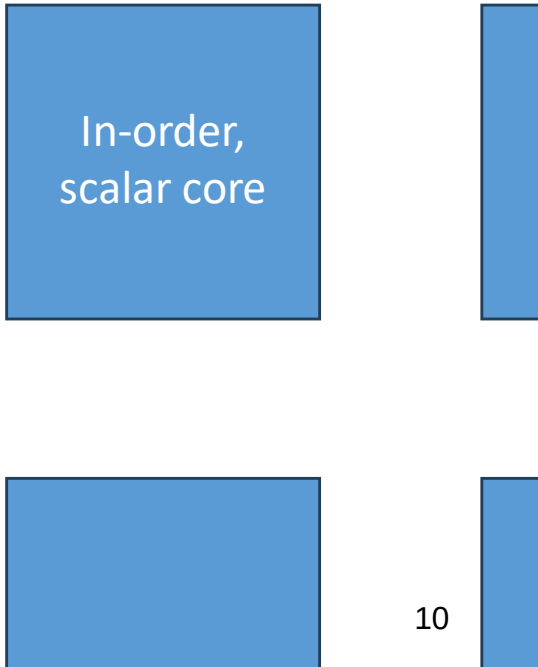
Built from arrays of *slimmed-down area-* and *energy-efficient* processors



Context

General-purpose accelerators

Built from arrays of *slimmed-down area-* and *energy-efficient* processors

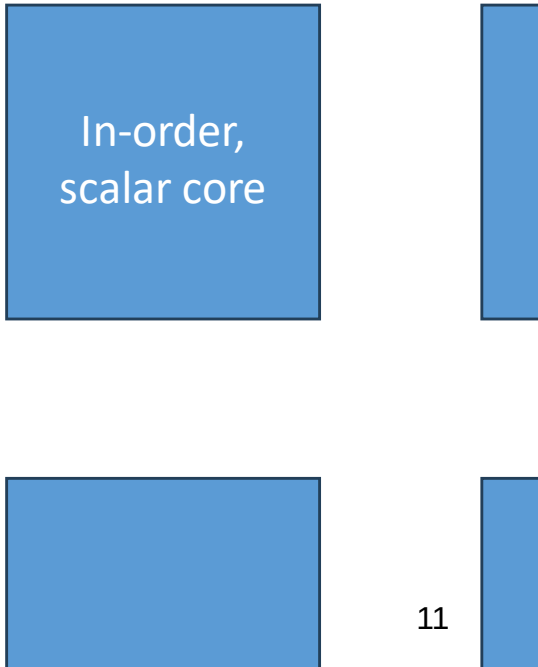


Context

General-purpose accelerators

Built from arrays of *slimmed-down area-* and *energy-efficient* processors

Highly sensitive to *pipeline stalls*
Unable to exploit ILP to hide stalls



Motivation

$$a_i = b * (c_i + d_i)$$

```
hwloop 2, %[len]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
```

RAW dependency

Motivation

```
hwloop  2, %[len]
fadd.d  ft3,  ft0,  ft1
fmul.d  ft2,  ft3,  %[b]
```

Motivation

Execution trace: (FPU with 3 pipeline stages)

```
hwloop  2, %[len]
fadd.d ft3, ft0, ft1
bubble
bubble
bubble
fmul.d ft2, ft3, %[b]
fadd.d ft3, ft0, ft1
```

```
hwloop  2, %[len]
                    , ft1
                    , %[b]
```

2 instructions every 5 cycles → low IPC

→ RAW dependency

Traditional solution

```
hwloop 2,%[len]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
```

Traditional solution

```
hwloop 2,%[len]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
```

Loop unrolling

Traditional solution

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
```

Loop unrolling

Traditional solution

```
hwloop  2,%[len/4]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
```

Loop unrolling

Software register renaming

Traditional solution

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fadd.d ft4, ft0, ft1
fmul.d ft2, ft4, %[b]
fadd.d ft5, ft0, ft1
fmul.d ft2, ft5, %[b]
fadd.d ft6, ft0, ft1
fmul.d ft2, ft6, %[b]
```

Loop unrolling

Software register renaming

Traditional solution

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fadd.d ft4, ft0, ft1
fmul.d ft2, ft4, %[b]
fadd.d ft5, ft0, ft1
fmul.d ft2, ft5, %[b]
fadd.d ft6, ft0, ft1
fmul.d ft2, ft6, %[b]
```

Loop unrolling

Software register renaming

Instruction reordering

Traditional solution

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft4, ft0, ft1
fadd.d ft5, ft0, ft1
fadd.d ft6, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft4, %[b]
fmul.d ft2, ft5, %[b]
fmul.d ft2, ft6, %[b]
```

