DESIGN, AUTOMATION & TEST IN EUROPE
01 – 05 February 2021 · virtual conference
The European Event for Electronic System Design & Test

Indirection Stream Semantic Register Architecture for Efficient Sparse-Dense Linear Algebra

Paul Scheffler	*IIS, ETH Zurich*	<paulsc@iis.ee.ethz.ch>
Florian Zaruba	*IIS, ETH Zurich*	<zarubaf@iis.ee.ethz.ch>
Fabian Schuiki	*IIS, ETH Zurich*	<fschuiki@iis.ee.ethz.ch>
Torsten Hoefler	*SPCL, ETH Zurich*	<htor@inf.ethz.ch>
Luca Benini	*IIS, ETH Zurich*	<lbenini@iis.ee.ethz.ch>

Sparse Linear Algebra

[1] B. Xie, J. Zhan, X. Liu, W. Gao, Z. Jia, X. He, and L. Zhang, "CVR: efficient vectorization of SpMV on x86 processors," in CGO '18, 2018, pp. 149–162.

[2] Nvidia Corporation, "Cuda Toolkit 10.0." [Online]. Available: https://developer.nvidia.com/cuda-toolkit

- Sparse tensors are *ubiquitous*
  - ML, CS, physics, economics…
  - Various formats and algorithms
- Common: *CSR Matrix × Vector* (CsrMV)
  - Large *control-to-compute* ratio
  - Low data reuse
  - *Memory indirection*

➢ SoA: low functional utilization
  - Xeon Phi KNL (CVR format[1]): 0.7% FP64
  - AGX Xavier (CuSPARSE[2]): 2.1% FP32

$$\vec{y} = \begin{bmatrix} 0 & 0 & 7 & 9 & 0 \\ 3 & 0 & 4 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 7 \\ 3 \\ 2 \\ 9 \end{bmatrix}$$

$\boldsymbol{A}$ $\quad$ $\vec{\boldsymbol{x}}$

```
A_vals[] = {7, 9, 3, 4, 1, 2, 7}
A_idcs[] = {2, 3, 0, 2, 3, 4, 2}
A_ptrs[] = {0, 2, 3, 3, 4}
```

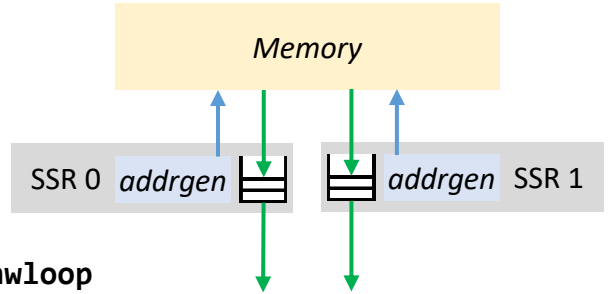
```
for i in 0 to A_nrows:
  for j in A_ptrs[i] to A_ptrs[i+1]:
    y[i] += A_vals[j] * x[A_idcs[j]]
```

[3] F. Schuiki, F. Zaruba, T. Hoefler, and L. Benini, "Stream Semantic Registers: A Lightweight RISC-V ISA Extension Achieving Full Compute Utilization in Single-Issue Cores," IEEE Trans. Comput., pp. 1–1, 2020.

Stream Semantic Registers[3]

- Lightweight RISC-V ISA extension
  - *Dense* LA: up to 100% FPU utilization
- Map registers to *memory streams*
  - Accesses become loads/stores
  - *Fixed-stride* address generators
  - Orthogonal to HW loops
- *Sparse-dense* LA: limited by *indirection*
  - With SSR: 11 → 14% max FPU utilization

➢ **Indirect *inside* SSR → issue only fmadds**




```
dotp: hwloop
      fmadd.d  ft2, ft0, ft1, ft2
```

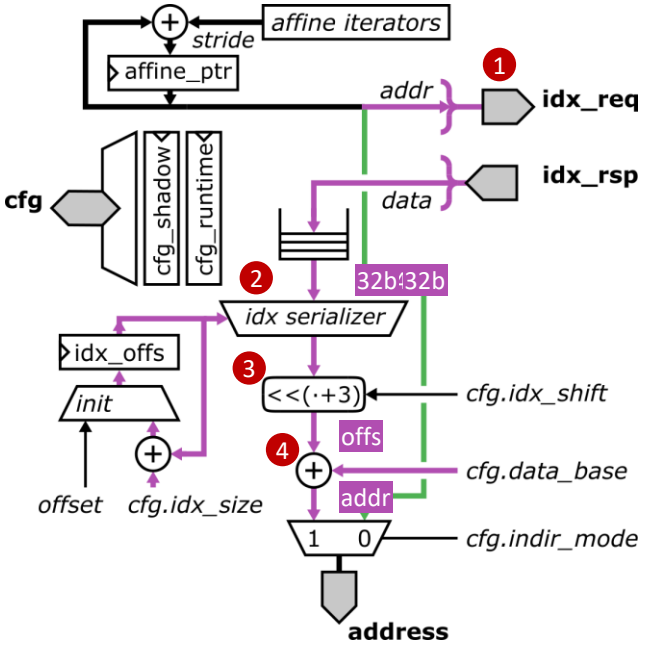
```
spvv: lw      t0, 0(%A_idcs)      ┐
      slli    t0, t0, 3           │  ft0 ← x[A_idcs[j]]
      add     t0, t0, %x          │
      fld     ft0, 0(t0)          ┘
      fld     ft1, 0(%A_vals)
      fmadd.d %y, ft0, ft1, %y
      addi    %A_idcs, %A_idcs, 4
      addi    %A_vals, %A_vals, 8
      bne     %A_vals, %A_end, spvv
```

