A 7.3 M Output Non-Zeros/J Sparse Matrix-Matrix Multiplication Accelerator using Memory Reconfiguration in 40 nm

Subhankar Pal, Dong-hyeon Park, Siying Feng, Paul Gao†, Jielun Tan, Austin Rovinski, Shaolin Xie†, Chun Zhao†, Aporva Amarnath, Timothy Wesley, Jonathan Beaumont, Kuan-Yu Chen, Chaitali Chakrabarti*, Michael Taylor†, Trevor Mudge, David Blaauw, Hun-Seok Kim, Ronald Dreslinski (Email: subh@umich.edu)
University of Michigan, Ann Arbor, MI †University of Washington, Seattle, WA *Arizona State University, Tempe, AZ

Abstract

A Sparse Matrix-Matrix multiplication (SpMM) accelerator with 48 heterogeneous cores and a reconfigurable memory hierarchy is fabricated in 40 nm CMOS. On-chip memories are reconfigured as scratchpad or cache and interconnected with synthesizable coalescing crossbars for efficient memory access in each phase of the algorithm. The 2.0 mm×2.6 mm chip exhibits 12.6× (8.4×) energy efficiency gain, 11.7× (77.6×) off-chip bandwidth efficiency gain and 17.1× (36.9×) compute density gain against a high-end CPU (GPU) across a diverse set of synthetic and real-world power-law graph based sparse matrices.

Keywords: Sparse matrix multiplier, synthesizable crossbar, decoupled access-execution, reconfigurablility and accelerator

Introduction

SpMM is a fundamental kernel in graph analytics and machine learning, where matrices are typically very large but have low densities, *e.g.* an adjacency matrix of Facebook friendships is 1.08 B×1.08 B with only 0.0003% Non-Zero Elements (NZEs) [1]. SpMM is quintessentially memory-bound rather than compute-bound, due to low data locality and compute-to-communication ratio. Thus, accelerating SpMM requires eliminating redundant memory accesses and maximizing data reuse.

The inner product method (Fig. 1) produces a small Number of Non-Zeros (NNZs) per byte fetched from off-chip due to failed index matches, leading to *unproductive loads*. Limited on-chip storage further forces repetitive fetching of the same data, worsening the memory bottleneck. Prior designs only demonstrate sparse matrix-vector multiplication [2] and relatively high-density (≥3%) matrix-matrix multiplication with small dimensions (≤256) [3]. This paper presents the first custom chip accelerating SpMM that addresses the off-chip memory access bottleneck for real-world sized matrices, evaluating densities ≥0.002% and dimensions ≤120k. It uses an outer product algorithm that we first proposed in [1], where *each* memory fetch generates useful results. A novel synthesizable coalescing crossbar magnifies on-chip bandwidth for accessing reconfigurable memory, attuned for maximum data reuse in the *multiply* phase using shared cache and deterministic accesses in the *merge* phase using scratchpads. Our chip achieves an energy efficiency of 7.3 M output NNZ/J and bandwidth efficiency of 11.7 M output NNZ per GB of fetched data (NNZ/GB).

Approach and Architecture

The outer product approach consists of two phases with disparate dataflow patterns to compute $A \times B = C$ (Fig. 1). In the *multiply* phase, each Processing Element (PE) multiplies an NZE of column i of A with all NZEs of row i of B, generating one Partial Product Matrix (PPM) row, with each NZE fetched only once. PPMs are stored as a set of linked lists of pointers to "chunks" in DRAM, where each list corresponds to one row of C and a chunk holds *(index, value)* pairs of a PPM row. This phase computes multiplications of *all* combinations of fetched elements, resulting in theoretically maximum reuse of inputs *without* index matching, thus circumventing the problem of unproductive loads. The on-chip memory is configured as *shared cache* to enable maximum reuse of NZEs in a row of B between PEs operating on different NZEs in a column of A.

During *merge*, each PE traverses certain PPM rows and merges them into single, sorted lists each corresponding to a row of C. Caches are seamlessly reconfigured into *software-managed scratchpads* to avoid evictions and take advantage of *merge*'s deterministic dataflow. If the chunks of a row of C exceed the on-chip scratchpad capacity, they are merged over multiple iterations with intermediate data being stored in DRAM.

