Accelerating Irregular Workloads with Cooperating Indexed Stream Registers

Paul Scheffler[1]*, Luca Benini[1,2]

[1]Integrated Systems Laboratory, ETH Zurich
[2]Department of Electrical, Electronic, and Information Engineering, University of Bologna

Abstract

Sparse and irregular workloads are crucial to various data-driven applications including computational physics, graph analytics, and sparse neural networks. However, their high control overhead and irregular memory access patterns are inefficiently handled by today's architectures, resulting in low functional unit utilization and low overall efficiency. While proposed hardware solutions accelerate irregular workloads, most are held back by a lack of generality or their large architectural impact. We present Cooperating Indexed Stream Registers*, a lightweight RISC-V ISA extension building on stream registers to accelerate three key operational patterns on index streams underlying many irregular workloads:* indirection*,* intersection*, and* union*. Handling these in hardware can accelerate one- and two-sided sparse linear algebra, stencil workloads, sparse neural networks, and quantized data streaming among others. In an existing eight-core RISC-V compute cluster, our extensions incur only 1.8% in additional area over affine stream registers, but accelerate sparse-dense linear algebra, sparse-sparse linear algebra, and stencil codes by up to 5.0x, 5.9x, and 3.7x, respectively while consuming up to 3.0x less energy.*

Introduction

Large-scale data-driven applications like sparse machine learning (ML), graph analytics, and computational physics are shifting computer architects' attention toward *irregular* workloads, which exhibit high control overhead and data-dependent memory accesses. However, today's processors are tuned for regular compute: even TOP500 systems achieve only tiny fractions of their peak compute on irregular workloads [1].

Many hardware proposals accelerate irregular workloads, but lack generality or incur significant hardware overheads. *Vector processors* accelerate indirection with scatter-gather, but require registered indices, limited vector lengths, and cannot accelerate operations between two sparse operands. Recent *GPU extensions* target sparse operations, but only efficiently handle low (structured or bitmapped) sparsity in either one [2] or both operands [3]. Some *accelerators* target general sparse linear algebra (LA) [4], but impose large area impacts, loose coupling, or restrictive dataflows.

One promising acceleration approach are *stream registers* (SRs), which map *memory streams* to architectural registers. We present *Cooperating Indexed Stream Registers* (CISRs) [5], a lightweight RISC-V ISA extension accelerating *general* irregular workloads; this includes one- and two-sided sparse LA, but also non-LA workloads like stencils, sparse convolution, and quantized streaming. CISRs extend an existing SR design targeting regular workloads [6] to accelerate three operations on *index streams* underlying many irregular workloads: *indirection*, *intersection* and *union*. CISRs are lightweight and modular, reusing existing datapaths where possible and incurring only 1.8 % in additional area in an eight-core RISC-V compute cluster [7] with affine SRs. In multicore workloads, CISRs enable speedups of up to 5.0×, 5.9×, and 3.7× over an RV32G baseline on sparse-dense LA, sparse-sparse LA, and stencils, respectively, enabling FPU utilizations of up to 93 % and consuming up to 3.0× less energy.

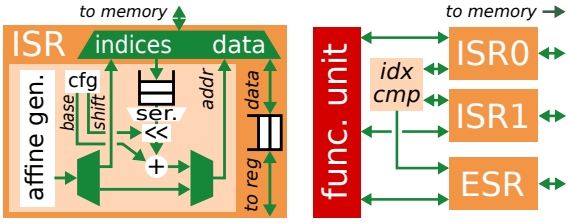
*Corresponding author: paulsc@iis.ee.ethz.ch

Architecture

We keep the existing SRs' interfaces [6]: streams are set up through custom instructions and SRs autonomously stream data to or from memory. Accesses to mapped registers implicitly push or pop streams, enabling continuous streaming even in single-issue in-order cores.

We add streaming *indirection* capabilities to our baseline SR, creating the *indexed stream register* (ISR) shown in Fig. 1a. We extend the address generator to fetch index arrays from memory, add them to a configured base pointer, and use the resulting addresses for indirect read (gather) or write (scatter) requests. This is similar to RVV indexed accesses, but we fetch indices directly from memory where they originate in data-dependent workloads, enabling near-memory index processing and much longer continuous streams.

We support 8, 16, 32, and 64 b indices, as well as a programmable shift enabling indirection into higher-level data axes or struct arrays. We fetch indices in packets utilizing the memory bus' full width. Indices and data share a single memory port, slightly reducing peak throughput to preserve memory topology. We retain the base SR's affine streaming capabilities, which we reuse to generate addresses for the fetched indices.



