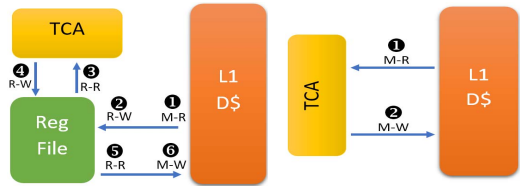
2022 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES) | 978-1-6654-7296-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/CASES55004.2022.00028

Work-in-Progress: NoRF: A Case Against Register File Operands in Tightly-Coupled Accelerators

David J. Schlais, Heng Zhuo, Mikko H. Lipasti
University of Wisconsin-Madison
schlais2@wisc.edu, hzhuo2@wisc.edu, mikko@engr.wisc.edu



(a) TCA integrated with RF (b) TCA integrated with NoRF

Fig. 1: Dataflow for a tightly-coupled accelerator (TCA) that uses RF vs. NoRF. Without reuse in the RF, RF requires many more steps of data movement external to the TCA.

I. Introduction and Motivation

Accelerators are often used to increase performance and/or energy efficiency of general-purpose CPUs. However, Tightly-Coupled Accelerators (TCAs) often perform computations on data structures that may not be a natural fit for general-purpose registers. The designer can either use the existing register file (RF), a RF tailored for the accelerator, or eschew a RF entirely (NoRF), accessing operands directly from the memory hierarchy. Designers for embedded and edge devices are particularly conscientious towards energy-efficient compute and data transfer. We explore the possibility of mini-DGEMM accelerators (example TCAs) within the context of CPUs and edge devices, which also have increasing applications for DGEMM compute. At a high level, register files help reduce memory accesses (steps 1, 2, 5, and 6 in Figure 1) when the compiler finds reuse of operands in the program dataflow. On the other hand, direct memory access simplifies the data movement by completely eliminating the intermediate reads and writes to a register file but issues more memory requests. This paper evaluates the difference between these options of operand delivery. Figure 2 shows that all recent vector extensions use a register file implementation. By this trend, it may seem natural to incorporate mini-matrices into the RF. However, we present quantitative and qualitative evidence to advocate for direct cache access for operands.

II. Related Work

Google's TPU [6] is a large-scale DNN accelerator for CNN/RNN inference that uses a systolic two-dimensional matrix-matrix multiplication array. DaDianNao [3] is a customized chip that uses both custom compute and storage to gain large speedups. EyeRiss [2] is a reconfigurable accelerator aimed at speeding up CNN inference computation (matrix multiplication) through optimized data movement through a row stationary processing dataflow on a spatial architecture. Compared to these accelerators, which perform matrix operations orders of magnitude larger, we look at how the architecture should change when implementing fine-grained TCAs. The MANIC architecture [4] reduces RF reads and writes of intermediate and dead registers by forwarding intermediate results, but all data from memory is still required to be inserted into the RF, whereas NoRF completely eschews the RF. Both coarse- and fine-grained matrix operations can be beneficial for different types of use case applications, where smaller matrix operations may have additional use in edge and embedded devices.

This work was supported in part by the National Science Foundation under grants CCF-1615014, CCF-1628384, and CCF-1813434.

Operand	Year	Bits	RF	Memory
Burroughs Scientific Processor	1982	48 (17 banks)	No	Yes
X86 SSE	2000	128	Yes	No
ARM Neon SIMD	2008	128	Yes	No
AVX-256	2011	256	Yes	No
AVX-512	2013	512	Yes	No
TensorCore 16FP	2017	256 (4x4x16)	Yes	No
TensorCore 64FP	2020	512 (4x2x64)	Yes	No
TCA:RF	N/A	512	Yes	No
TCA: NoRF	N/A	512	No	Yes

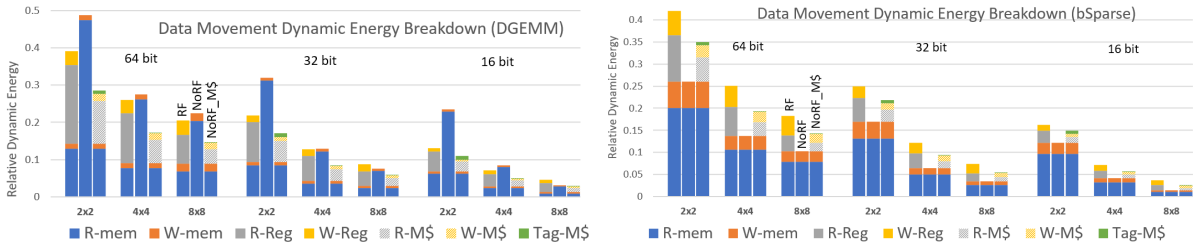
Fig. 2: Evolution of adding wide vector operations

III. TCAs with RF vs NoRF Implementations

One TCA implementation passes operands through a RF. When there is temporal reuse, RFs reduce the number of requests to the memory hierarchy. However, without reuse, RFs cause unnecessary data movement reads and writes. Reuse can be made through optimizing the software/compiler to tile algorithms to reuse registers before accessing memory again. Concurrent operations can be made through the use of vector, instead of scalar, registers. A TCA (doing matrix operations, for example) could additionally specialize a register file to have each register hold an entire 'unit' worth of data, such as an entire 8x8 submatrix. We assume an RF that holds submatrices as elements, with an optimized compiler for reuse, as our baseline.

