

Entropy Maximization in Sparse Matrix by Vector Multiplication ($\max_E \text{SpMV}$)

ABHISHEK JAIN, ISMAIL BUSTANY, HENRI FRAISSE, MANSIMRAN BENIPAL, DINESH GAITONDE, and PAOLO D'ALBERTO, Xilinx

The peak performance of any SpMV depends primarily on the available memory bandwidth and the capability to use it effectively. GPUs, ASICs, and new FPGAs have higher and higher bandwidth; however, for large scale and highly sparse matrices we find still difficult utilizing this bandwidth because the SpMV random access pattern and workload imbalance. We propose a matrix permutation pre-processing step that aims to maximize the entropy of the distribution of the nonzero elements. We seek any permutation that uniformly distributes the non-zero elements' distribution, thereby generating a SpMV problem that is amenable to work load balancing or to speed up sort algorithms. We conjecture these permutations would be most effective for matrices with no dense rows or columns and, as in preconditioning, when the matrix is reused. We shall show that entropy maximization is an optimization that any architecture may take advantage although in different ways. Most importantly, any developer can consider and deploy. We shall present cases where we can improve performance by 15% on AMD-based systems.

ACM Reference Format:

Abhishek Jain, Ismail Bustany, Henri Fraisse, Mansimran Benipal, Dinesh Gaitonde, and Paolo D'Alberto. 2020. Entropy Maximization in Sparse Matrix by Vector Multiplication ($\max_E \text{SpMV}$). 1, 1 (July 2020), 27 pages.

1 INTRODUCTION

To define the scope of this work, the obvious questions to ask are: first, what randomization or entropy maximization is in the context of sparse matrices; second, why would we use it; third, when it does work. We shall provide formal definitions in the following sections. Briefly, we will permute randomly the rows and columns of a sparse matrix before multiplying it with a dense vector (SpMV) with the aim of speeding this operation. Undoubtedly, this scheme requires some restrictions about the matrix structure, one among them is that it has no or few dense columns or rows. In the case, where there are dense columns or rows, a sparse/dense partitioning scheme should be used. For the remainder of this manuscript, we shall assume the former nonzero structure. We use randomization because it is the poor man's way for preconditioning SpMV in our context, and we do not mean it in a pejorative sense.

Preconditioning speeds up the convergence rate of an iterative linear solver by linearly transforming the associated matrix into a form that affords a faster reduction of the residual error at every iteration. The cost of this transformation is justified by the runtime reduction it affords. Likewise, we foresee randomization playing a similar role for SpMV in the context of iterative linear solvers and other methods (e.g. in convolutions) where the matrix is reused.

Sparse linear algebra and GraphBLAS kernels are memory bound and there is a common thread in the scientific computing community to develop acceleration libraries mostly for multi-core systems. These predominantly include multi-core processors and GPUs. The goal is a balanced work distribution and, when applicable, minimal communication

Authors' address: Abhishek Jain, abhishek@xilinx.com; Ismail Bustany, ismailb@xilinx.com; Henri Fraisse, henrif@xilinx.com; Mansimran Benipal, mansimra@xilinx.com; Dinesh Gaitonde, dineshg@xilinx.com; Paolo D'Alberto, paolod@xilinx.com, Xilinx, 2100 Logic Dr, San Jose, California, 95124.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Association for Computing Machinery.

Manuscript submitted to ACM

[6, 11]. When storage strategy and algorithms must be considered together then GPUs provide the work horse for abundant thrust in research [1]. These works aim at optimal solutions and strive for a clear and complete understanding/exploitation of the software-hardware interface; usually the hardware is composed of symmetric computational units. Interestingly, the SpMV's space and time complexity, which are small, may not warrant more performance because we typically end up utilizing only one-thousandth fraction of the available hardware capacity.

The peak performance of any SpMV accelerator depends primarily on the available memory bandwidth (i.e., DRAM such as DDR or HBM) and the capability of the accelerator to effectively use it. Because SpMV is memory-bound, a more important metric than peak performance alone is the fraction of bandwidth utilized, which captures the overall efficiency of the architecture. GPU platforms exhibit very high bandwidth, see the experimental Section 8: Ellesmere DDR5 224GB/s, Fiji HBM 512GB/s, and Vega 20 HBM 1TB/s. Although utilizing this much bandwidth efficiently is difficult for large scale and highly sparse matrices due to very high random access pattern. Custom architectures based on FPGA or ASIC devices can maximize bandwidth utilization by highly customized data-paths and memory hierarchy designs [3, 4, 13]. Most of the existing accelerators saturate the relatively low memory bandwidth available on FPGA platforms (less than 80 GB/s) [3, 4, 8, 10, 12, 13]. Modern FPGA platforms have multiple HBM stacks to provide large memory bandwidth. However, there is no implementation (currently available) that saturates all of the available DRAM bandwidth for SpMV kernel on HBM-enabled FPGA platforms. Scalability of accelerator design remains a major concern, and it is an active area of research.

FPGA platforms used in early works exhibit low peak performance due to the scarcity of external memory bandwidth [3, 7, 14]. For example, Microsoft's implementation of SpMV uses an FPGA platform which only has 2 DDR2-400 memory banks with a resulting bandwidth of 6.4 GB/s [7]. The accelerator is running at 100 MHz, it reads 64 Bytes of data every cycle, which corresponds to 5 non-zeros at every cycle (a non-zero is about 12 Bytes). At best, the peak performance is 10 double precision operations every cycle at 100 MHz, which is 1 GFLOPS (only). In 2009, Convey systems Inc. released the Convey HC-1 FPGA platform. It has 16 DDR2-677 memories resulting in overall 80 GB/s memory bandwidth [10]. The accelerator logic runs at 150 MHz. It consumes 512 Bytes of data every cycle, which corresponds to around 40 non-zeros every cycle. At best, the peak performance is 80 double precision operations every cycle at 150 MHz, which is 12 GFLOPS.

One of the key building blocks for custom architecture solutions is a multi-ported buffer used to storing vector entries [3]. During execution, multiple column indices are used as addresses to read corresponding vector entries; we shall provide more details about the application in Section 2. Designing a buffer with a very large number of read ports is challenging. One solution is *banking* as a mechanism to store partitioned vector entries. Although banking could allow very high throughput indexing unless the same entry is required multiple times and its reads are purely sequential causing loss of bandwidth. For example, hashing techniques and data duplication are possible solutions for this problem. However, another issue arises: When we distribute SpMV computations across p -nodes, some of the nodes, say k , finish later than the rest because of unbalanced work loads (i.e., number of nonzero element) in row/column major traversal. This is a common phenomena for matrices where few rows or columns are dense. These k nodes are referred to as *laggard nodes*. By applying random permutation of columns/rows, we are attempting to balance the loads across all p workers so that there is no laggards. From this hardware vantage point, randomization or maximizing the entropy of the non-zero element distribution is an optimization transform and provides a clear context for our work.

Clearly, optimally accelerating SpMV is a hard many-parameters optimization problem dependent on the choice of algorithm, data structures, and dedicated hardware (CPU, GPUs, FPGA's, Custom ASIC's). Rather, our goal is to provide a tool, we may say a naive tool, to help understand how the structure of the matrix may impact the HW-SW

solution. For the readers in the field of algorithms, SpMV can be mapped into a sorting algorithm. For example, finding elements $x_{i,j}$ and $x_{i,k>j}$ in a sparse matrix requires to find row i and then columns j and k . Sorting is a method to find if an element is in a list with no prior or limited knowledge of its contents. Sorting can be used to prepare the matrix and to find elements in between sparse matrices and sparse vectors. In custom architectures, sorting networks are used to route matrix and vector elements to functional units. In a sense, if one is stuck with a sorting algorithm and a poor distribution, randomization may alter the distribution and throttle performance. Interestingly, the best sorting algorithm is a function of the distribution of the elements [5, 9].

We organize this work as follows: In Section 2, we define the matrix by vector operation; in Section 3, we define what we mean by randomization or entropy maximization. We use randomization to create a uniform distribution in Section 5 and measure uniformity by entropy in Section 4. We present how we drive our experiments to show the effects of randomization in Section 6. In the last sections, we present a summary of the results: we present our task work loads for the given benchmarks in Section 7, and the complete set of measures for an AMD CPU and GPUs systems in Section 8.

2 BASIC NOTATIONS

Let us start by describing the basic notations so we can clear the obvious (or not). A Sparse-matrix by vector multiplication SpMV on an (semi) ring based on the operations $(+, *)$ is defined as $\mathbf{y} = \mathbb{M}\mathbf{x}$ so that $y_i = \sum_j M_{i,j} * x_j$ where $M_{i,j}=0$ are not represented nor stored. Most of the experimental results in Section 8 are based on the classic addition $(+)$ and multiplication $(*)$ in floating point precision using 64 bits (i.e., double floating point precision) albeit are extensible to other semi-rings. For instance, it is well known that SpMV defined on the semi-ring $(\min, +)$ is a kernel in computing an all-pairs shortest paths starting with a graph adjacency matrix, and in using a Boolean algebra we can check if two nodes are connected, which is slightly simpler.

We identify a sparse matrix \mathbb{M} of size $M \times N$ as having $O(M + N)$ non-zero elements, number of non zero nnz . Thus the complexity of $\mathbb{M}\mathbf{x}$ is $O(M + N) \approx 2nnz$. Also, we must read at least nnz elements and thus the complexity is $\Theta(M + N) \approx nnz$. We can appreciate that reading the data is as complex as the overall operation. Of course, the definition of sparsity may vary. We represent the matrix \mathbb{M} by using the coordinate list COO or and the compressed sparse row CSR¹ formats. The COO represents the non-zero of a matrix by a triplet (i, j, v) ; very often there are three identical-in-size vectors for the ROW, COLUMN, and VALUE. The COO format takes $3 \times nnz$ space and two consecutive elements in the value array are not bound to be neither in the same row nor column. In fact, we know only that $VALUE[i] = M_{ROW[i], COLUMN[i]}$.

The CSR format stores elements in the same row and with increasing column values consecutively. There are three arrays V, COL, and ROW. The ROW is sorted in increasing order. Its size is M , and $ROW[i]$ is an index in V and COL describing where i -th row starts (i.e., if row i exists). Accordingly, $M_{i,*}$ is stored in $V[ROW[i] : ROW[i + 1]]$. The column indices are stored at $COL[ROW[i] : ROW[i + 1]]$ and sorted increasingly. The CSR format takes $2 \times nnz + M$ space and a row vector of the matrix can be found in $O(1)$.

The computation $y_i = \sum_j M_{i,j} * x_j$ is a sequence of scalar products and, using the CSR format, is computed as follows:

$$Index = ROW[i] : ROW[i + 1]$$

$$y_i = \sum_{\ell \in Index} V[\ell] * x_{COL[\ell]}$$

¹a.k.a. Compressed row storage CRS.

The matrix row is contiguous (in memory) and rows are stored in increasing order. However, the access of the dense vector \mathbf{x} has no particular pattern, well increasing.

The COO format can be endowed with certain properties. For example, we can sort the array by row and add row information to achieve the same properties of CSR. In contrast, transposing a "sorted" COO matrix simply entails swapping of the arrays ROW and COL. Think about matrix multiply (one of us does constantly). Each scalar product achieves peak performance if the reads of the vector \mathbf{x} are streamlined as much as possible and so the reads of the vector V . If we have multiple cores, each could compute a subset of the y_i and a clean data load balancing can go a long way. If we have few functional units, we would like to have a constant stream of independent $*$ and $+$ operations but with data already in registers. That is, data pre-fetch will go a long way especially for $x_{COL[i]}$, which may have an irregular pattern.

