www.inesc-id.pt

Read the Paper!

Unlimited Vector Extension with Data Streaming Support

Joao Mario Domingos[1], Nuno Neves[1,2], Nuno Roma[1], Pedro Tomás[1]

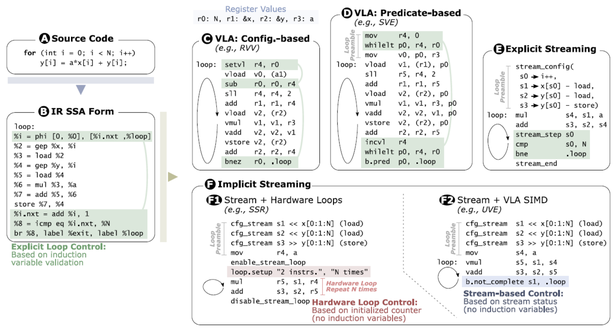
[1]INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal
[2]Instituto de Telecomunicações, Coimbra, Portugal

Funded by: FCT Fundação para a Ciência e a Tecnologia; European Processor Initiative

Summary

Unlimited vector extension (UVE) is a novel instruction set architecture extension that takes streaming and SIMD processing together into the modern computing scenario. It aims to overcome the shortcomings of state-of-the-art scalable vector extensions by adding data streaming as a way to simultaneously reduce the overheads associated with loop control and memory access indexing, as well as with memory access latency. This is achieved through a new set of instructions that pre-configure the loop memory access patterns. These attain accurate and timely data prefetching on predictable access patterns, such as in multidimensional arrays or in indirect memory access patterns. Each of the configured data streams is associated to a general-purpose vector register, which is then used to interface with the streams. In particular, iterating over a given stream is simply achieved by reading/writing to the corresponding input/output stream, as the data is instantly consumed/produced.

To evaluate the proposed UVE, a proof-of-concept gem5 implementation was integrated in an out-of-order processor model, based on the ARM Cortex-A76, thus taking into consideration the typical speculative and out-of-order execution paradigms found in high-performance computing processors. The evaluation was carried out with a set of representative kernels, by assessing the number of executed instructions, its impact on the memory bus and its overall performance. Compared to other state-of-the-art solutions, such as the ARM Scalable Vector Extension (SVE), the obtained results show that the proposed extension attains average performance speedups over 2.4x for the same processor configuration, including vector length.



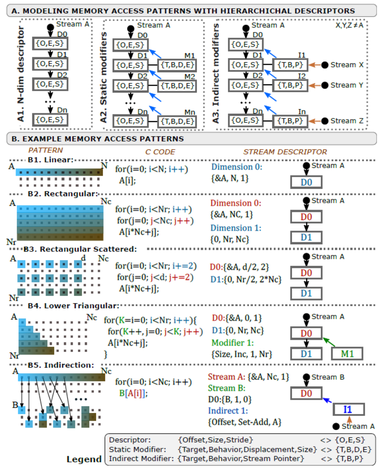
Background and Motivation

Vector-length agnostic SIMD limitations

- Conventional SIMD extensions operate with fixed-size registers (e.g., Intel AVX, ARM NEON).
- Recent vector-length agnostic (VLA) SIMD extensions (e.g., ARM SVE and RISC-V RVV) abstract the physical vector register size in the source code, allowing different processors to adopt distinct vector sizes, while requiring no code modifications.
- VLA extensions impose non-negligible loop overheads with predicate (ARM SVE) and vector configuration instructions (RVV) to both control the loop iteration and to disable vector elements outside loop bounds.

The Unlimited Vector Extension (UVE)

- **Memory Access Decoupling –** Adoption of a *stream-based paradigm* to directly stream data to the register file, decoupling memory accesses from computation, and allowing data load/store to occur in parallel with data manipulation.
- **Indexing-free loops –** Memory access patterns (for each load/store) are exactly described at the loop preamble, effectively *removing memory access and address calculation instructions from the code*, accelerating the loop.
- **Simplified vectorization –** Transparent scatter-gather operations for *complex, multi-dimensional, strided, and indirect patterns* are performed by a Streaming Engine, transforming non-coalesced accesses into linear patterns automatically aligned for vectorization.
- **Implicit load/store –** *Each active data stream is associated with a different vector register*, allowing reading/writing to/from the register to automatically trigger the input/output (load/store) stream iteration, without additional stream stepping instructions.
- **Register-size agnostic -** UVE code is agnostic to the register size. Operations over out-of-bound elements (e.g., when the number of elements to process is not a multiple of the vector length) is prevented by *automatically disabling all vector register elements that fall out of bounds*, according to stream iteration.



UVE RISC-V Extension*

UVE currently features a RISC-V instruction extension with 450 instructions comprising integer, floating-point and memory (including streaming) instructions, including support for:

- Stream configuration
- Stream control
- Vector register predication
- Loop control
- Vector manipulation
- Scalar processing

Streaming Engine

The UVE Streaming Engine is responsible for managing all input and output streams.

Its architecture is composed of:

- A **Stream Configuration** module that accepts stream descriptors from the loop preamble
- A **Stream Table** that maintains all stream information and iteration state
- A **Stream Scheduler** responsible for the selection and issuing of stream descriptors
- **Stream Load/Store Processing Modules** that perform the descriptor iteration and address generation
- **Load/Store FIFOs and Memory Request Queues** responsible for hold outstanding memory requests and storing input/output stream data

*Compilation methods for UVE and stream-based extensions are already under development.

Check the paper here

Data Streaming Model

In the UVE streaming model, a ***stream*** is defined as a *predicable n-dimensional sequence of data that is transferred between the memory and the processor*.