Loop unrolling

Software register renaming

Instruction reordering

Traditional solution

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft4, ft0, ft1
fadd.d ft5, ft0, ft1
fadd.d ft6, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft4, %[b]
fmul.d ft2, ft5, %[b]
fmul.d ft2, ft6, %[b]
```

Loop unrolling

Software register renaming

Instruction reordering

Zero stalls

Traditional solution

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft4, ft0, ft1
fadd.d ft5, ft0, ft1
fadd.d ft6, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft4, %[b]
fmul.d ft2, ft5, %[b]
fmul.d ft2, ft6, %[b]
```

Loop unrolling

Software register renaming

Instruction reordering

Zero stalls

Increased register pressure

Traditional solution

```
hwloop  2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft4, ft0, ft1
fadd.d ft5, ft0, ft1
fadd.d ft6, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft4, %[b]
fmul.d ft2, ft5, %[b]
fmul.d ft2, ft6, %[b]
```

Loop unrolling

Software register renaming

Instruction reordering

Zero stalls

Increased register pressure

Traditional solution

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft4, ft0, ft1
fadd.d ft5, ft0, ft1
fadd.d ft6, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft4, %[b]
fmul.d ft2, ft5, %[b]
fmul.d ft2, ft6, %[b]
```

Loop unrolling

Software register renaming

Instruction reordering

Zero stalls

Increased register pressure

Traditional solution

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft4, ft0, ft1
fadd.d ft5, ft0, ft1
fadd.d ft6, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft4, %[b]
fmul.d ft2, ft5, %[b]
fmul.d ft2, ft6, %[b]
```

Loop unrolling

Software register renaming

Instruction reordering

Zero stalls

Increased register pressure

Limited applicability

Can we do better?

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft4, ft0, ft1
fadd.d ft5, ft0, ft1
fadd.d ft6, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft4, %[b]
fmul.d ft2, ft5, %[b]
fmul.d ft2, ft6, %[b]
```

Loop unrolling

Software register renaming

Instruction reordering

Zero stalls

Increased register pressure

Limited applicability

Can we do better?

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft4, ft0, ft1
fadd.d ft5, ft0, ft1
fadd.d ft6, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft4, %[b]
fmul.d ft2, ft5, %[b]
fmul.d ft2, ft6, %[b]
```

Loop unrolling

Software register renaming

Instruction reordering

Zero stalls

Increased register pressure

Limited applicability

Can we do better?

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft4, ft0, ft1
fadd.d ft5, ft0, ft1
fadd.d ft6, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft4, %[b]
fmul.d ft2, ft5, %[b]
fmul.d ft2, ft6, %[b]
```

Loop unrolling

Software register renaming

Instruction reordering

Zero stalls

No register pressure increase

Can we do better?

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft4, ft0, ft1
fadd.d ft5, ft0, ft1
fadd.d ft6, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft4, %[b]
fmul.d ft2, ft5, %[b]
fmul.d ft2, ft6, %[b]
```

Loop unrolling

~~Software register renaming~~

Instruction reordering

Zero stalls

No register pressure increase

Can we do better?

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
```

Loop unrolling

~~Software register renaming~~

Instruction reordering

Zero stalls

No register pressure increase

Can we do better?

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
```

Loop unrolling

~~Software register renaming~~

Instruction reordering

Zero stalls

No register pressure increase

Incorrect with regular semantics

Scalar chaining

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
```

Loop unrolling

~~Software register renaming~~

Instruction reordering

Zero stalls

No register pressure increase

Incorrect with regular semantics

Scalar chaining

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
```

Loop unrolling

~~Software register renaming~~

Instruction reordering

Zero stalls

No register pressure increase

Scalar chaining

```
hwloop 2,%[len/4]
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fadd.d ft3, ft0, ft1
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
fmul.d ft2, ft3, %[b]
```

Loop unrolling

~~Software register renaming~~

Instruction reordering

Zero stalls

No register pressure increase

Correct with FIFO semantics

Conclusion

- Demonstrate performance and energy efficiency improvements[1] in 4% and 10% range, respectively
- Measure negligible area and timing overhead


ETH zürich
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
PULP
A RISC-V ISA Extension for Chaining in Scalar Processors
Luca Colagrande[1], Jayanth Jonnalagadda[2], Luca Benini[1,3]
[1]Integrated Systems Laboratory (IIS) ETH Zürich; [2]D-ITET, ETH Zürich; [3]DEI, University of Bologna
1 Introduction
Modern general-purpose accelerators integrate a large number of programmable area- and energy-efficient processing elements (PEs), to deliver high performance while meeting stringent power delivery and thermal dissipation constraints. In this context, PEs are often implemented by scalar in-order cores, which are highly sensitive to pipeline stalls. Traditional software techniques, such as loop unrolling, mitigate the issue at the cost of increased register pressure, limiting flexibility. We propose
The RF is augmented with a valid bit per register to implement head-of-line blocking at the consumer's side of the logical FIFO.

[1] P. Scheffler et al., "Saris: Accelerating stencil computations on energy-efficient risc-v compute clusters with indirect stream registers," in DAC'24: Proceedings of the 61st ACM/IEEE Design Automation Conference.