3 RANDOMIZATION AND ENTROPY MAXIMIZATION

We define *Randomization* as row or column permutation transform of the matrix \mathbb{M} (thus a permutation of \mathbf{y} and \mathbf{x}), and we choose these by a pseudo-random process. The obvious question to ask is why should we seek randomization transform? The sparsity of a given matrix \mathbb{M} has a non-zero element distribution induced by the nature of the original problem or by some imposed ordering on the respective nodes of its associated graph. This distribution may be computationally incompatible with the chosen algorithm or architecture. For instance, it can induce some load imbalance in the computation. We could break this load imbalance by seeking to maximize entropy for this distribution. Our conjecture is that would favor the average case performance rather than the worse case when operating on the "max-entropy transformed" matrix.

For linear system solvers, if we know the matrix \mathbb{M} , and we know the architecture, preconditioning (when affordable) is a better solution. If we run experiments long enough, we choose the best permutation(s) for the architecture, permute \mathbb{M} , and go on testing the next. On one end, preconditioning exerts a full understanding of both the matrix (the problem) and how the final solution will be computed (architecture). On the other end, the simplicity of a random permutation requires no information about the matrix, the vector, and the architecture. Such a simplicity can be exploited directly in Hardware. We are after an understanding when randomization is just enough: We seek to let the hardware do its best with the least effort, or at least with the appearance to be effortless.

Interestingly, this work stems from a sincere surprise about randomization efficacy and its application on custom SpMV. Here, we wish to study this problem systematically so that to help future hardware designs. Intuitively, if we can achieve a uniform distribution of the rows of matrix \mathbb{M} we can have provable expectation of its load balancing across multiple cores. If we have a uniform distribution of accesses on \mathbf{x} we could exploit column load balancing and exploit better sorting algorithms: In practice, the reading of $x_{COL[i]}$ can be reduced to a sorting, and there we know that different sparsity may require different algorithms. This may be lot to unpack but it translates to a better performance of the sequential algorithm without changing the algorithm or to improved bandwidth utilization.

We will show that (different) randomness affects architectures and algorithms differently, making randomization a suitable optimization transform especially when the application and hardware are at odds: Hardware (unless programmable) is difficult to change and the matrix sparsity is simple to change. We want to show that there is a randomness hierarchy that we can distinguish as global and local. There are simple-to-find cases where the sparsity breaks randomness optimization. For instance, matrices with dense rows or columns are better partitioned into sparse and dense components and operated on separately.

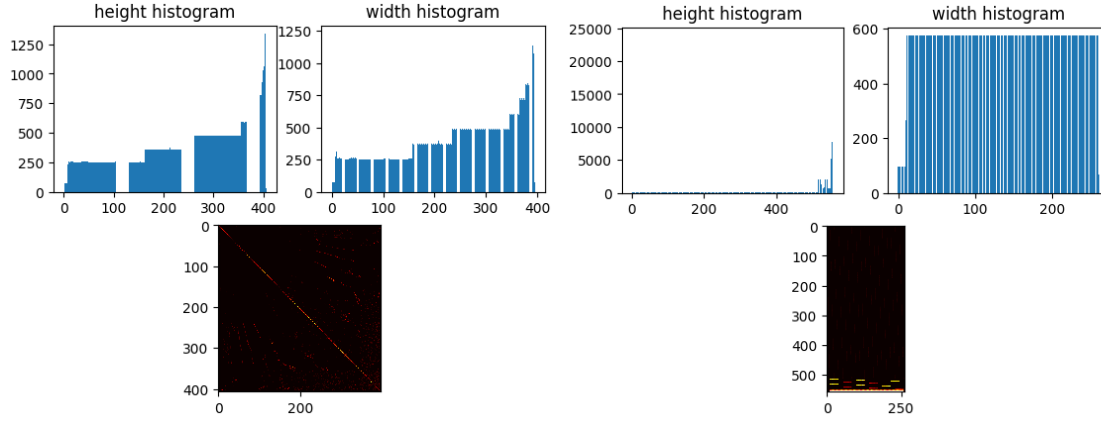


Fig. 1. Left: OPF 3754. Right: LP OSA 07. These are histograms where we represent normalized buckets and counts

4 ENTROPY

Patterns in sparse matrices are often visually pleasing, see Figure 1 where we present the height histogram, the width histograms, and a two-dimensional histogram as heat map. We will let someone else using AI picture classification. Intuitively, we would like to express a measure of uniform distribution and here we apply the basics: *Entropy*. Given an histogram $i \in [0, M - 1]$ $h_i \in \mathbb{N}$, we define $S = \sum_{i=0}^{M-1} h_i$ and thus we have a probability distribution function $p_i = \frac{h_i}{S}$. The *information* of bin i is defined as $I(i) = -\log_2 p_i$. If we say that the stochastic variable X has PDF p_i than the entropy of X is defined as.

$$H(x) = - \sum_{i=0}^{M-1} p_i \log_2 p_i = \sum_{i=0}^{M-1} p_i I(i) = E[I_x] \quad (1)$$

The maximum entropy is when $\forall i, p_i = p = \frac{1}{M}$; that is, we are observing a uniform distributed event. Our randomization should aim at higher entropy numbers. The entropy for matrix LP OSA 07 is 8.41 and for OPF 3754 is 8.39. We use the entropy specified in the Scipy stats module. A single number is concise and satisfying. If you are pondering why they are so close contrary to their sparsity we discuss this next.

5 UNIFORM DISTRIBUTION

We know that we should **not** compare the entropy numbers of two matrices because entropy does not use any information about the order of the buckets only their probabilities. By construction, the matrices are quite different in sparsity and in shapes, however their entropy numbers are very close. Two matrices with the same number of non-zeros, spaced well enough in the proper number of bin, will have the same entropy. To appreciate their different sparsity, we should compare their entropy distributions by Jensen-Shannon measure (which is a symmetric measure, please do not use Kullback-Leibler KL divergence) [2]. Or we could use a representation of a hierarchical 2d-entropy, see Figure 2, where the entropy is split into 2x2, 4x4 and 8x8 (or fewer if the distribution is not square). We have a hierarchical entropy heat maps.

We can see that a granular entropy summarizes better the nature of the matrix because it keep some spatial information. In this work, the entropy vector is used mostly for visualization purpose more than for comparison purpose.

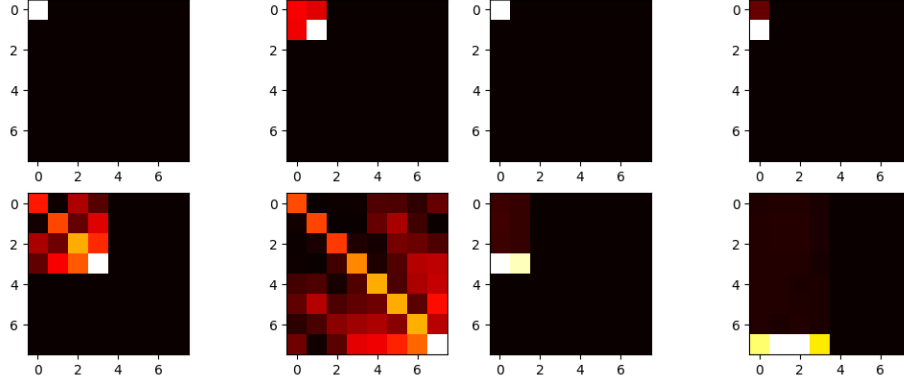


Fig. 2. Hierarchical 2D entropy for OPF 3754 (left) and LP OSA 07 (right).

Of course, we can appreciate how the matrix LP OSA 07 has a few very heavy rows and they are clustered. This matrix will help us showing how randomization need some tips. Now we apply row and column random permutation once by row and one by column: Figure 3: OPF has now entropy 11.27 and LP 9.26. The numerical difference is significant. The good news is that for entropy, being an expectation, we can use simple techniques like bootstrap to show that the difference is significant or we have shown that Jensen-Shannon can be used and a significance level is available. What we like to see is the the hierarchical entropy heat map is becoming *more* uniform for at least one of the matrix.

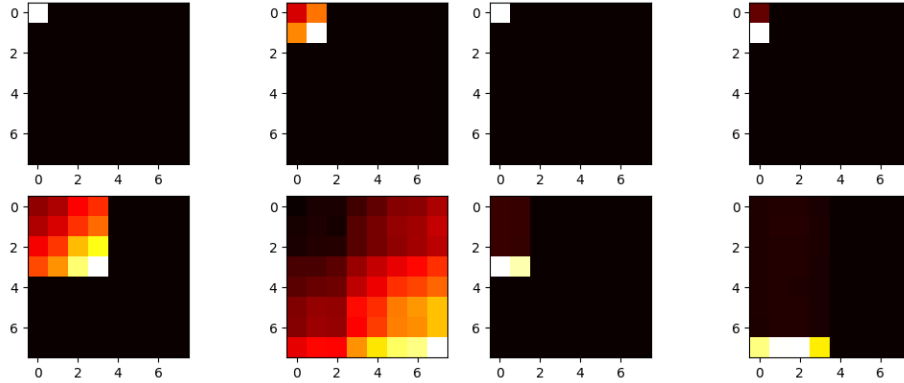


Fig. 3. Hierarchical 2D entropy after row and column random permutation for OPF 3754 (left) and LP OSA 07 (right).

In practice, permutations need some help especially for relatively large matrices. As you can see, the permutation affects locally the matrix. Of course, it depends on the implementation of the random permutation, we use *numpy* for this. It is reasonable that a slightly modified version of the original is still a random selection and unfortunately they seem too likely in practice. We need to compensate or help the randomization. If we are able to identify the row and column that divide high and low density, we could use them as pivot for a shuffle like in a quick-sort algorithm. We could apply a sorting algorithm but its complexity will the same of SpMV. We use a gradients operations to choose the element with maximum steepness, Figure 4 and 5

LP achieves entropy 8.67 and 9.58 and OPF achieves 10.47 and 11.40.

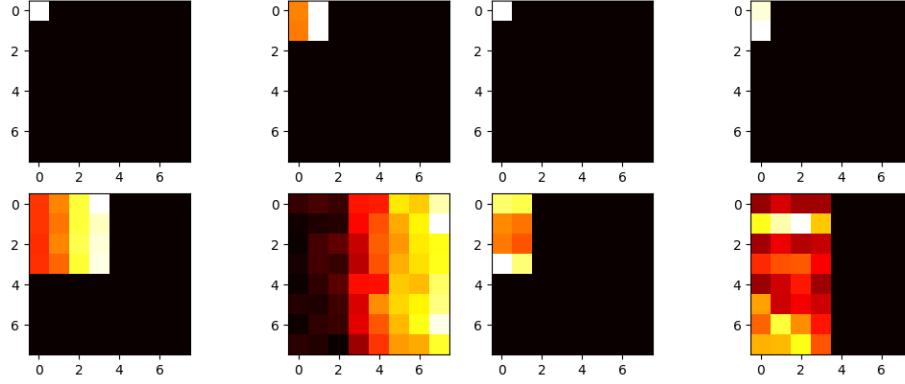


Fig. 4. Hierarchical 2D entropy after height gradient based shuffle and row random permutation for OPF 3754 (left) and LP OSA 07 (right).