Through diagrams and intuition, it is unclear whether or not the register file is actually providing benefit to the TCA for data movement. Having a register file as an intermediate changes a datapath of the L1D$ to the TCA from 2 total reads/writes to 6 total reads/writes (Figure 1). If a register file element is read and immediately removed, it seems beneficial

2643-1726/22/$31.00 ©2022 IEEE
DOI 10.1109/CASES55004.2022.00028



(a) DGEMM (b) SpGEMM

Fig. 3: Energy breakdown of data transfer in both DGEMM and SpGEMM algorithms for RF (R), NoRF (N), and NoRF with M$ (M$) TCA implementations varied with different bit widths (64, 32, 16) and different TCA sizes (2x2, 4x4, 8x8).

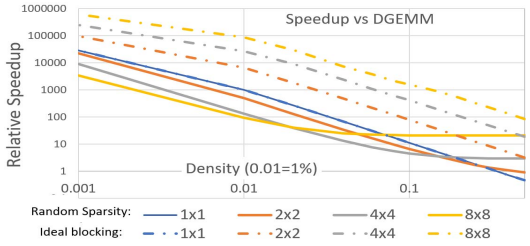


Fig. 4: Block-sparse SGEMM performance with random- (solid line) and ideally-blocked (dotted line) matrices.

to completely bypass the register file. This implementation of using **No R**egister **F**ile can be called a **NoRF** implementation. A potential downside of eliminating the register file is less operand reuse. However, TCAs that buffer loaded operands (which we deem a matrix cache, or M$), capture reuse in memory-to-memory operations in the same way that reuse can be done through the register file. At first it may seem like buffering data causes similar overheads to the RF, but by having a known datapath and access patterns (ie, having different banks for A, B, and C matrix elements), this implementation eliminates the multi-ported nature of the RF.

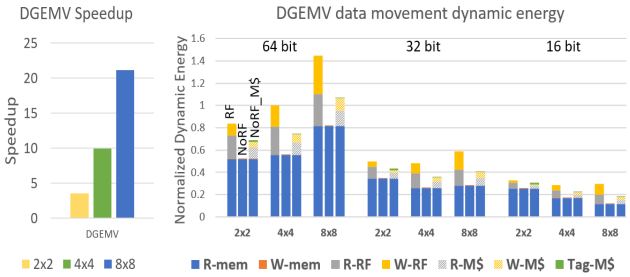
IV. Results

We use CACTI [7] to estimate the energy associated for L1 cache access, at 40nm with a design focus on low power.

In order to test execution performance of the TCA design with a cycle-accurate simulator, we use gem5 [1]. We configure gem5 to mimic Intel's Sunnycove [8] architecture. We then use Eigen [5] as an existing software library designed to exploit SIMD and vector registers as a SW alternative. We manually insert the specialized TCA instruction as well as specialized RF load/store instructions for comparison.

Speedup from Eigen is limited (2x-4x improvement) by the width of vector registers and the number of floating point execution units. As expected, custom TCAs which supply additional floating point MACs and custom logic to supply those MACs see additional speedup (5x-80x in 2x2-8x8 TCAs), with significant speedups for the larger DGEMM TCAs, as the work done each invocation grows by a factor of O(n^3), reducing control flow operation and utilizing larger vector loads (graph removed in WiP paper due to space limitations).

Sparse matrices can still utilize the TCAs proposed by utilizing a blocked-sparse format. We test both ideal case



(a) DGEMV TCA performance (b) DGEMV TCA energy breakdown

Fig. 5: DGEMV performance and energy consumption.

(where all non-zero elements are fully packed into TCA blocks), as well as a pessimistic, randomly distributed sparse matrix (no local density). Figure 4 shows that for very sparse matrices, the highest speedups are achieved, since limited calculations are required. Interestingly enough, however, when non-ideal packing is involved, the 1x1 (traditional CSR algorithm) surpasses each of the TCA execution times at a specific crossover point for each TCA size.

Figure 3a demonstrates up to a 39% reduction in energy consumption of data movement by completely eliminating the register file, and instead implementing a specialized TCA datapath with a matrix cache. Fig. 3a and Fig. 3b show that the larger the TCA (8x8), the more inherent energy reduction there is in data movement. This is because the number of partial producs grows $O(n^3)$. In other words, the more work the accelerator can do per access, the smaller the inefficiencies in data movement. In blocked sparse matrix multiplication (SpGEMM), the output element partial products are calculated out of order. For this reason, disabling the M$ provides the best energy wins, up to 60% over the RF design.

The main takeaway from Figure 5 is that the M$ does not capture reuse in the DGEMV algorithm, and maximum energy reduction occurs by using NoRF without the M$ enabled. The DGEMV geomean energy reduction for NoRF with the M$ is 21%, and NoRF without M$ has a geomean energy reduction of 41%. All around, these advantages build a case for NoRF-based implementations in these TCAs, which goes against the long-lived tradition of register files for operand delivery in compute within the core.

REFERENCES

[1] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[2] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[3] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun *et al.*, "Dadiannao: A machine-learning supercomputer," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2014, pp. 609–622.

[4] G. Gobieski, A. Nagi, N. Serafin, M. M. Isgenc, N. Beckmann, and B. Lucia, "Manic: A vector-dataflow architecture for ultra-low-power embedded systems," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2019, pp. 670–684.

[5] G. Guennebaud, B. Jacob *et al.*, "Eigen: a c++ linear algebra library," *URL http://eigen. tuxfamily. org, Accessed*, vol. 22, 2014.

[6] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," *arXiv preprint arXiv:1704.04760*, 2017.

[7] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, "Cacti 5.1," Technical Report HPL-2008-20, HP Labs, Tech. Rep., 2008.

[8] Wikichip, "Sunny cove - microarchitectures," 2019, last accessed 25 November 2019. [Online]. Available: https://en.wikichip.org/wiki/intel/microarchitectures/sunny$_cove$