Memory access modeling and description

UVE adopts a compiler-friendly memory access representation model that combines nested loop-based indexing and loop- or data-dependent (indirect) index dynamic ranges into an n-dimensional affine function:

$$y(X) = y_{base} + \sum_{k=0}^{dim_s} x_k \times S_k$$
$$\textbf{with } X = \{x_0, \ldots, x_{dim_s}\} \textbf{ and } x_k \in [O_k, E_k + O_k]$$

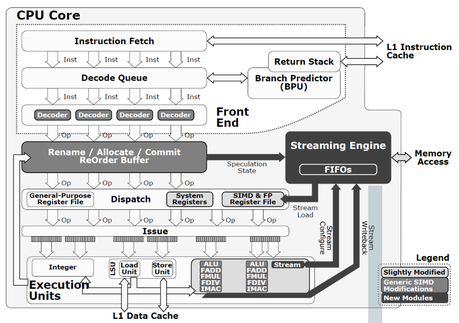
Each stream access (y) is represented as the sum of a base address with per-dimension (k) pairs of indexing variables (x - within a range defined by the loop header) and stride multiplication factors (S).

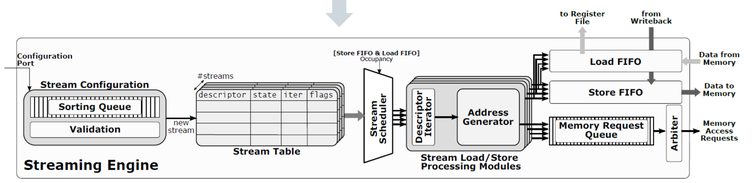
- Each stream is represented by a hardware-friendly hierarchical descriptor representation that encodes the variables of each affine function dimension.
- Combines multiple functions to represent complex, dynamic, and/or indirect access patterns.

UVE Microarchitecture Support

The proposed UVE provides a processor base architecture extension comprising inclusion of:

1. 32 stream vector registers with configurable size.
2. A streaming interface to associate each register with a data stream and perform its iteration.
3. 16 predicate registers to handle conditional code.
4. Register and stream renaming to support speculative execution.
5. Support for commit and squash of streams to handle miss-speculation and commit events.

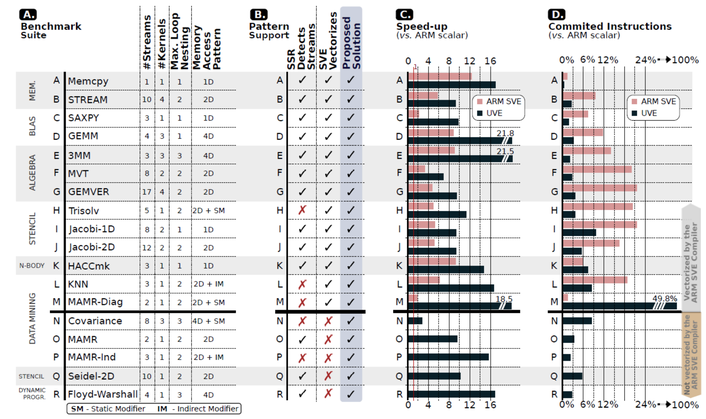




Experimental Setup

- Modified Gem5 simulator to support the proposed UVE microarchitecture and streaming engine.
- Comparison with baseline simulator configurations emulating an ARM Cortex-A76 with NEON and SVE support.
- Selection of representative benchmarks from several application domains, such as memory, linear algebra/BLAS, stencil, data mining, dynamic programming, and n-body (physics) systems.

CPU (@1.5GHz)	4-wide instruction fetch, 4-wide μOp commit 8-wide μOp issue/dispatch/writeback 80 IQ, 32 LQ, 48 SQ, 128 ROB entries 128 Int RF, 192 FP RF, 48×512-bit Vector RF
Functional Units	2×Int ALUs with a 24-entry scheduler 2×Int vector/FP FUs with a 24-entry scheduler 2×Load + 1×Store ports with a 24-entry scheduler
Streaming Engine	2×Stream Load/store Processing Modules 8-entry Load/Store FIFOs per stream (default) 1×Load + 1×Store ports with a 24-entry scheduler
L1-I / L1-D	64KB 4-Way LRU Stride Prefetcher with depth 16
L2	256 KB 8-Way LRU AMPM Prefetcher [20], QueueSize 32 Snoop-based cache coherence protocol
DRAM	Dual-Channel DDR3-1600 8x8 11-11-11



Results, Conclusions, and Future Work

- Memory access linearization provides improved vectorization capabilities over the ARM compiler allowing the vectorization of a wider range of complex loop patterns.
- Average performance improvement by 2.4x over ARM SVE:
  - Loop acceleration through code size reduction, with an average 60.9% less committed instructions.
  - Significant load-to-use latency reduction, resulting from the streaming engine preemptive and autonomous data acquisition to the vector registers.
  - Increased effective memory bandwidth utilization, from the streaming infrastructure management and data buffering.
- UVE imposes minimal hardware impacts with the inclusion of the Streaming Engine (with an estimated footprint close to ½ of an L1 cache).
- Hardware prototypes currently under development, targeting modern processor pipelines and dedicated acceleration frameworks.

CONTACT US

nuno.neves@inesc-id.pt

Acknowledgments: This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) under projects UIDB/50021/2020 and PTDC/EEI-HAC/30485/2017, and by funds from the European Union Horizon 2020 Research and Innovation programme under grant agreement No. 826647.