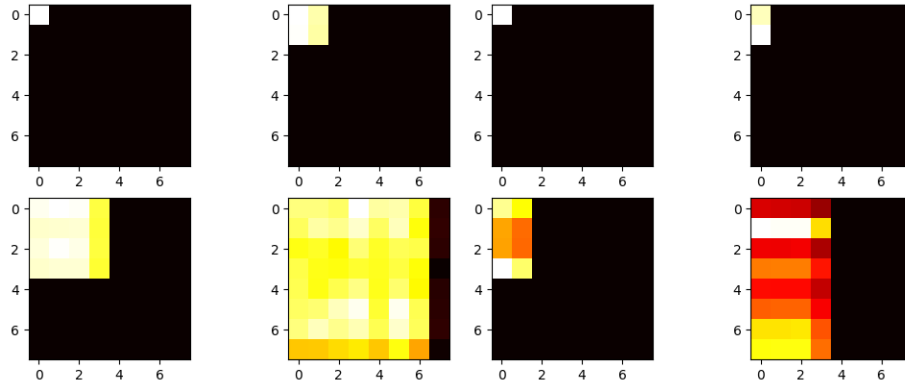


Fig. 5. Hierarchical 2D entropy after height and width gradient shuffle and row and column random permutation for OPF 3754 (left) and LP OSA 07 (right).

If the goal is to achieve a uniformly sparse matrix, it seems that we have the tools to compute and to measure such a sparsity. We admit that we do not try to find the best permutation. But our real goal is to create a work bench where randomization can be tested on different architectures and different algorithms. A randomization with a measurable uniform distribution is preferable than just random. We are interested to find out when random is enough or not enough. Also, consider that to achieve a uniform distribution, we do not need a random transformation and any permutation balancing the number of non-zero is possible, but for now not looked for.

6 MEASURING THE RANDOMIZATION EFFECTS

Whether or not this ever applied to the reader, when we have timed algorithms (i.e., measure execution time), we came to expect variation. The introduction of randomization may hide behind the ever present variance, after all these are

algorithms on *small* inputs and small error can be comparable to the overall execution time. Here, we must address this concern even before describing the experiments.

First, we execute every algorithm between 1000 and 5000 times. The time of each experiment is in the seconds, providing a granularity for which we are confident the measuring time error is under control. Thus, for each experiment we provide an average execution time: we measure the time and we divide by the number of trials. Cold starts, the first iteration, are still accounted. To make the measure portable across platform we present GFLOPS, that is, Giga (10^{12}) floating operations per second: $2 * nnz$ divided by the average time in seconds.

Then we repeat the same experiment 32 times. Permutations in *numpy* Python uses a seed that is time sensitive: thus every experiment is independent from the previous. The number 32 is an old statistic trick and it is a minimum number of independent trials to approximate a normal distribution. In practice, they are not but the number is sufficient for most of the cases and it is an excellent starting point.

A short hand legend: **Reg** is the regular matrix without any permutation; **R** stands for random Row permutation; **G-R** stands for gradient-based row shuffle and random row permutation; **G-C** stands for gradient-based column shuffle and random column permutation; **R-C** stands for random row and column permutation. This legend is used in the pictures to be concise, in the tables in the following sections, we use a verbose description. We shall clarify the gradient based approach in the experimental results section 8. Intuitively, we help the random permutation by a quick targeting of high and low volume of the histogram (and thus the matrix).

In Figure 6, we show two plots respectively of the CPU performance using COO and CSR SpMV algorithms for the matrix OPF 3754. The figure represents histograms: The x is GFLOPS and the y label is the number of counts. Thus we show what is the performance distribution of an algorithm. We can see that the CSR algorithms are consistent and the Regular (i.e., the original) has always the best performance. Also the variance of the computation time is small and the shape is approximately Gaussian. Different story for the COO, the permutations introduce long tails, thus $2\times$ performance advantage.

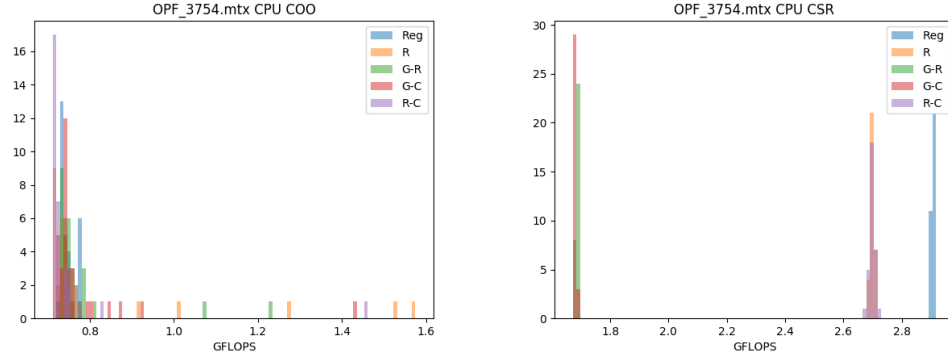


Fig. 6. CPU COO (left) and CPU CSR (left) for OPF 3754

If we take the original matrix and split into parts having the same number of rows, and execute them in parallel using different cores, we can see in Figure 7 that randomization is quite useful.

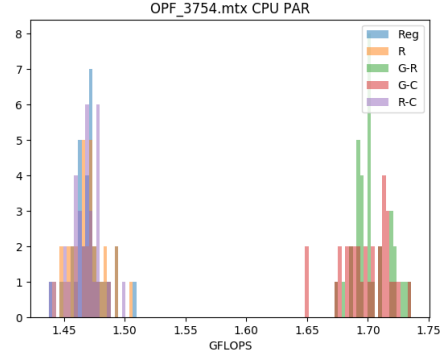


Fig. 7. Parallel CPU CSR for OPF 3754

216 In Figure 8, 9 and 10, randomization is harmful to the GPU implementation. The OPF 375 matrix is mostly diagonal,
 217 thus the vector \mathbf{x} is read in close quarters, randomization breaks it. If the load balance is fixed (i.e., by dividing the
 218 matrix by row and in equal row), randomization is beneficial.

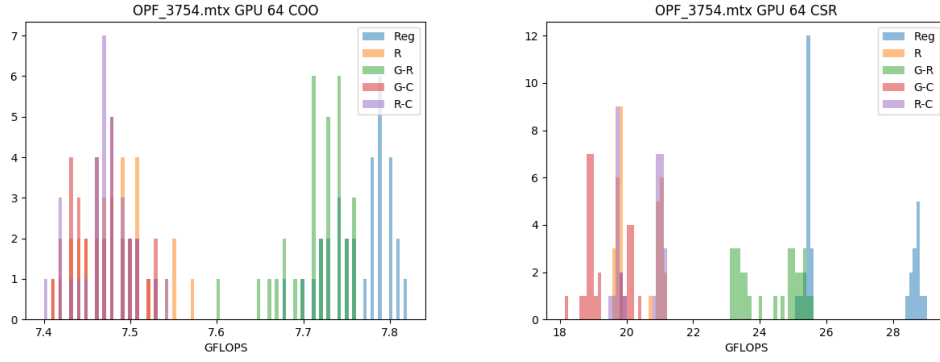


Fig. 8. Vega 20, GPU 64bits COO (left) and GPU CSR (right) for OPF 3754

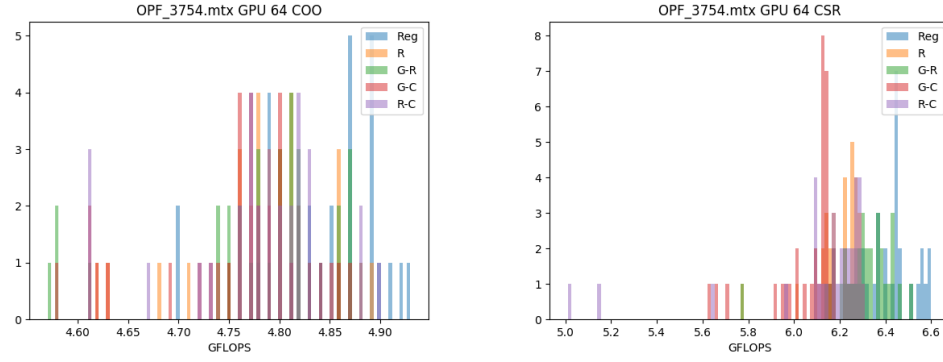


Fig. 9. Ellesmere, GPU 64bits COO (left) and GPU CSR (right) for OPF 3754

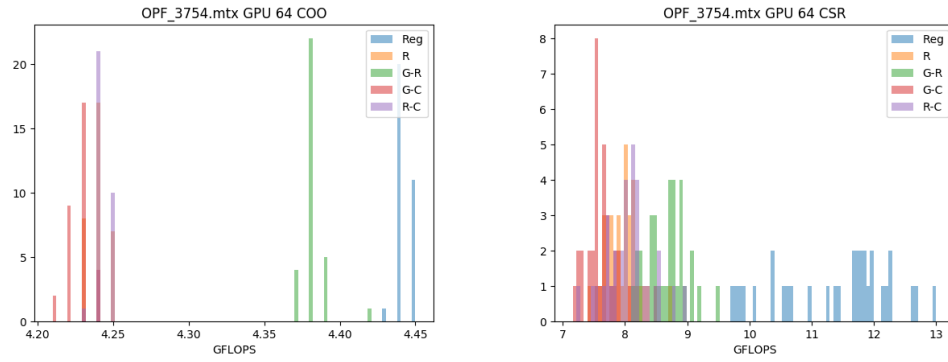


Fig. 10. Fiji, GPU 64bits COO (left) and GPU CSR (right) for OPF 3754

219 For matrix LP OSA 07, randomization helps clearly only for CPU CSR as we show in Figure 11. In Figure 12, 13, and
 220 14, we can see that randomization is harmful but for one GPU, we can show that a single exception is possible (40%
 221 improvement).