Our Contributions

1. *Indirection SSR* (ISSR) **architecture**
2. **Programming model** and **sparse-dense product kernels**
3. Significant **performance** and **energy** benefits
4. Comparison to SoA **CPU, GPU, HW accelerator** approaches

Address Generator Extension

- Idea: Index lookup *inside* address generator
  - Add read-only memory port
  - Stay backward-compatible
- Indirection:
  1. Fetch *64b* index *words*
  2. Serialize to *16b* or *32b* indices
  3. Shift: 64b offset (FP64) + dense axis stride
  4. Add base address

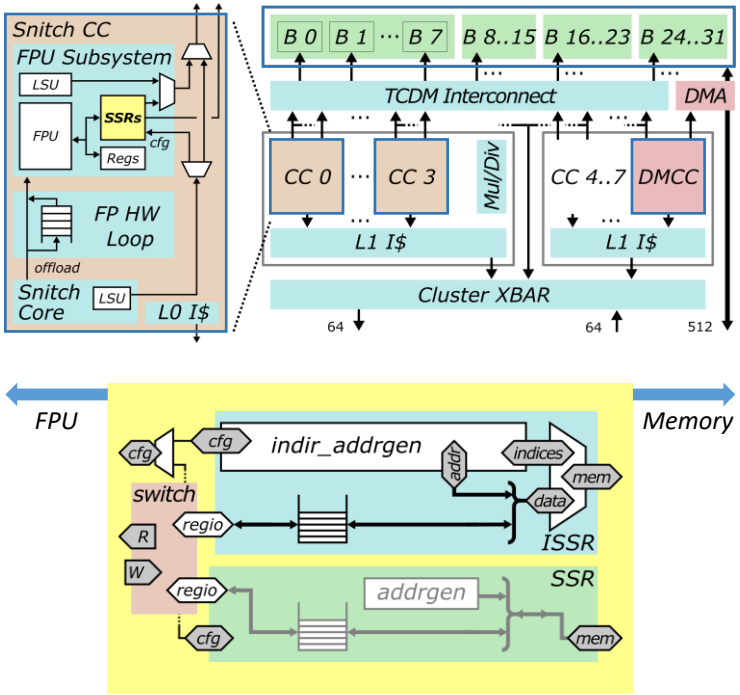


Integration for Evaluation

[4] F. Zaruba, F. Schuiki, T. Hoefler, and L. Benini, "Snitch: A tiny pseudo dual-issue processor for area and energy efficient execution of floating-point intensive workloads," IEEE Trans. Comput., pp. 1–1, 2020.

[5] A. Kurth, W. Rönninger, T. Benz, M. Cavalcante, F. Schuiki, F. Zaruba, and L. Benini, "An Open-Source Platform for High-Performance Non-Coherent On-Chip Communication," arXiv:2009.05334 [cs.AR], 2020.

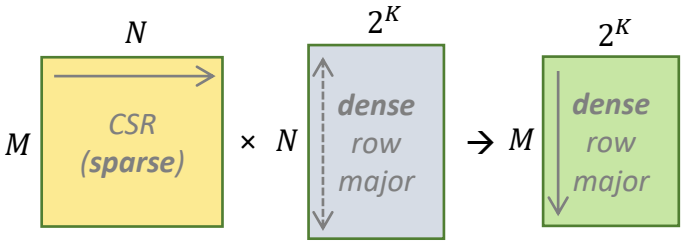
- Evaluated in *Snitch* Cluster[4]
  - 8 cores: DP-FPU with 2 SSRs, `frep`
  - 1 core: 512-bit DMA[5]
  - 256 KiB banked scratchpad
- 1 ISSR + 1 SSR per core
  - +4.4 kGE in address generator
  - +24% SSR, +0.8% cluster area
- Combine ISSR data, index ports
  - ➢ Prefer 80% max. utilization over 1.5× interconnect area