Circuit Implementation

Fig. 4 shows the design consisting of two compute substrates. The first, composed of 32 PEs (4 PEs/tile), computes the *multiply* phase. Each PE has a 32-bit Floating-Point (FP) multiplier and supports out-of-order loads/stores. The second substrate consists of 8 Cortex M0+M4F pairs (1 pair/tile) for *merge*. Each M0/M4F shares the reconfigurable network resources with 4 PEs. The network consists of a fully-synthesizable Swizzle-Switch Network (SSN) crossbar based on [4], with the pull-down networks replaced by OR trees (Fig. 7). This enables fast process migration, especially at advance nodes. The crossbars support request coalescing+multicast (Fig. 2) and Least-Recently Granted (LRG) arbitration (Fig. 3). The downstream L0 crossbar connects to the reconfigurable L0 cache, which provides second-level coalescing. For the *multiply* phase, the L0 is a multi-banked cache, allowing NZEs of B to be shared. For *merge*, it is reconfigured into a multi-banked scratchpad by disabling the tag array and is shared by an M0-M4F pair. Through another set of crossbars, the L0 cache in each tile connects to the L1 layer, which interfaces to the front side bus (FSB).

During the *multiply* phase, the PE manager fetches and distributes work to PEs. In the *merge* phase, the head elements of each PPM row are loaded into the L0 scratchpad by the M0, and sorted in pipeline fashion by the M4F, resulting in *decoupled access-execution*, which is key to maximizing memory-level parallelism for efficient use of off-chip bandwidth. The element with the smallest index is stored to memory and the next element in that PPM row is fetched, until the list is empty. When matching indices are encountered, elements are summed together before they are stored to memory. The M0/M4Fs and PEs are clock-gated during *multiply* and *merge*, respectively.

Measured Results

SpMM was evaluated using matrix squaring on synthetic matrices and power-law graphs, representative of real-world sparse matrices. The 2.0 mm×2.6 mm accelerator achieves 6.1-8.4 M NNZ/J and 6.4-15.5 M NNZ/GB. The chip operates with optimal energy efficiency at (41.7 MHz, 0.860 V) for *multiply* and (352.0 MHz, 0.864 V) for *merge* (Fig. 5). Clock sweeps show that while *multiply* performance hits a roofline, *merge* performance saturates slowly, as *merge* is more compute-heavy. For the bandwidth sweeps, simulation results are appended to measured results considering higher bandwidth and more compute units. The "knee" lines show that *multiply* is ∼30× more sensitive to bandwidth than *merge*. Based on Fig. 5, scaling out our chip to 16× the current configuration would meet the CPU's performance at 9.5× less bandwidth, 16.7× lower power and 0.08× the area, as the chip makes optimal use of available bandwidth by minimizing off-chip traffic. For *merge*, the measured average performance benefit of reconfiguring from L0 cache to scratchpad across the suite of matrices in Fig. 8 is 25.7%. Our SSN crossbar gives the chip a 24.9% performance gain at 86.3% the energy and 1.3% more area over a MUX crossbar based design. Fig. 6 compares against state-of-the-art software libraries on high-end CPU and GPU. The improvements show similar trends, except for power-law graphs, where the rate of runtime increase with increasing NNZ is greater for CPU than GPU.

In summary, our chip achieves measured energy efficiency improvements of 12.6× against a Core i7 CPU and 8.4× over a V100 GPU. A summary and comparison table are given in Tables 1 and 2 and die photo in Fig. 9. Owing to the memory-bound nature of SpMM, *bandwidth efficiency* is considered the paramount metric, for which our design achieves 11.7× and 77.6× improvements over the CPU and GPU, respectively.

References

[1] S. Pal *et al.*, *HPCA*, pp. 724-736, February, 2018.

[2] R. Dorrance and D. Marković, *VLSI*, pp. 1-2, June 2016.

[3] M. Anders *et al.*, *ISSCC*, pp. 39-40, June 2018.

[4] S. Satpathy *et al.*, *ISSCC*, pp. 478-480, February 2012.

[5] F. Khorasani *et al.*, PACT, pp. 39-50, October 2015.

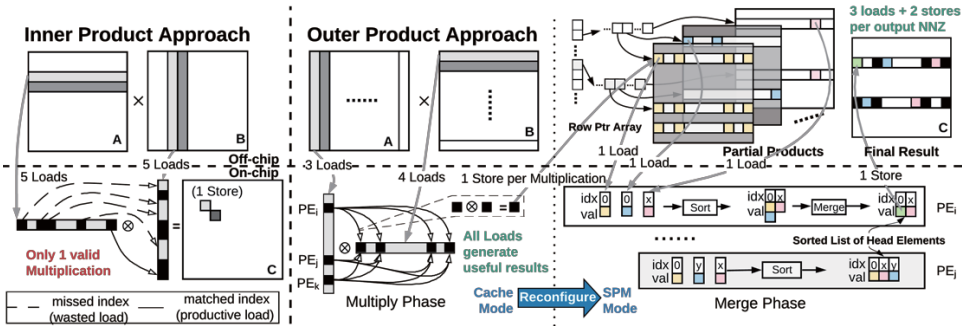


Fig 1. Inner and outer product SpMM algorithms. Outer product involves zero wasted loads/index comparisons.

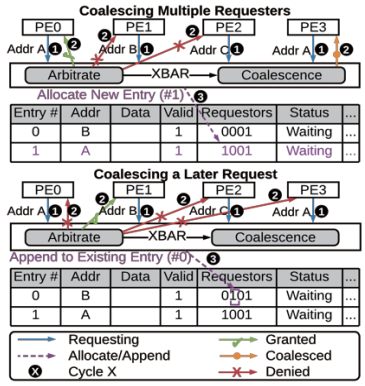


Fig 2. Crossbar and cache coalescence.

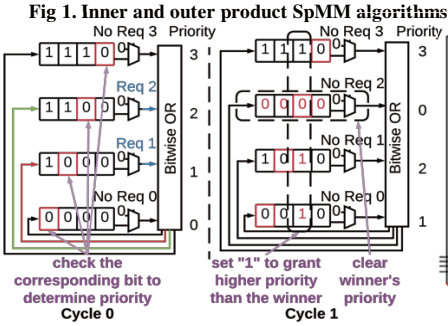


Fig 3. Least Recently Granted (LRG) Scheme.

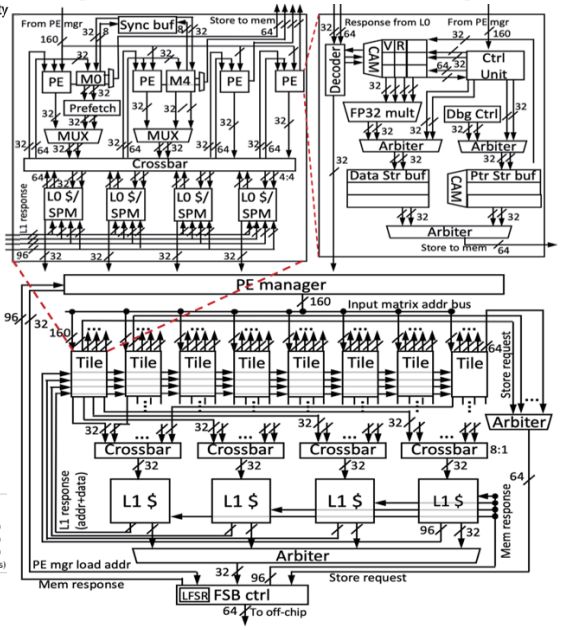
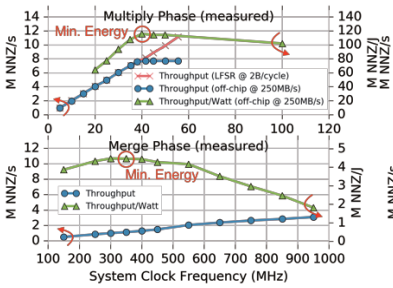


Fig 4. Microarchitectural diagram of top level, a tile and a PE.



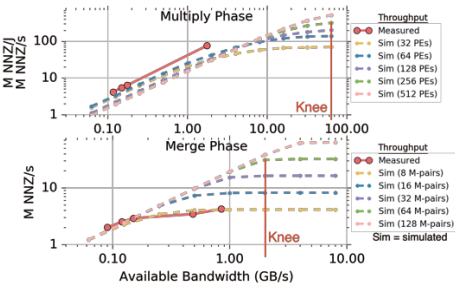


Fig 5. Clock and bandwidth sweeps for matrix dim. 100k, density 0.0008%. For measurements with increased bandwidth, on-chip LFSR is used for multiply and M0 is used for merge.

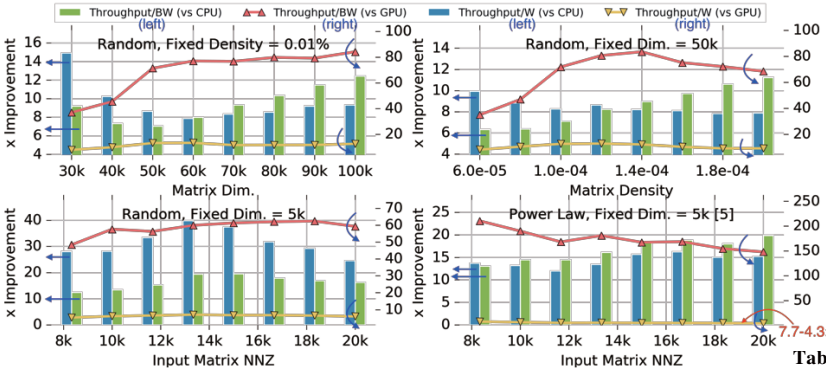


Fig 6. Measured results over different matrices showing energy and bandwidth efficiency of the proposed chip normalized to Core i7 CPU and V100 GPU running SpMM packages.

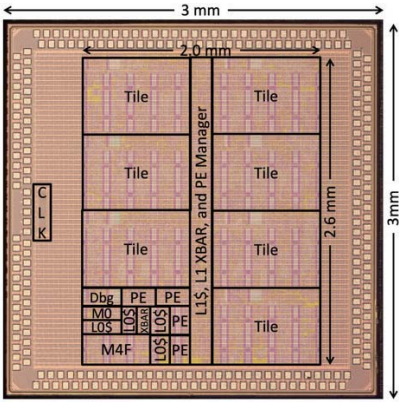


Fig 9. Annotated die photo with GDS overlay.

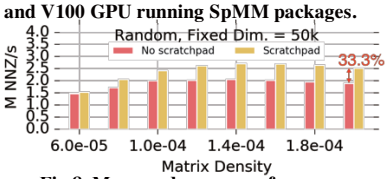


Fig 8. Measured merge performance.

Table 1. Chip characterization summary.

Technology	40 nm CMOS
Die Size	3.0 mm × 3.0 mm
Block Size	2.0 mm × 2.6 mm
# Transistors	25,134,927
Total SRAM	112 KB
Data Precision	Single-Precision Floating Point
Nominal Frequency (Minimum Energy)	41.7 MHz @ 0.860 V (Multiply) 352.0 MHz @ 0.864 V (Merge)
Maximum Frequency	950.0 MHz @ 1.27 V
Nominal Power Consumption	66.6 mW (Multiply) 226.0 mW (Merge)

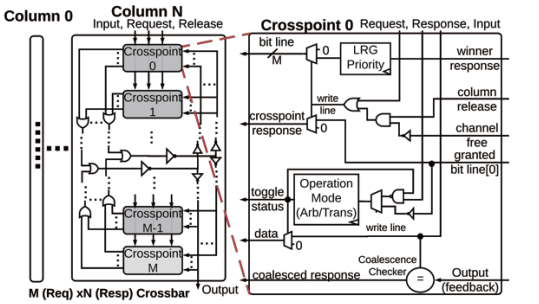


Fig 7. Crossbar schematic showing crosspoints and OR tree.

Table 2. Key metrics and comparison vs. CPU/GPU and prior work.

System / Feature	Intel Core i7-6700K (MKL)	NVIDIA Tesla V100 (CUSP)	DSP [2] *	ASIC [3] *	This work
Kernel	SpMM	SpMM	SpMV†	DMM^	SpMM
Max. Matrix Dimension	120,000	120,000	217,918	256	120,000
Min. Matrix Density	0.002%	0.002%	0.003%	3%	0.002%
Reconfigurability	x	x	x	✓	✓
Process (nm)	14	12	40	14	40
Core Count	8	5120	4	16	48
Total Core Area (mm^2)	122.00	815.00	0.93	0.02	5.20
Frequency (MHz)	4000	1250	515	800	744
Off-Chip Memory Bandwidth (GB/s)	34.10	900.00	3.20 ~ 8.53	N/A (on-chip only)	0.24
Power (W)	58.84	123.95	0.06	0.04	0.25
Compute Density (NNZ/s/mm^2) [×10^6]**	0.0279	0.0129	9.0183	N/A	0.4775
Energy Efficiency (NNZ/J) [×10^6]	0.58	0.87	129.95	N/A	7.28
Bandwidth Efficiency (NNZ/GB) [×10^6]	**1.00**	**0.15**	**0.98 ~ 2.61**	**N/A**	**11.73**

† Sparse Matrix-Vector Multiplication
^ Dense Matrix-Matrix Multiplication
* not directly comparable to this work
** area normalized to 40 nm technology