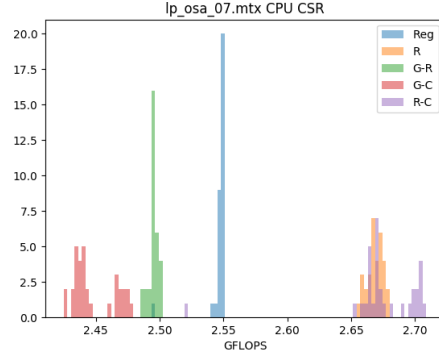


Fig. 11. CPU CSR for LP OSA 07

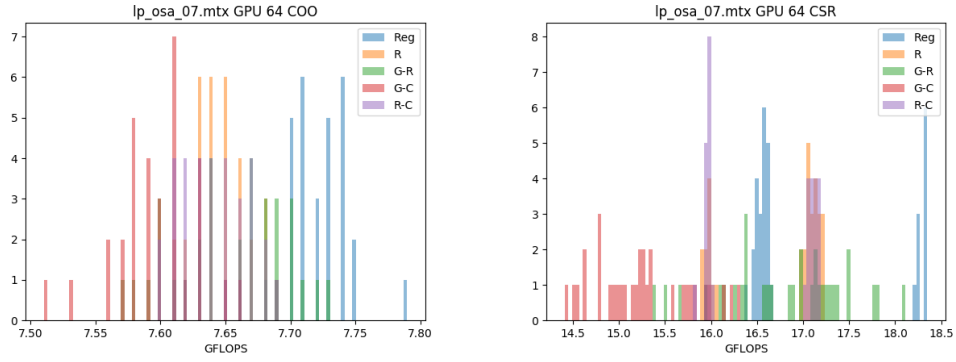


Fig. 12. Vega 20, GPU 64bits COO (left) and GPU CSR (right) for OPF 3754

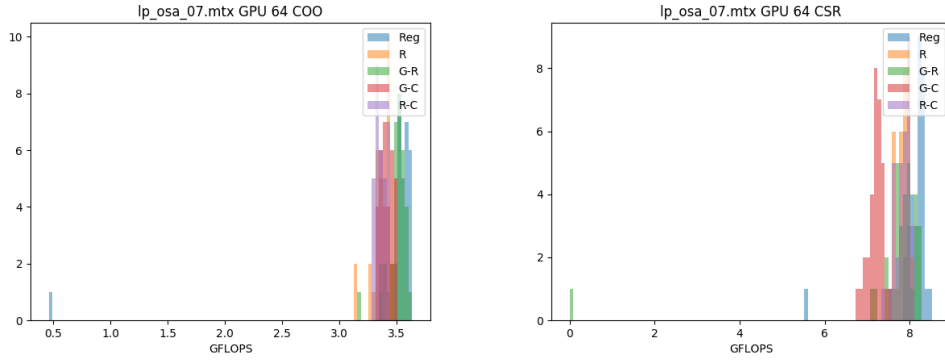


Fig. 13. Ellesmere, GPU 64bits COO (left) and GPU CSR (right) for OPF 3754

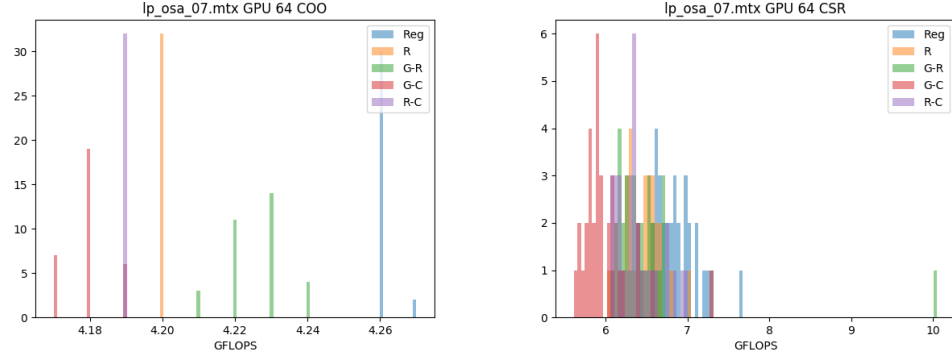


Fig. 14. Fiji, GPU 64bits COO (left) and GPU CSR (right) for OPF 3754

222 An example, the matrix MULT DCOP 01, is where randomization is useful for the CPU, GPU, and the parallel version
 223 Figure 15, 16 - 19 and the gains can be up to 10-15%. Consider, we can achieve these improvements without any insights
 224 to the architecture, the algorithms and their relationships.

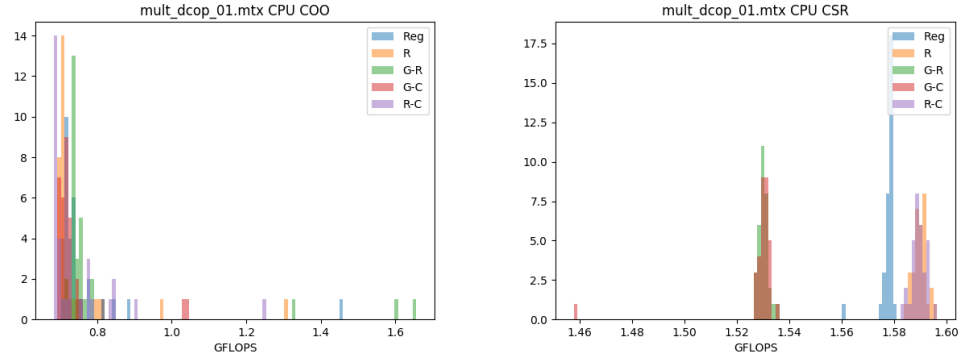


Fig. 15. CPU COO (left) and CPU CSR (right) for MULT DCOP 01

225 What does it mean when randomization does not work? The matrices we use in this work are not chosen randomly
 226 (pun not intended), they are the matrices that are difficult to handle in our custom SpMV engines using a combination
 227 of sorting networks and systolic arrays. If randomization does not work in our simplified work bench, will not work in
 228 our specialized architecture because the reorganization of the matrix or the input and output vector does not have
 229 the necessary parallelism, data locality, and data streaming. We need to do something else. In this case disrupting the
 230 memory pattern is not sufficient. Thus, if we cannot beat the pattern, we must exploit it, well not in this work.

231 7 WORKLOADS

232 In the previous sections, we defined what we mean for randomization and we present our tools of tricks for the measure
 233 of the effects of randomization. Here we describe the work loads, the applications, we use to test the effects of the
 234 randomization.

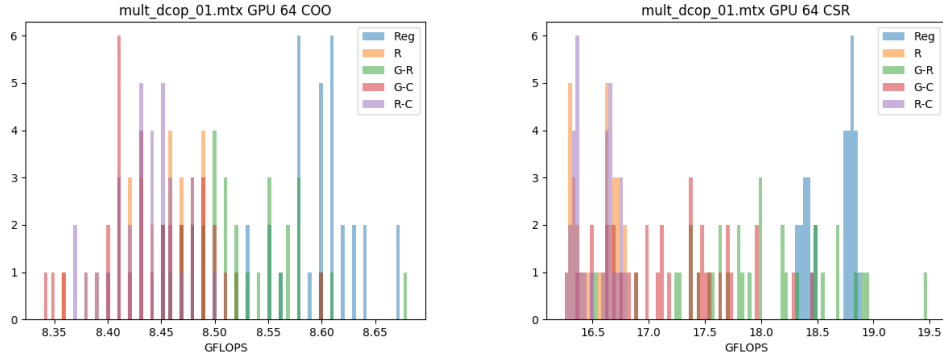


Fig. 16. Vega 20, GPU 64bits COO (left) and GPU CSR (right) for MULT DCOP 01

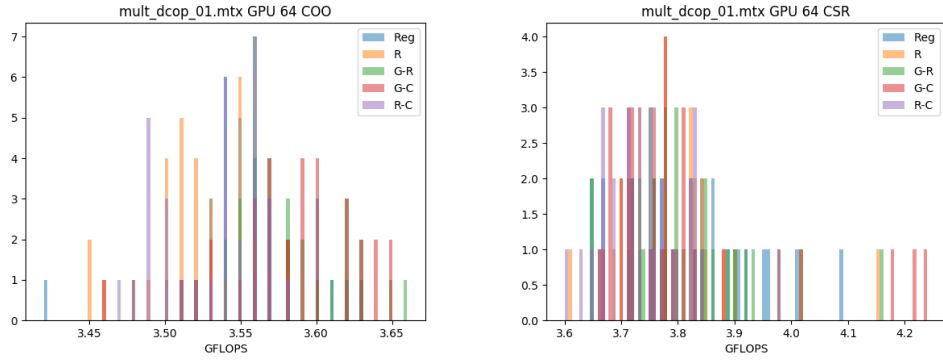


Fig. 17. Ellesmere, GPU 64bits COO (left) and GPU CSR (right) for MULT DCOP 01

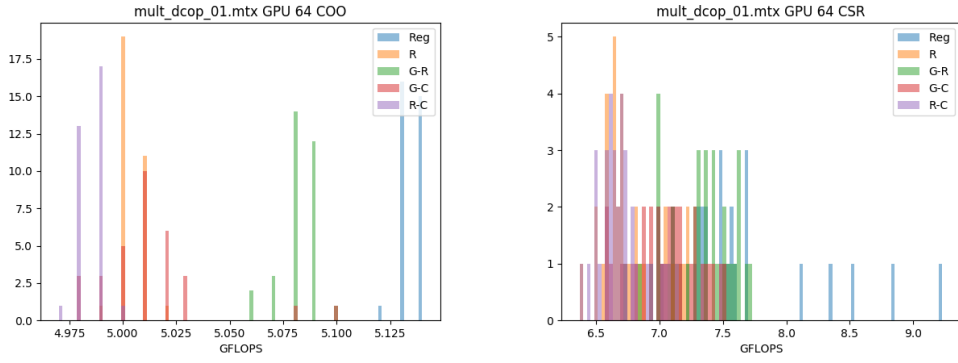


Fig. 18. Fiji, GPU 64bits COO (left) and GPU CSR (right) for MULT DCOP 01

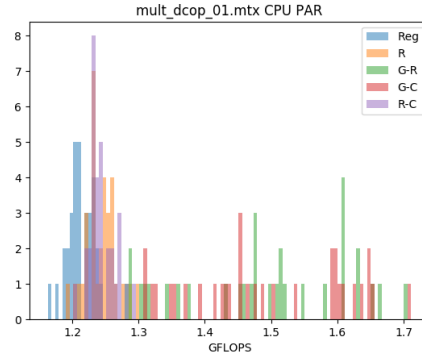


Fig. 19. Parallel CPU CSR for MULT DCOP 01

7.1 Python COO and CSR algorithms

The simplicity to compute the SpMV by the code $z = A * b$ in Python is very rewarding. By change of the matrix storage format, $A = A.tocsr(); z = A * b$, we have a different algorithm. The performance exploitation is moved to the lower level. The CSR implementation is often two times faster but there are edge cases where the COO and COO with randomization can go beyond and be surprisingly better: MUL DCOP 03 is an example where COO can do well.

Intuitively, Randomization can affect the performance because the basic implementation is a sorting algorithm and it is a fixed algorithm. There are many sorting algorithms and each can be optimal for a different initial distribution. If we knew what is the sorting algorithm we could tailor the input distribution. Here we just play with it.

In Section 8, we present all the results for CPU and GPUS. Keep in mind that these problems are hard, in the sense they do not have fancy performance sheets (these architectures can achieve Tera FLOPs sustained performance for dense computations). If we go through diligently, we can see that there is a 15x performance difference between the single thread CPU and Vega 20 GPU (i.e, 3 vs 40 GFLOPS).

7.2 Parallel CSR using up to 16 cores

Python provides the concept of Pool to exploit a naive parallel computation. We notice that work given to a Pool is split accordingly to the number of elements to separate HW cores. We also noticed that the work load move from a core to another, thus not ideal. Also we notice that Pool introduce a noticeable overhead: a Pool of 1, never achieves the performance of the single thread $z = A * b$. Using Pool allows us to investigate how a naive row partitioning without counting can scale up with number of cores. We tested by splitting the rows to 1–16 cores evenly (one thread per core) and we present the performance for only the best configuration. The randomization goal is to distribute the work uniformly: a balanced work distribution avoid the unfortunate case where a single core does all the work. We are pleased by the simplicity of the benchmark and we know we can do better.

7.3 GPU COO and CSR algorithms

In this work, we use AMD GPUs and *rocSPARSE* is their current software. The software has a few glitches but overall can be used for different generation of AMD GPUs. We use the COO and CSR algorithms and we provide performance measure for double precision only. The ideas of using different GPUs: it is important to verify that the randomization

can be applied independently of the HW. We are not here to compare performance across GPUs and CPUs. Often the limitation is the software, how the software can exploit the hardware or how the software will make easy to use a specific GPU. For example, the Fiji architecture is clearly superior to the Ellesmere, however the latter have better support and the system overall is more stable and user friendly.

The performance of the CSR algorithm is about two times faster than the COO. Most of the algorithms count the number of sparse elements in a row and thus they can decide the work load partition accordingly. Counting give you an edge but without changing the order of the computation there could be cases where the work load is not balanced and a little randomization could help and it does.

7.4 Randomization sometimes works

For the majority of the cases we investigated and reported in the following sections, Randomization does not work. However, there are cases where randomization does work and does work for different algorithms and architectures. If you are in the business of preconditioning, permutations are pretty cheap. If you can find a good one just consider like a preconditioning matrix, which it is.

This shows also that HW has to be more conscious, well the HW designer should, and accept that there are options at software level, at matrix level and beyond.