(a) *Indexed SR* **(b)** *Cooperating ISRs*

Figure 1: *Architecture of CISR extension components.*

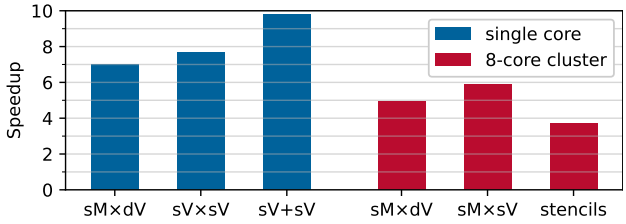


Figure 2: *Peak CISR single core and cluster speedups.*

To handle operations *between* sparse operands, two or three of our SRs *cooperate*, which we then call CISRs. For this, we assume that as in common sparse tensor formats like CSR or CSF, operands are composed of *sparse vectors* encoded as two arrays, one storing ordered nonzero *values* and one storing their *indices*.

For *multiplicative* operations, we form the *intersection* of operand indices to find value pairs that influence the result, while *additive* operations require forming their *union*. We do this by adding an *index comparator* between two ISRs as shown in Fig. 1b. Each ISR reuses its existing datapath to stream the indices of one operand. The comparator advances whichever index stream has the smaller current index.

When intersecting, the ISRs fetch the current index' value only if both indices match, avoiding redundant accesses. During union, if an index occurs only in one operand, the opposite ISR emits a neutral additive element (0). In both cases, the we can continuously issue useful instructions using the SRs without control overhead; the comparator informs our branch unit or hardware loop when a joint stream ends. For element-wise operations, the joint indices may be read by a third *egress SR* (ESR) writing them out alongside register-written values as a sparse result vector.

Results

CISRs are highly versatile. In addition to many workloads leveraging sparse-dense or sparse-sparse LA, they can accelerate any scatter-gather task and stream arbitrary memory sequences to or from registers. This enables workloads with irregular, but repeated access patterns like stencils or sparse convolution to achieve near-full FPU utilization. They can also stream *compressed* high-precision data encoded as compact indices indexing a lookup table. Through intersection, they can even accelerate graph pattern matching.

We evaluate CISRs by integrating them in each of the eight cores of the RV32 Snitch compute cluster [7]. We replace the previously included three affine SRs [6], which stream between the internal 64-bit-wide tightly-coupled scratchpad and registers `ft0` to `ft2`.

In GlobalFoundries' 12LP+ technology, each ISR incurs only 10 kGE in area, 3 kGE more than an affine SR. A CISR team with an ESR incurs only 11 kGE more per core than affine SRs, resulting in only 1.8 % of additional cluster area and no clock speed impact.

Figure 2 shows peak workload speedups enabled by CISRs over an RV32G baseline. On a single core, CISR *indirection* enables sparse-dense LA speedups of up to 7.0× and FPU utilizations of up to 79 %, while *intersection* and *union* enable sparse-sparse multiply and add speedups of up to 7.7× and 9.8×, respectively. In an eight-core cluster, CISRs enable sparse-dense matrix-vector multiply (M×V) speedups of up to 5.0× and use up to 2.9× less energy. Sparse-sparse M×V is up to 5.9× faster and uses up to 3.0× less energy. Selected cluster stencil kernels are up to 3.7× faster and achieve FPU utilizations of up to 93 %.

Conclusion

We present CISRs, a lightweight RISC-V SR extension accelerating general irregular workloads. By extending an SR architecture with hardware *indirection*, *intersection*, and *union*, we create a highly versatile extension enabling multicore speedups of up to 5.9× and energy savings of up to 3.0× over a RV32G baseline. We hope that CISRs can provide a first step toward formal RISC-V SR and irregular workload ISA extensions.

References

[1] *June 2022 HPCG Results*. `https://www.hpcg-benchmark.org/custom/index.html%3Flid=155&slid=313.html`.

[2] Nvidia. *NVIDIA A100 Tensor Core GPU Architecture*.

[3] Yang Wang et al. "Dual-side Sparse Tensor Core". In: *2021 ACM/IEEE ISCA 48*. 2021, pp. 1083–1095.

[4] Shail Dave et al. "Hardware Acceleration of Sparse and Irregular Tensor Computations of ML Models: A Survey and Insights". In: *Proc. IEEE* 109 (2021), pp. 1706–1752.

[5] Paul Scheffler et al. "Sparse Stream Semantic Registers: A Lightweight ISA Extension Accelerating General Sparse Linear Algebra". Not yet published.

[6] Fabian Schuiki et al. "Stream Semantic Registers: A Lightweight RISC-V ISA Extension Achieving Full Compute Utilization in Single-Issue Cores". In: *IEEE Trans. Comput.* 70 (2021), pp. 212–227.

[7] Florian Zaruba et al. "Snitch: A Tiny Pseudo Dual-Issue Processor for Area and Energy Efficient Execution of Floating-Point Intensive Workloads". In: *IEEE Trans. Comput.* 70 (2021), pp. 1845–1860.