Sparse-Dense Product Kernels

- *Dot product* (SpVV): minimal ISSR showcase
  1. Setup: set up SSRs, accumulators
  2. `frep` loop: *continuous fmadd stream*
  3. Teardown: reduce, write back
- `frep` *decouples* core and FPU
  - Core free to set up next job
- *CsrMV:* accelerated through *row unrolling*, streaming entire matrix in *one job*
- *CsrMM:* reuse CsrMV with 2^K *stride*

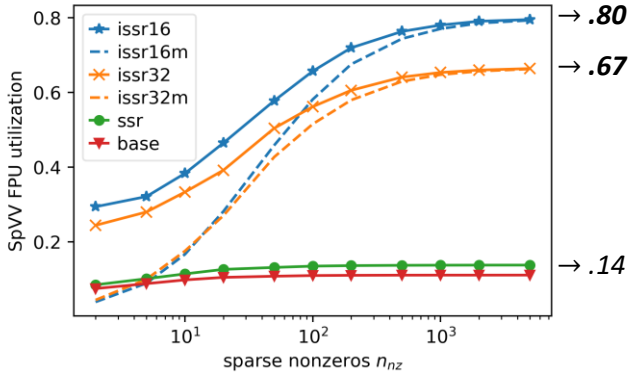
```
call      ssr_setup_ft0
call      issr_setup_ft1
csrsi     ssr_ena, 1
fcvt.d.w  ft2, zero
…
frep      %len
fmadd.d   ft2, ft0, ft1, ft2
…
fadd.d    ft8, ft6, ft7
fsd       ft8, 0(%res)
csrci     ssr_ena
```

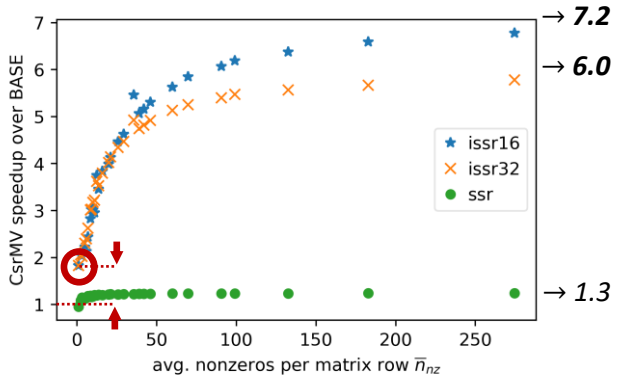


[6] T. A. Davis and Y. Hu, "The University of Florida Sparse Matrix Collection," ACM Trans. Math. Softw., vol. 38, no. 1, Dec. 2011.

Results: Single-Core Performance

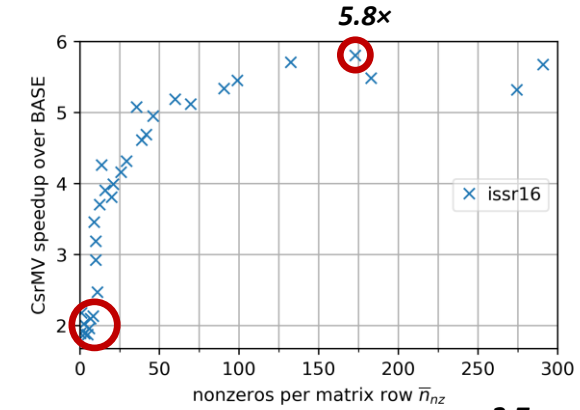
- Compare **16b** / **32b** index ISSR kernels to
  - BASE: optimized RISC-V baseline
  - SSR: regular SSRs only
- All data in cluster memory
- *SpVV*: up to **80%** / **67%** FPU utilization
  - Slower ISSR convergence due to high peak
- *CsrMV*: approach **7.2×** / **6.0×** speedups
  - Real-world matrices[6]: 2-3.2k cols, 1.3-680k nonzeros
  - Optimizations: decent speedup even for low $\overline{n}_{nz}$





Results: Cluster CsrMV

- Eight-core CsrMV with same kernels
  - Same real-world matrices
  - Distribute rows, stationary vector
  - Double-buffered DMA matrix transfer
- Despite parallelization: up to **5.8×** faster
  - **1.9×** for $\overline{n}_{nz} = 1$, over **5×** for $\overline{n}_{nz} \geq 50$
  - Reduced by: row imbalance, work sharing, bank conflicts, I$ misses, vector transfer
- Up to **2.7×** more energy-efficient
  - **142 → 53 pJ** per sparse-dense MAC



Related Work

[1] B. Xie, J. Zhan, X. Liu, W. Gao, Z. Jia, X. He, and L. Zhang, "CVR: efficient vectorization of SpMV on x86 processors," in CGO '18, 2018, pp. 149–162.