8 EXPERIMENTAL RESULTS

The main hardware setup is a AMD Threadripper with 16 cores. We have three Radeon GPUs: Vega 20 7nm, Pro 2xFiji, and Pro 2xEllesmere.

Vega 20 can deliver 3.5TFLOPS in double precision and it has 1TB/s HBM memory. Each Fiji provides 0.5 TFLOPS in double precision and has 512GB/s HBM, the card has two chips. The Ellesmere provides 0.3TFLOPS in double precision and has 224GB/s DDR5, the card has two chips. In the performance plots presented earlier and in the following, you will notice that the performance gap between these GPUs is not so marked. We can safely state that $vega \sim 2 \times Fiji$ and $Fiji \sim 2 \times ellesmere$

There are 4 basic randomization formats:

- **Random Row Permutation**, we take the original matrix and permute the rows.
- **Random Row and Column Permutation**, we take the original matrix and permute the rows and the columns.
- **Gradient based row permutation**, we compute the row histogram and we compute the gradient: $h_{i+1} - h_i$. We find a single point where the gradient is maximum, this is the pivot for a shuffle like a magician would shuffle a deck of cards. Then we permute the two parts randomly.
- **Gradient based row and column permutation**, As above but also for the columns.

For large matrices (large number of columns and rows) a permutation tends to be a close variation of the original, still a random permutation. The gradient allows us to describe two area of the original matrix where there is a clear and de-marked density variation: for example, there are two uniform distributed sub matrices but one denser than the other. A shuffle redistribute every other sample/card to different parts and these can be permuted locally.

We report in the following the performance results, we introduce a * following the best performance. This is tedious to read and, we assure, to write. The code and the results are available as software repository.

9 VEGA VII AND THREADRIPPER

296	mult_dcop_03.mtx
297	Regular
298	
299	CPU COO min 0.728 max 0.880 mean 0.757
300	CPU CSR min 1.563 max 1.581 mean 1.577
301	GPU 64 COO min 8.540 max* 8.670 mean 8.619
302	CSR min 18.320 max 18.930 mean 18.620
303	CPU PAR min 1.170 max 1.269 mean 1.226
304	H min 9.689 max 9.689 mean 9.689
305	Row-Premute
306	CPU COO min 0.710 max 0.845 mean 0.724
307	CPU CSR min 1.549 max* 1.597 mean 1.589
308	GPU 64 COO min 8.360 max 8.540 mean 8.442
309	CSR min 16.260 max 16.780 mean 16.551
310	CPU PAR min 1.205 max 1.319 mean 1.263
311	H min 10.737 max 10.742 mean 10.740
312	Row-Gradient
313	CPU COO min 0.706 max 1.603 mean 0.806
314	CPU CSR min 1.493 max 1.534 mean 1.528
315	GPU 64 COO min 8.430 max 8.610 mean 8.527
316	CSR min 17.070 max*18.970 mean 18.115
317	CPU PAR min 1.331 max 1.695 mean 1.513
318	H min 10.576 max 10.585 mean 10.580
319	Column-Gradient
320	CPU COO min 0.694 max* 1.632 mean 0.797
321	CPU CSR min 1.491 max 1.534 mean 1.529
322	GPU 64 COO min 8.350 max 8.520 mean 8.429
323	CSR min 15.970 max 18.180 mean 17.124
324	CPU PAR min 1.321 max* 1.728 mean 1.514
325	H min 10.826 max*10.840 mean 10.833
326	Row-Column-Permute
327	CPU COO min 0.688 max 0.757 mean 0.696
328	CPU CSR min 1.490 max 1.595 mean 1.584
329	GPU 64 COO min 8.380 max 8.500 mean 8.445
330	CSR min 16.230 max 16.780 mean 16.513
331	CPU PAR min 1.192 max 1.274 mean 1.237
332	H min 10.737 max 10.742 mean 10.740
333	mult_dcop_01.mtx
334	Regular
335	CPU COO min 0.710 max 1.453 mean 0.761
336	CPU CSR min 1.561 max 1.581 mean 1.578
337	GPU 64 COO min 8.520 max 8.670 mean 8.597
338	CSR min 18.320 max 18.870 mean 18.636
339	CPU PAR min 1.163 max 1.246 mean 1.212
340	H min 9.689 max 9.689 mean 9.689
341	Row-Premute
342	CPU COO min 0.699 max 1.305 mean 0.745
343	CPU CSR min 1.585 max 1.597 mean 1.590
344	GPU 64 COO min 8.360 max 8.520 mean 8.446
345	CSR min 16.260 max 16.780 mean 16.528
346	CPU PAR min 1.192 max 1.298 mean 1.242
347	H min 10.738 max 10.742 mean 10.740
348	Row-Gradient
349	CPU COO min 0.709 max* 1.656 mean 0.819
350	CPU CSR min 1.527 max 1.535 mean 1.530
351	GPU 64 COO min 8.450 max* 8.680 mean 8.527
352	CSR min 16.520 max*19.480 mean 17.984
353	CPU PAR min 1.280 max 1.704 mean 1.485
354	H min 10.572 max 10.585 mean 10.581
355	Column-Gradient
356	CPU COO min 0.698 max 1.042 mean 0.737
357	CPU CSR min 1.458 max 1.536 mean 1.528
358	GPU 64 COO min 8.340 max 8.600 mean 8.443
359	CSR min 16.360 max 18.450 mean 17.247
360	CPU PAR min 1.307 max* 1.712 mean 1.494
361	H min 10.823 max*10.841 mean 10.835
362	Row-Column-Permute
363	CPU COO min 0.683 max 1.247 mean 0.749
364	CPU CSR min 1.583 max* 1.595 mean 1.590
365	GPU 64 COO min 8.370 max 8.500 mean 8.435
366	CSR min 16.250 max 16.780 mean 16.518
367	CPU PAR min 1.206 max 1.291 mean 1.243
368	H min 10.738 max 10.742 mean 10.740

369	mult_dcop_02.mtx
370	Regular
371	CPU COO min 1.615 max* 1.677 mean 1.652
372	CPU CSR min 1.539 max 1.579 mean 1.575
373	GPU 64 COO min 8.530 max* 8.700 mean 8.614
374	CSR min 18.290 max 18.890 mean 18.597
375	CPU PAR min 1.120 max 1.248 mean 1.211
376	H min 9.689 max 9.689 mean 9.689
377	Row-Premute
378	CPU COO min 0.684 max 0.780 mean 0.705
379	CPU CSR min 1.558 max* 1.596 mean 1.588
380	GPU 64 COO min 8.360 max 8.490 mean 8.433
381	CSR min 16.240 max 16.750 mean 16.552
382	CPU PAR min 1.182 max 1.277 mean 1.242
383	H min 10.737 max 10.742 mean 10.740
384	Row-Gradient
385	CPU COO min 0.704 max 1.373 mean 0.790
386	CPU CSR min 1.518 max 1.535 mean 1.529
387	GPU 64 COO min 8.420 max 8.590 mean 8.517
388	CSR min 16.680 max*19.550 mean 17.907
389	CPU PAR min 1.328 max* 1.713 mean 1.484
390	H min 10.572 max 10.585 mean 10.581
391	Column-Gradient
392	CPU COO min 0.697 max 1.460 mean 0.742
393	CPU CSR min 1.517 max 1.534 mean 1.527
394	GPU 64 COO min 8.330 max 8.490 mean 8.420
395	CSR min 16.020 max 18.390 mean 17.303
396	CPU PAR min 1.321 max 1.709 mean 1.557
397	H min 10.823 max*10.843 mean 10.835
398	Row-Column-Permute
399	CPU COO min 0.691 max 0.746 mean 0.698
400	CPU CSR min 1.568 max 1.595 mean 1.587
401	GPU 64 COO min 8.350 max 8.500 mean 8.436
402	CSR min 16.250 max 16.780 mean 16.517
403	CPU PAR min 1.187 max 1.280 mean 1.228
404	H min 10.739 max 10.743 mean 10.740
405	lp_fit2d.mtx
406	Regular
407	CPU COO min 0.774 max 0.804 mean 0.793
408	CPU CSR min 2.538 max 2.550 mean 2.547
409	GPU 64 COO min 7.060 max 7.170 mean 7.101
410	CSR min 15.650 max*18.700 mean 18.031
411	CPU PAR min 1.537 max 1.645 mean 1.590
412	H min 11.109 max 11.109 mean 11.109
413	Row-Premute
414	CPU COO min 0.740 max 0.776 mean 0.746
415	CPU CSR min 3.302 max* 3.328 mean 3.317
416	GPU 64 COO min 7.040 max* 7.180 mean 7.098
417	CSR min 15.690 max 18.580 mean 16.732
418	CPU PAR min 1.327 max 1.482 mean 1.422
419	H min 11.098 max 11.105 mean 11.101
420	Row-Gradient
421	CPU COO min 0.739 max* 2.092 mean 1.091
422	CPU CSR min 2.539 max 2.546 mean 2.543
423	GPU 64 COO min 7.040 max 7.150 mean 7.100
424	CSR min 15.520 max 18.560 mean 17.547
425	CPU PAR min 1.401 max 1.661 mean 1.525
426	H min 11.109 max 11.109 mean 11.109
427	Column-Gradient
428	CPU COO min 0.726 max 2.065 mean 1.011
429	CPU CSR min 2.539 max 2.550 mean 2.546
430	GPU 64 COO min 6.800 max 7.140 mean 7.080
431	CSR min 15.480 max 18.560 mean 16.866
432	CPU PAR min 1.391 max* 1.737 mean 1.563
433	H min 11.329 max 11.333 mean 11.331
434	Row-Column-Permute
435	CPU COO min 0.746 max 0.782 mean 0.754
436	CPU CSR min 3.310 max 3.324 mean 3.318
437	GPU 64 COO min 7.030 max 7.160 mean 7.100
438	CSR min 15.730 max 18.530 mean 17.362
439	CPU PAR min 1.340 max 1.451 mean 1.401
440	H min 11.099 max 11.104 mean 11.102
441	bloweya.mtx
442	Regular

443		CPU COO	min	0.727	max*	1.815	mean	0.892	517		GPU 64 COO	min	11.340	max*	11.860	mean	11.441
444		CPU CSR	min	2.867	max*	2.936	mean	2.917	518		CSR	min	36.010	max*	40.960	mean	38.048
445		GPU 64 COO	min	0.000	max	0.000	mean	0.000	519		CPU PAR	min	2.019	max	2.204	mean	2.130
446		CSR	min	0.000	max	0.000	mean	0.000	520		H	min	8.228	max	8.228	mean	8.228
447		CPU PAR	min	1.680	max*	1.751	mean	1.719	521	Row-Premute							
448		H	min	7.205	max	7.205	mean	7.205	522		CPU COO	min	0.718	max	0.751	mean	0.732
449	Row-Premute								523		CPU CSR	min	2.488	max	2.507	mean	2.498
450		CPU COO	min	0.678	max	1.483	mean	0.746	524		GPU 64 COO	min	10.810	max	11.090	mean	10.949
451		CPU CSR	min	2.311	max	2.326	mean	2.320	525		CSR	min	24.860	max	26.410	mean	25.527
452		GPU 64 COO	min	6.840	max*	7.270	mean	6.930	526		CPU PAR	min	1.978	max	2.290	mean	2.135
453		CSR	min	15.650	max	16.800	mean	16.233	527		H	min	11.836	max	11.840	mean	11.838
454		CPU PAR	min	1.649	max	1.730	mean	1.682	528	Row-Gradient							
455		H	min	11.026	max	11.031	mean	11.029	529		CPU COO	min	0.722	max	1.794	mean	0.769
456	Row-Gradient								530		CPU CSR	min	2.407	max	2.421	mean	2.416
457		CPU COO	min	0.708	max	1.209	mean	0.779	531		GPU 64 COO	min	11.210	max	11.480	mean	11.317
458		CPU CSR	min	1.648	max	1.735	mean	1.709	532		CSR	min	31.920	max	34.690	mean	33.246
459		GPU 64 COO	min	6.920	max	7.080	mean	7.015	533		CPU PAR	min	2.184	max*	2.302	mean	2.232
460		CSR	min	16.950	max	19.500	mean	17.794	534		H	min	10.742	max	10.757	mean	10.748
461		CPU PAR	min	1.497	max	1.743	mean	1.608	535	Column-Gradient							
462		H	min	10.298	max	10.304	mean	10.301	536		CPU COO	min	0.720	max	0.916	mean	0.742
463	Column-Gradient								537		CPU CSR	min	2.395	max	2.410	mean	2.402
464		CPU COO	min	0.709	max	1.536	mean	0.817	538		GPU 64 COO	min	10.840	max	11.070	mean	10.946
465		CPU CSR	min	1.705	max	1.753	mean	1.735	539		CSR	min	24.340	max	26.140	mean	25.393
466		GPU 64 COO	min	6.800	max	7.120	mean	6.865	540		CPU PAR	min	2.184	max	2.272	mean	2.223
467		CSR	min	15.480	max*	17.710	mean	16.470	541		H	min	11.873	max	11.882	mean	11.878
468		CPU PAR	min	1.446	max	1.718	mean	1.591	542	Row-Column-Permute							
469		H	min	10.880	max	10.886	mean	10.883	543		CPU COO	min	0.707	max	0.748	mean	0.714
470	Row-Column-Permute								544		CPU CSR	min	2.458	max	2.511	mean	2.506
471		CPU COO	min	0.670	max	1.024	mean	0.706	545		GPU 64 COO	min	10.880	max	11.070	mean	10.957
472		CPU CSR	min	2.199	max	2.340	mean	2.326	546		CSR	min	24.890	max	26.490	mean	25.642
473		GPU 64 COO	min	6.800	max	6.980	mean	6.933	547		CPU PAR	min	2.209	max	2.282	mean	2.240
474		CSR	min	15.610	max	16.900	mean	16.227	548		H	min	11.834	max*	11.840	mean	11.838
475		CPU PAR	min	1.598	max	1.668	mean	1.632	549	brainpc2.mtx							
476		H	min	11.025	max*	11.032	mean	11.029	550	Regular							
477	lp_osa_07.mtx								551		CPU COO	min	0.732	max	0.751	mean	0.744
478	Regular								552		CPU CSR	min	2.885	max*	2.916	mean	2.909
479		CPU COO	min	0.715	max	1.798	mean	0.885	553		GPU 64 COO	min	0.000	max	0.000	mean	0.000
480		CPU CSR	min	2.495	max	2.551	mean	2.547	554		CSR	min	0.000	max	0.000	mean	0.000
481		GPU 64 COO	min	7.650	max*	7.790	mean	7.718	555		CPU PAR	min	1.276	max	1.299	mean	1.286
482		CSR	min	16.390	max*	18.350	mean	17.093	556		H	min	7.478	max	7.478	mean	7.478
483		CPU PAR	min	0.963	max	1.012	mean	0.995	557	Row-Premute							
484		H	min	8.412	max	8.412	mean	8.412	558		CPU COO	min	0.727	max	0.855	mean	0.736
485	Row-Premute								559		CPU CSR	min	2.385	max	2.411	mean	2.397
486		CPU COO	min	0.720	max*	2.078	mean	1.104	560		GPU 64 COO	min	8.120	max	8.410	mean	8.206
487		CPU CSR	min	2.656	max*	2.679	mean	2.669	561		CSR	min	18.670	max	19.960	mean	19.536
488		GPU 64 COO	min	7.610	max	7.690	mean	7.647	562		CPU PAR	min	1.293	max	1.340	mean	1.314
489		CSR	min	15.910	max	17.210	mean	16.750	563		H	min	9.809	max	9.813	mean	9.811
490		CPU PAR	min	0.890	max	0.940	mean	0.918	564	Row-Gradient							
491		H	min	9.255	max	9.258	mean	9.256	565		CPU COO	min	0.696	max*	1.546	mean	0.785
492	Row-Gradient								566		CPU CSR	min	1.361	max	1.420	mean	1.411
493		CPU COO	min	0.725	max	2.078	mean	1.041	567		GPU 64 COO	min	8.190	max*	8.550	mean	8.302
494		CPU CSR	min	2.487	max	2.502	mean	2.495	568		CSR	min	18.700	max*	21.000	mean	19.890
495		GPU 64 COO	min	7.570	max	7.730	mean	7.655	569		CPU PAR	min	1.435	max	1.666	mean	1.549
496		CSR	min	15.370	max	18.100	mean	16.803	570		H	min	9.721	max	9.727	mean	9.723
497		CPU PAR	min	1.435	max	1.796	mean	1.592	571	Column-Gradient							
498		H	min	8.637	max	8.678	mean	8.672	572		CPU COO	min	0.698	max	1.467	mean	0.746
499	Column-Gradient								573		CPU CSR	min	1.377	max	1.423	mean	1.414
500		CPU COO	min	0.724	max	1.990	mean	1.000	574		GPU 64 COO	min	8.110	max	8.290	mean	8.187
501		CPU CSR	min	2.425	max	2.477	mean	2.448	575		CSR	min	18.090	max	20.190	mean	19.217
502		GPU 64 COO	min	7.510	max	7.660	mean	7.596	576		CPU PAR	min	1.345	max*	1.681	mean	1.518
503		CSR	min	14.410	max	16.290	mean	15.267	577		H	min	10.369	max*	10.372	mean	10.370
504		CPU PAR	min	1.238	max	1.774	mean	1.534	578	Row-Column-Permute							
505		H	min	9.447	max*	9.603	mean	9.576	579		CPU COO	min	0.698	max	1.390	mean	0.788
506	Row-Column-Permute								580		CPU CSR	min	2.387	max	2.410	mean	2.399
507		CPU COO	min	0.738	max	1.950	mean	1.071	581		GPU 64 COO	min	8.120	max	8.260	mean	8.191
508		CPU CSR	min	2.522	max	2.709	mean	2.675	582		CSR	min	18.530	max	19.960	mean	19.307
509		GPU 64 COO	min	7.600	max	7.690	mean	7.641	583		CPU PAR	min	1.295	max	1.347	mean	1.319
510		CSR	min	15.820	max	17.190	mean	16.572	584		H	min	9.809	max	9.813	mean	9.811
511		CPU PAR	min	0.891	max	0.944	mean	0.924	585	shermanACb.mtx							
512		H	min	9.255	max	9.258	mean	9.256	586	Regular							
513	ex19.mtx								587		CPU COO	min	0.712	max	1.201	mean	0.756
514	Regular								588		CPU CSR	min	1.558	max	1.601	mean	1.596
515		CPU COO	min	0.732	max*	1.837	mean	1.076	589		GPU 64 COO	min	7.080	max*	7.370	mean	7.184
516		CPU CSR	min	2.563	max*	2.586	mean	2.577	590		CSR	min	17.580	max*	19.480	mean	18.770

Manuscript submitted to ACM

739	CPU CSR	min	2.358	max	2.413	mean	2.392	813		CSR	min	19.960	max	21.190	mean	20.696
740	GPU 64 COO	min	11.430	max	11.770	mean	11.549	814		CPU PAR	min	1.303	max	1.371	mean	1.345
741	CSR	min	24.470	max	25.580	mean	24.785	815		H	min	10.059	max	10.062	mean	10.061
742	CPU PAR	min	1.758	max	1.896	mean	1.829	816	Row-Gradient							
743	H	min	11.872	max	11.877	mean	11.875	817		CPU COO	min	0.723	max	0.984	mean	0.753
744	Row-Gradient							818		CPU CSR	min	1.781	max	1.809	mean	1.803
745	CPU COO	min	0.716	max	0.775	mean	0.739	819		GPU 64 COO	min	9.380	max	9.660	mean	9.464
746	CPU CSR	min	1.651	max	1.689	mean	1.675	820		CSR	min	15.770	max	19.090	mean	18.037
747	GPU 64 COO	min	12.100	max	12.410	mean	12.205	821		CPU PAR	min	1.775	max	1.924	mean	1.868
748	CSR	min	31.670	max	34.910	mean	33.370	822	Column-Gradient	H	min	10.205	max	10.233	mean	10.219
749	CPU PAR	min	2.079	max	2.286	mean	2.207	823								
750	H	min	11.111	max	11.116	mean	11.113	824		CPU COO	min	0.715	max	0.926	mean	0.757
751	Column-Gradient							825		CPU CSR	min	1.729	max	1.802	mean	1.791
752	CPU COO	min	0.715	max	1.021	mean	0.743	826		GPU 64 COO	min	9.080	max	9.270	mean	9.158
753	CPU CSR	min	1.655	max	1.674	mean	1.666	827		CSR	min	13.980	max	15.780	mean	14.938
754	GPU 64 COO	min	11.340	max	11.560	mean	11.463	828		CPU PAR	min	1.751	max	1.906	mean	1.846
755	CSR	min	23.770	max	25.470	mean	24.489	829		H	min	11.213	max	11.232	mean	11.222
756	CPU PAR	min	2.056	max	2.172	mean	2.118	830	Row-Column-Permute							
757	H	min	12.040	max	12.047	mean	12.043	831		CPU COO	min	0.732	max	1.598	mean	0.785
758	Row-Column-Permute							832		CPU CSR	min	2.594	max	2.602	mean	2.599
759	CPU COO	min	0.677	max	0.785	mean	0.687	833		GPU 64 COO	min	9.340	max	9.460	mean	9.394
760	CPU CSR	min	2.325	max	2.434	mean	2.369	834		CSR	min	19.950	max	21.500	mean	20.544
761	GPU 64 COO	min	11.450	max	11.650	mean	11.538	835		CPU PAR	min	1.326	max	1.374	mean	1.354
762	CSR	min	24.330	max	25.560	mean	25.008	836		H	min	10.059	max	10.062	mean	10.061
763	CPU PAR	min	1.631	max	1.776	mean	1.709	837	mhd4800a.mtx							
764	H	min	11.873	max	11.877	mean	11.875	838	Regular							
765	OPF_3754.mtx							839		CPU COO	min	0.759	max	0.795	mean	0.780
766	Regular							840		CPU CSR	min	2.479	max	2.565	mean	2.557
767	CPU COO	min	0.726	max	0.774	mean	0.747	841		GPU 64 COO	min	5.490	max	5.650	mean	5.552
768	CPU CSR	min	2.898	max	2.919	mean	2.908	842		CSR	min	16.700	max	19.460	mean	18.004
769	GPU 64 COO	min	7.600	max	7.820	mean	7.766	843		CPU PAR	min	1.456	max	1.523	mean	1.492
770	CSR	min	25.070	max	29.030	mean	26.756	844		H	min	7.132	max	7.132	mean	7.132
771	CPU PAR	min	1.437	max	1.508	mean	1.471	845	Row-Premute							
772	H	min	8.393	max	8.393	mean	8.393	846		CPU COO	min	0.695	max	0.943	mean	0.726
773	Row-Premute							847		CPU CSR	min	2.480	max	2.488	mean	2.485
774	CPU COO	min	0.714	max	1.574	mean	0.817	848		GPU 64 COO	min	5.410	max	5.490	mean	5.453
775	CPU CSR	min	2.686	max	2.711	mean	2.699	849		CSR	min	15.700	max	17.520	mean	16.678
776	GPU 64 COO	min	7.410	max	7.570	mean	7.484	850		CPU PAR	min	1.422	max	1.514	mean	1.474
777	CSR	min	19.600	max	21.190	mean	20.307	851		H	min	10.959	max	10.966	mean	10.963
778	CPU PAR	min	1.443	max	1.505	mean	1.469	852	Row-Gradient							
779	H	min	11.267	max	11.272	mean	11.269	853		CPU COO	min	0.723	max	2.029	mean	0.990
780	Row-Gradient							854		CPU CSR	min	2.411	max	2.427	mean	2.421
781	CPU COO	min	0.723	max	1.232	mean	0.775	855		GPU 64 COO	min	5.490	max	5.560	mean	5.534
782	CPU CSR	min	1.672	max	1.691	mean	1.685	856		CSR	min	16.350	max	19.560	mean	17.784
783	GPU 64 COO	min	7.600	max	7.760	mean	7.716	857		CPU PAR	min	1.441	max	1.509	mean	1.477
784	CSR	min	23.160	max	25.590	mean	24.304	858		H	min	9.512	max	9.526	mean	9.520
785	CPU PAR	min	1.675	max	1.736	mean	1.703	859	Column-Gradient							
786	H	min	10.463	max	10.472	mean	10.468	860		CPU COO	min	0.721	max	1.802	mean	0.871
787	Column-Gradient							861		CPU CSR	min	2.393	max	2.408	mean	2.404
788	CPU COO	min	0.726	max	1.431	mean	0.778	862		GPU 64 COO	min	5.410	max	5.480	mean	5.453
789	CPU CSR	min	1.671	max	1.685	mean	1.679	863		CSR	min	15.680	max	17.870	mean	16.540
790	GPU 64 COO	min	7.410	max	7.530	mean	7.467	864		CPU PAR	min	1.429	max	1.488	mean	1.468
791	CSR	min	18.140	max	20.350	mean	19.315	865		H	min	10.931	max	10.945	mean	10.938
792	CPU PAR	min	1.650	max	1.736	mean	1.699	866	Row-Column-Permute							
793	H	min	11.393	max	11.401	mean	11.397	867		CPU COO	min	0.728	max	1.646	mean	1.037
794	Row-Column-Permute							868		CPU CSR	min	2.472	max	2.488	mean	2.480
795	CPU COO	min	0.711	max	1.458	mean	0.751	869		GPU 64 COO	min	5.410	max	5.480	mean	5.449
796	CPU CSR	min	2.678	max	2.717	mean	2.700	870		CSR	min	15.760	max	17.560	mean	16.654
797	GPU 64 COO	min	7.400	max	7.540	mean	7.471	871		CPU PAR	min	1.428	max	1.513	mean	1.474
798	CSR	min	19.560	max	21.150	mean	20.453	872		H	min	10.959	max	10.967	mean	10.963
799	CPU PAR	min	1.440	max	1.499	mean	1.467	873	gen4.mtx							
800	H	min	11.266	max	11.272	mean	11.269	874	Regular							
801	c-47.mtx							875		CPU COO	min	0.737	max	1.977	mean	1.431
802	Regular							876		CPU CSR	min	2.674	max	2.688	mean	2.681
803	CPU COO	min	0.754	max	1.829	mean	1.204	877		GPU 64 COO	min	5.900	max	6.000	mean	5.954
804	CPU CSR	min	2.610	max	2.624	mean	2.618	878		CSR	min	13.650	max	15.410	mean	14.657
805	GPU 64 COO	min	9.530	max	9.870	mean	9.640	879		CPU PAR	min	1.468	max	1.521	mean	1.491
806	CSR	min	23.990	max	25.910	mean	24.992	880		H	min	9.234	max	9.234	mean	9.234
807	CPU PAR	min	1.311	max	1.380	mean	1.357	881	Row-Premute							
808	H	min	8.364	max	8.364	mean	8.364	882		CPU COO	min	0.740	max	2.048	mean	1.121
809	Row-Premute							883		CPU CSR	min	2.777	max	2.798	mean	2.790
810	CPU COO	min	0.740	max	0.885	mean	0.755	884		GPU 64 COO	min	5.910	max	5.970	mean	5.944
811	CPU CSR	min	2.574	max	2.611	mean	2.597	885		CSR	min	13.700	max	15.370	mean	14.541
812	GPU 64 COO	min	9.320	max	9.510	mean	9.397	886		CPU PAR	min	1.468	max	1.546	mean	1.502

887		H	min 10.250 max 10.255 mean 10.252	961	CPU COO	min 0.735 max 1.806 mean 0.878
888	Row-Gradient			962	CPU CSR	min 2.706 max 2.744 mean 2.726
889		CPU COO	min 0.740 max 1.790 mean 0.994	963	GPU 64 COO	min 6.390 max 6.500 mean 6.433
890		CPU CSR	min 2.663 max 2.682 mean 2.674	964	CSR	min 19.780 max 22.870 mean 20.936
891		GPU 64 COO	min 5.890 max* 6.160 mean 5.946	965	CPU PAR	min 1.710 max 1.865 mean 1.785
892		CSR	min 13.780 max*17.520 mean 15.601	966	H	min 10.251 max 10.267 mean 10.257
893		CPU PAR	min 1.479 max* 1.619 mean 1.569	967	Column-Gradient	
894		H	min 9.939 max 9.955 mean 9.948	968		
895	Column-Gradient			969	CPU COO	min 0.728 max 1.792 mean 0.986
896		CPU COO	min 0.743 max 1.991 mean 0.981	970	CPU CSR	min 2.521 max 2.720 mean 2.703
897		CPU CSR	min 2.620 max 2.654 mean 2.646	971	GPU 64 COO	min 6.280 max 6.370 mean 6.327
898		GPU 64 COO	min 5.840 max 5.910 mean 5.885	972	CSR	min 18.000 max 19.720 mean 19.040
899		CSR	min 13.130 max 17.040 mean 15.008	973	CPU PAR	min 1.649 max 1.741 mean 1.702
900		CPU PAR	min 1.477 max 1.607 mean 1.559	974	H	min 11.113 max 11.121 mean 11.117
901		H	min 10.858 max*10.876 mean 10.864	975	Row-Column-Permute	
902	Row-Column-Permute			976	CPU COO	min 0.714 max 1.525 mean 0.957
903		CPU COO	min 0.742 max 2.010 mean 1.124	977	CPU CSR	min 2.876 max 2.892 mean 2.884
904		CPU CSR	min 2.789 max* 2.800 mean 2.795	978	GPU 64 COO	min 6.280 max 6.370 mean 6.322
905		GPU 64 COO	min 5.900 max 5.980 mean 5.941	979	CSR	min 17.960 max 19.670 mean 18.670
906		CSR	min 13.640 max 15.410 mean 14.556	980	CPU PAR	min 1.667 max 1.754 mean 1.710
907		CPU PAR	min 1.462 max 1.540 mean 1.504	981	TSOPF_RS_b39_c7.mtx	
908		H	min 10.250 max 10.253 mean 10.252	982	Regular	
909	Maragal_6.mtx			983	CPU COO	min 0.771 max 0.793 mean 0.780
910	Regular			984	CPU CSR	min 3.219 max* 3.232 mean 3.227
911		CPU COO	min 0.725 max 0.741 mean 0.729	985	GPU 64 COO	min 11.070 max*11.200 mean 11.142
912		CPU CSR	min 2.345 max 2.409 mean 2.372	986	CSR	min 37.050 max*42.100 mean 39.040
913		GPU 64 COO	min 18.200 max 18.770 mean 18.357	987	CPU PAR	min 1.910 max 2.027 mean 1.982
914		CSR	min 38.310 max*40.240 mean 39.477	988	H	min 7.304 max 7.304 mean 7.304
915		CPU PAR	min 0.789 max 0.813 mean 0.797	989	Row-Premute	
916		H	min 9.930 max 9.930 mean 9.930	990	CPU COO	min 0.701 max 0.722 mean 0.707
917	Row-Premute			991	CPU CSR	min 2.931 max 2.952 mean 2.942
918		CPU COO	min 0.709 max 0.779 mean 0.715	992	GPU 64 COO	min 10.860 max 11.030 mean 10.928
919		CPU CSR	min 2.675 max 2.715 mean 2.696	993	CSR	min 28.730 max 30.880 mean 29.483
920		GPU 64 COO	min 17.810 max 18.030 mean 17.935	994	CPU PAR	min 1.760 max 1.922 mean 1.851
921		CSR	min 29.650 max 30.580 mean 30.109	995	H	min 10.537 max 10.541 mean 10.539
922		CPU PAR	min 0.857 max 0.940 mean 0.904	996	Row-Gradient	
923		H	min 10.777 max 10.779 mean 10.778	997	CPU COO	min 0.747 max 0.808 mean 0.757
924	Row-Gradient			998	CPU CSR	min 2.606 max 2.648 mean 2.624
925		CPU COO	min 0.710 max* 1.566 mean 0.755	999	GPU 64 COO	min 10.850 max 11.120 mean 10.999
926		CPU CSR	min 2.042 max 2.159 mean 2.120	1000	CSR	min 33.910 max 37.600 mean 35.909
927		GPU 64 COO	min 18.460 max*18.960 mean 18.665	1001	CPU PAR	min 2.154 max* 2.245 mean 2.203
928		CSR	min 25.650 max 27.330 mean 26.549	1002	H	min 9.636 max 9.646 mean 9.642
929		CPU PAR	min 2.257 max 2.612 mean 2.416	1003	Column-Gradient	
930		H	min 11.251 max 11.301 mean 11.285	1004	CPU COO	min 0.718 max* 1.693 mean 0.802
931	Column-Gradient			1005	CPU CSR	min 2.502 max 2.585 mean 2.547
932		CPU COO	min 0.711 max 0.743 mean 0.725	1006	GPU 64 COO	min 10.700 max 10.990 mean 10.804
933		CPU CSR	min 2.036 max 2.161 mean 2.110	1007	CSR	min 27.230 max 29.380 mean 28.488
934		GPU 64 COO	min 17.840 max 18.860 mean 18.149	1008	CPU PAR	min 2.128 max 2.227 mean 2.172
935		CSR	min 19.410 max 20.690 mean 20.066	1009	H	min 11.131 max*11.222 mean 11.208
936		CPU PAR	min 2.174 max* 2.546 mean 2.349	1010	Row-Column-Permute	
937		H	min 12.011 max*12.072 mean 12.052	1011	CPU COO	min 0.709 max 0.726 mean 0.716
938	Row-Column-Permute			1012	CPU CSR	min 2.917 max 2.958 mean 2.940
939		CPU COO	min 0.712 max 0.971 mean 0.737	1013	GPU 64 COO	min 10.840 max 11.030 mean 10.930
940		CPU CSR	min 2.732 max* 2.751 mean 2.743	1014	CSR	min 28.780 max 30.810 mean 29.578
941		GPU 64 COO	min 17.720 max 18.070 mean 17.911	1015	CPU PAR	min 1.757 max 1.834 mean 1.792
942		CSR	min 29.600 max 30.500 mean 29.961	1016	H	min 10.537 max 10.540 mean 10.539
943		CPU PAR	min 0.827 max 0.954 mean 0.913			
944		H	min 10.776 max 10.778 mean 10.777			
945	aft01.mtx			1017	10 ELLESMERE	
946	Regular			1018	aft01.mtx	
947		CPU COO	min 0.735 max* 2.079 mean 1.069	1019	Regular	
948		CPU CSR	min 3.132 max* 3.154 mean 3.145	1020	GPU 64 COO	min 4.080 max* 4.280 mean 4.186
949		GPU 64 COO	min 6.390 max* 6.610 mean 6.457	1021	CSR	min 9.660 max*12.660 mean 11.485
950		CSR	min 19.990 max*23.250 mean 21.820	1022	H	min 7.811 max 7.811 mean 7.811
951		CPU PAR	min 1.746 max* 1.865 mean 1.812	1023	Row-Premute	
952		H	min 7.811 max 7.811 mean 7.811	1024	GPU 64 COO	min 3.860 max 4.090 mean 4.001
953	Row-Premute			1025	CSR	min 9.520 max 10.340 mean 9.936
954		CPU COO	min 0.714 max 1.648 mean 0.840	1026	H	min 11.161 max 11.167 mean 11.165
955		CPU CSR	min 2.864 max 2.892 mean 2.883	1027	Row-Gradient	
956		GPU 64 COO	min 6.280 max 6.380 mean 6.329	1028	GPU 64 COO	min 4.010 max 4.240 mean 4.135
957		CSR	min 17.980 max 19.700 mean 19.105	1029	CSR	min 5.890 max 11.350 mean 6.882
958		CPU PAR	min 1.729 max 1.850 mean 1.782	1030	H	min 10.246 max 10.262 mean 10.256
959		H	min 11.162 max 11.168 mean 11.165	1031	Column-Gradient	
960	Row-Gradient					

