Accelerators for Sparse Matrix Multiplication

Gerardo Hernandez Macoto, Prof. Xiaochen Guo,

Department of Electrical and Computer Engineering

Introduction

Sparse matrices are defined as matrices that contain mostly zero values. Skipping zeros allow the matrices to be stored with smaller memory footprint and to be computed for matrix multiplication more efficiently.
Research Problem: how to design architectures to compute sparse matrix multiplication that have a high processing element (PE) utilization.
Other spMspM accelerators with different architectures and algorithms have been proposed.
The Sparse TPU (STPU) [1] paper proposed three algorithms for packing greedy packing, partition-wise packing, Collision-aware packing.
SIGMA [2] is an architecture that offers high utilization for its PEs regardless of its kernel shape and sparsity, and it's a general matrix-matrix multiplication (GEMM) accelerator.
Systolic array is a hardware structure built for fast and efficient computation of dense matrix multiplication, which was adapted for STPU [1] and compute fabric for accelerating sparse GEMMs [2].
The algorithm proposed allows packing to be applied to any intermediate matrices with lower overhead.
The purpose of this research is to create an efficient spMspM Accelerator that can handle sparsity, help reduce data movement, decrease the total cycle runtime, and reduce energy consumption.

Method

Our algorithm was designed by Hesam, and it sorts columns of one matrix and sorts rows of another matrix based on the number of non-zero elements in each of the two matrices, then the sorted columns and rows are packed into compressed rows and columns. This purpose of sorting before packing is to keep the non-zero elements in a row or column close to each other after packing, which results in faster combination of partial sums.
What I had done was to measure the speed up of the sorting algorithm and compared it with the Collision-aware packing algorithm in STPU. The packing runtime is important because every matrix that needs to be computed requires this preprocessing step. Understand the latency of the packing algorithm gives us insight on whether this algorithm can be deployed online or not.

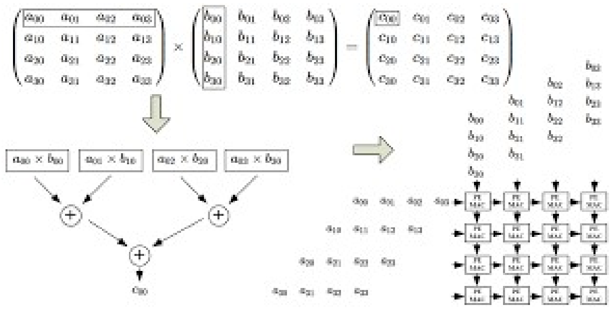


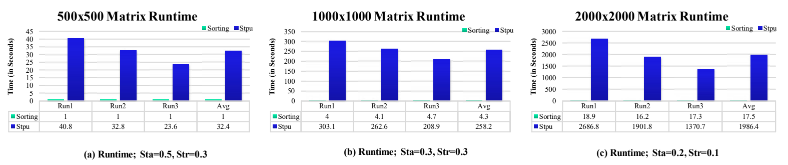
Figure 1: Systolic array [3] algorithm are used for matrix-matrix multiplication; the output matrix is computed by having one row from top to bottom being computed with the other matrix column from the left to the right side.

Discussion

The runtime was taken into consideration and a time function was incorporated to test the runtime for the sorting and stpu algorithm to compare which algorithm was faster.
Our design significantly outperformed the STPU algorithm.
The larger the sparsity, the longer the runtime according to the results from the total runtime for sorting based design and STPU.
Performance was the same across every run. STPU and Systolic array resulted in the same performance result (total cycle runtime). Our design and SIGMA had different results in comparison to STPU and Systolic array, but our architecture resulted in the lowest total cycle runtime throughout every run.

Results

Each matrix contains three runs, each run containing different sparsity values. The average was calculated for Fig. 2.
The results provided in Fig. 2 compares the Collision-aware algorithm with our algorithm.
The performance/total cycle runtime for comparison of our design and different architectures are provided in Fig. 3-5.



	Run1	Run2	Run3	Avg
Sorting	1	1	1	1
Stpu	40.8	32.8	23.6	32.4

	Run1	Run2	Run3	Avg
Sorting	4	4.1	4.7	4.3
Stpu	303.1	262.6	208.9	258.2

	Run1	Run2	Run3	Avg
Sorting	18.9	16.2	17.3	17.5
Stpu	2686.8	1901.8	1370.7	1986.4

Figure 2: Results of total runtime between Collision-aware algorithm and our algorithm with different sparsity size.

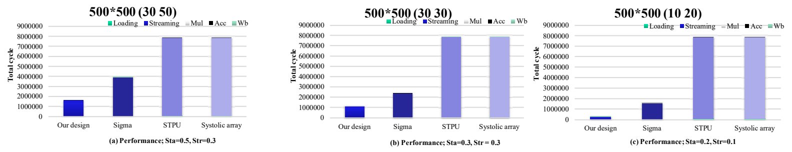


Figure 3: Results from the total cycle runtime (performance) for 500x500 Matrix of our architecture in comparison to SIGMA, STPU, and Systolic array.

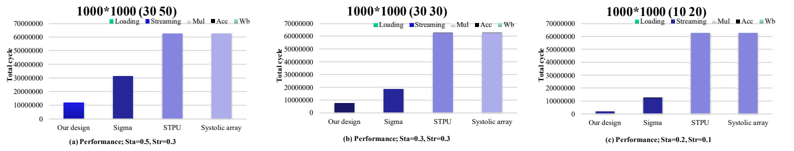


Figure 4: Results from the total cycle runtime (performance) for 1000x1000 Matrix of our architecture in comparison to SIGMA, STPU, and Systolic array.

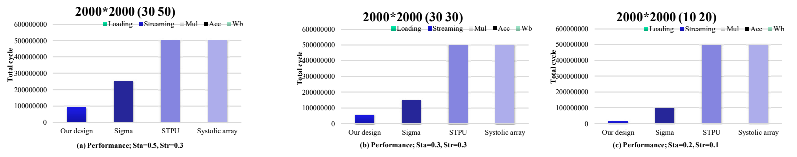


Figure 5: Results from the total cycle runtime (performance) for 2000x2000 Matrix of our architecture in comparison to SIGMA, STPU, and Systolic array.

Conclusion

The sorting algorithm outperformed STPU's collision-aware packing algorithm along with the architecture outperforming different architectures (STPU, SIGMA, and Systolic array) in terms of runtime and performance. Performance of architecture design accomplishes goal of reducing the total cycle runtime and algorithm accomplishes reducing energy consumption. The architecture design is a work in progress; overall, the algorithm and architecture has a promising outcome in achieving purpose of research.

References

[1] E. Qin et al., "SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training," 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2020.

[2] He, Xin et al. "Sparse-TPU: adapting systolic arrays for sparse matrices." Proceedings of the 34th ACM International Conference on Supercomputing (2020): n. pag.

[3] Yang, Zhijie et al. "Systolic Array Based Accelerator and Algorithm Mapping for Deep Learning Algorithms." NPC (2018).

Acknowledgement(s):

Thank you to Professor Xiaochen Guo, Hesam Eddin Shbani, and Shuo Yin

LEHIGH UNIVERSITY