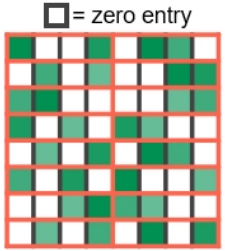
[2] Nvidia Corporation, "Cuda Toolkit 10.0." [Online]. Available: https://developer.nvidia.com/cuda-toolkit

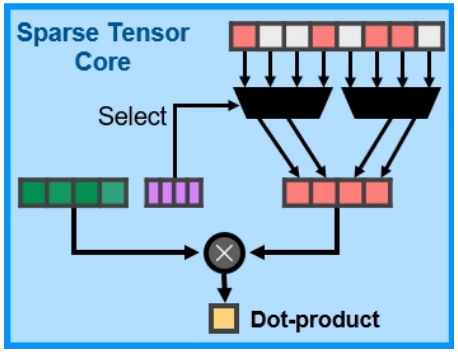
[7] A. Sodani, R. Gramunt, J. Corbal, H. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y. Liu, "Knights Landing: Second-Generation Intel Xeon Phi Product," IEEE Micro, vol. 36, no. 2, pp. 34–46, 2016.

[8] N. Stephens, S. Biles, M. Boettcher, J. Eapen, M. Eyole, G. Gabrielli, M. Horsnell, G. Magklis, A. Martinez, N. Premillieu, A. Reid, A. Rico, and P. Walker, "The ARM Scalable Vector Extension," IEEE Micro, vol. 37, no. 2, pp. 26–39, 2017

[9] Nvidia Corporation, "NVIDIA A100 Tensor Core GPU Architecture." [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf

- *CPUs*: ISSR similar *to scatter-gather*
  - SIMD-based: Xeon Phi KNL[7], Arm SVE[8]
  - CVR SpMV on KNL[1]: 0.7% DP-FP (**70×**)
- *GPUs:* sparsity tackled in SW *and* HW
  - CuSPARSE[2] CsrMV (1080 Ti): 17% FP64 (**2.8×**)
  - A100[9]: only *structured* sparsity (2 in 4)
- *HW Accelerators*: hard to compare
  - Not capable of general-purpose compute
  - DL: various precisions → goal is *accuracy*
  - ➢ Can adopt some *high-level methods* in SW





[9]

Conclusion

- Extended SSR for *indirection* → *ISSR*
- Efficient sparse-dense product kernels
- Evaluated ISSR in *Snitch Cluster*
  - +4.4 kGE or +0.8% in eight-core cluster
  - *Single core*: up to **7.2×** faster, **80%** FPU util.
  - *Cluster*: up to **5.8×** faster, **2.7×** less energy
- Compared ISSR to SoA approaches
  - **2.8×** peak FP64 util. of CuSPARSE[2] (1080 Ti)
  - More flexible than GPU, accelerator HW

Thank You!

Questions?

[1] B. Xie, J. Zhan, X. Liu, W. Gao, Z. Jia, X. He, and L. Zhang, "CVR: efficient vectorization of SpMV on x86 processors," in CGO '18, 2018, pp. 149–162.

[2] Nvidia Corporation, "Cuda Toolkit 10.0." [Online]. Available: https://developer.nvidia.com/cuda-toolkit

[3] F. Schuiki, F. Zaruba, T. Hoefler, and L. Benini, "Stream Semantic Registers: A Lightweight RISC-V ISA Extension Achieving Full Compute Utilization in Single-Issue Cores," IEEE Trans. Comput., pp. 1–1, 2020.

[4] F. Zaruba, F. Schuiki, T. Hoefler, and L. Benini, "Snitch: A tiny pseudo dual-issue processor for area and energy efficient execution of floating-point intensive workloads," IEEE Trans. Comput., pp. 1–1, 2020.

[5] A. Kurth, W. Rönninger, T. Benz, M. Cavalcante, F. Schuiki, F. Zaruba,and L. Benini, "An Open-Source Platform for High-Performance Non-Coherent On-Chip Communication," arXiv:2009.05334 [cs.AR], 2020.

[6] T. A. Davis and Y. Hu, "The University of Florida Sparse Matrix Collection," ACM Trans. Math. Softw., vol. 38, no. 1, Dec. 2011.

[7] A. Sodani, R. Gramunt, J. Corbal, H. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y. Liu, "Knights Landing: Second-Generation Intel Xeon Phi Product," IEEE Micro, vol. 36, no. 2, pp. 34–46, 2016.

[8] N. Stephens, S. Biles, M. Boettcher, J. Eapen, M. Eyole, G. Gabrielli, M. Horsnell, G. Magklis, A. Martinez, N. Premillieu, A. Reid, A. Rico, and P. Walker, "The ARM Scalable Vector Extension," IEEE Micro, vol. 37, no. 2, pp. 26–39, 2017

[9] Nvidia Corporation, "NVIDIA A100 Tensor Core GPU Architecture." [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf