A Sparse Matrix-Matrix multiplication (SpMM) accelerator with 48 heterogeneous cores and a reconfigurable memory hierarchy is fabricated on a 40 nm CMOS. On-chip memories are configured 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 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, reconfigurability and accelerator

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 1.08B × 1.08B 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, resulting in high bandwidth inefficiency. In contrast, the multiply phase fetches and merges pointers to “chunks” in DRAM, where each list corresponds to a row of A with NZEs and is stored in a column of B. When matching indices are encountered, elements are summed and sorted in pipeline fashion by the M4F, resulting in theoretically maximum reuse of NZEs in a row of B. Caches are seamlessly reconfigured from L0 cache to scratchpad across the suite of matrices in Fig. 8. Our design achieves 11.7× measured energy efficiency gain at 86.3% the energy and 1.3% more area over a MUX crossbar based design. The “knee” lines show that when 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. On high-end CPU and GPU, the improvements show similar trends, except for power-law graphs, where the rate of runtime improvement decreases with increasing NNZ.

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 determinisic 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.

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 in Arecibo's report.
References


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.

Table 1. Chip characterization summary.

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

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 7. Crossbar schematic showing crosspoints and OR tree.

Table 3. Chip characteristics summary.

Fig 8. Measured merge performance.