1032		GPU 64 COO min 3.850 max 4.100 mean 4.012	1106		H min 7.380 max 7.380 mean 7.380
1033		CSR min 5.460 max 8.790 mean 6.005	1107	Row-Premute	
1034		H min 11.112 max 11.122 mean 11.117	1108		GPU 64 COO min 4.820 max 4.940 mean 4.859
1035	Row-Column-Permute		1109		CSR min 5.080 max 6.520 mean 6.342
1036		GPU 64 COO min 3.850 max 4.080 mean 3.990	1110		H min 10.042 max 10.047 mean 10.044
1037		CSR min 5.420 max 6.760 mean 5.977	1111	Row-Gradient	
1038		H min 11.162 max*11.169 mean 11.165	1112		GPU 64 COO min 4.810 max* 4.940 mean 4.876
1039	bloweya.mtx		1113		CSR min 6.100 max* 6.560 mean 6.307
1040	Regular		1114		H min 9.681 max 9.704 mean 9.694
1041		GPU 64 COO min 0.000 max 0.000 mean 0.000	1115	Column-Gradient	
1042		CSR min 0.000 max 0.000 mean 0.000	1116		GPU 64 COO min 4.810 max 4.930 mean 4.869
1043		H min 7.205 max 7.205 mean 7.205	1117		CSR min 4.820 max 6.460 mean 6.208
1044	Row-Premute		1118		H min 10.554 max*10.661 mean 10.638
1045		GPU 64 COO min 3.800 max 3.940 mean 3.875	1119	Row-Column-Permute	
1046		CSR min 3.710 max 4.570 mean 4.399	1120		GPU 64 COO min 4.810 max 4.940 mean 4.864
1047		H min 11.025 max 11.031 mean 11.028	1121		CSR min 5.930 max 6.520 mean 6.379
1048	Row-Gradient		1122		H min 10.041 max 10.047 mean 10.044
1049		GPU 64 COO min 3.800 max* 4.120 mean 3.962	1123	cvxqp3.mtx	
1050		CSR min 4.340 max* 4.670 mean 4.546	1124	Regular	
1051		H min 10.296 max 10.307 mean 10.300	1125		GPU 64 COO min 3.350 max* 3.590 mean 3.483
1052	Column-Gradient		1126		CSR min 5.430 max* 9.260 mean 8.333
1053		GPU 64 COO min 3.880 max 4.100 mean 3.978	1127		H min 8.646 max 8.646 mean 8.646
1054		CSR min 4.240 max 4.570 mean 4.412	1128	Row-Premute	
1055		H min 10.881 max 10.886 mean 10.883	1129		GPU 64 COO min 3.230 max 3.480 mean 3.371
1056	Row-Column-Permute		1130		CSR min 7.560 max 8.220 mean 7.900
1057		GPU 64 COO min 3.800 max 3.980 mean 3.885	1131		H min 11.027 max 11.033 mean 11.030
1058		CSR min 4.130 max 4.540 mean 4.399	1132	Row-Gradient	
1059		H min 11.025 max*11.033 mean 11.029	1133		GPU 64 COO min 3.240 max 3.510 mean 3.396
1060	brainpc2.mtx		1134		CSR min 6.990 max 7.890 mean 7.574
1061	Regular		1135		H min 11.060 max 11.069 mean 11.064
1062		GPU 64 COO min 0.000 max 0.000 mean 0.000	1136	Column-Gradient	
1063		CSR min 0.000 max 0.000 mean 0.000	1137		GPU 64 COO min 3.240 max 3.480 mean 3.374
1064		H min 7.478 max 7.478 mean 7.478	1138		CSR min 6.980 max 7.900 mean 7.557
1065	Row-Premute		1139		H min 11.126 max*11.134 mean 11.130
1066		GPU 64 COO min 3.840 max* 6.750 mean 4.110	1140	Row-Column-Permute	
1067		CSR min 4.260 max* 4.500 mean 4.437	1141		GPU 64 COO min 3.110 max 3.470 mean 3.365
1068		H min 9.809 max 9.813 mean 9.811	1142		CSR min 4.810 max 8.210 mean 7.742
1069	Row-Gradient		1143		H min 11.026 max 11.032 mean 11.030
1070		GPU 64 COO min 0.640 max 4.030 mean 3.864	1144	ex19.mtx	
1071		CSR min 4.270 max 4.470 mean 4.383	1145	Regular	
1072		H min 9.722 max 9.727 mean 9.724	1146		GPU 64 COO min 2.450 max* 2.610 mean 2.564
1073	Column-Gradient		1147		CSR min 4.490 max 4.760 mean 4.714
1074		GPU 64 COO min 0.640 max 4.070 mean 3.898	1148		H min 8.228 max 8.228 mean 8.228
1075		CSR min 4.230 max 4.500 mean 4.386	1149	Row-Premute	
1076		H min 10.368 max*10.372 mean 10.370	1150		GPU 64 COO min 2.000 max 2.040 mean 2.021
1077	Row-Column-Permute		1151		CSR min 4.640 max 4.780 mean 4.733
1078		GPU 64 COO min 3.980 max 4.110 mean 4.027	1152		H min 11.835 max 11.840 mean 11.838
1079		CSR min 4.320 max 4.490 mean 4.437	1153	Row-Gradient	
1080		H min 9.809 max 9.813 mean 9.811	1154		GPU 64 COO min 2.240 max 2.390 mean 2.329
1081	c-47.mtx		1155		CSR min 4.570 max* 4.850 mean 4.807
1082	Regular		1156		H min 10.742 max 10.752 mean 10.747
1083		GPU 64 COO min 3.980 max* 4.080 mean 4.026	1157	Column-Gradient	
1084		CSR min 4.760 max 4.850 mean 4.812	1158		GPU 64 COO min 2.010 max 2.050 mean 2.034
1085		H min 8.364 max 8.364 mean 8.364	1159		CSR min 4.570 max 4.760 mean 4.701
1086	Row-Premute		1160		H min 11.872 max*11.881 mean 11.878
1087		GPU 64 COO min 3.880 max 4.010 mean 3.942	1161	Row-Column-Permute	
1088		CSR min 4.040 max 4.900 mean 4.807	1162		GPU 64 COO min 2.000 max 2.040 mean 2.023
1089		H min 10.059 max 10.063 mean 10.061	1163		CSR min 0.770 max 4.780 mean 4.594
1090	Row-Gradient		1164		H min 11.835 max 11.840 mean 11.838
1091		GPU 64 COO min 3.900 max 4.050 mean 3.976	1165	gen4.mtx	
1092		CSR min 4.380 max 4.740 mean 4.630	1166	Regular	
1093		H min 10.201 max 10.228 mean 10.214	1167		GPU 64 COO min 4.880 max 4.980 mean 4.900
1094	Column-Gradient		1168		CSR min 10.020 max*11.300 mean 10.716
1095		GPU 64 COO min 3.860 max 3.990 mean 3.936	1169		H min 9.234 max 9.234 mean 9.234
1096		CSR min 4.350 max 4.610 mean 4.525	1170	Row-Premute	
1097		H min 11.204 max*11.241 mean 11.222	1171		GPU 64 COO min 4.860 max 4.930 mean 4.890
1098	Row-Column-Permute		1172		CSR min 0.330 max 11.200 mean 10.038
1099		GPU 64 COO min 3.890 max 4.020 mean 3.953	1173		H min 10.249 max 10.254 mean 10.252
1100		CSR min 4.490 max* 4.920 mean 4.840	1174	Row-Gradient	
1101		H min 10.058 max 10.063 mean 10.061	1175		GPU 64 COO min 4.860 max* 4.990 mean 4.908
1102	case9.mtx		1176		CSR min 9.160 max 11.240 mean 10.435
1103	Regular		1177		H min 9.939 max 9.961 mean 9.947
1104		GPU 64 COO min 0.000 max 0.000 mean 0.000	1178	Column-Gradient	
1105		CSR min 0.000 max 0.000 mean 0.000	1179		GPU 64 COO min 4.780 max 4.880 mean 4.816

1180		CSR min 7.770 max 10.570 mean 9.407	1254	Row-Premute		
1181		H min 10.851 max*10.876 mean 10.864	1255		GPU 64 COO min 4.420 max 4.520 mean 4.445	
1182	Row-Column-Permute		1256		CSR min 10.520 max 10.880 mean 10.696	
1183		GPU 64 COO min 4.850 max 4.950 mean 4.886	1257		H min 10.960 max*10.968 mean 10.963	
1184		CSR min 10.220 max 11.280 mean 10.748	1258	Row-Gradient		
1185		H min 10.250 max 10.255 mean 10.252	1259		GPU 64 COO min 4.570 max 4.690 mean 4.605	
1186	lp_fit2d.mtx		1260		CSR min 4.550 max 13.350 mean 12.479	
1187	Regular		1261		H min 9.508 max 9.527 mean 9.520	
1188		GPU 64 COO min 4.360 max* 4.640 mean 4.515	1262	Column-Gradient		
1189		CSR min 10.080 max 10.900 mean 10.491	1263		GPU 64 COO min 4.430 max 4.530 mean 4.461	
1190		H min 11.109 max 11.109 mean 11.109	1264		CSR min 10.250 max 10.940 mean 10.603	
1191	Row-Premute		1265		H min 10.934 max 10.945 mean 10.939	
1192		GPU 64 COO min 4.170 max 4.630 mean 4.476	1266	Row-Column-Permute		
1193		CSR min 0.910 max 10.910 mean 10.257	1267		GPU 64 COO min 4.420 max 4.520 mean 4.450	
1194		H min 11.098 max 11.104 mean 11.101	1268		CSR min 7.380 max 10.900 mean 10.598	
1195	Row-Gradient		1269		H min 10.959 max 10.967 mean 10.963	
1196		GPU 64 COO min 4.370 max 4.630 mean 4.529	1270	mult_dcop_01.mtx		
1197		CSR min 10.030 max 10.970 mean 10.624	1271	Regular		
1198		H min 11.109 max 11.109 mean 11.109	1272		GPU 64 COO min 3.420 max 3.630 mean 3.555	
1199	Column-Gradient		1273		CSR min 3.650 max 4.090 mean 3.814	
1200		GPU 64 COO min 4.250 max 4.640 mean 4.499	1274		H min 9.689 max 9.689 mean 9.689	
1201		CSR min 8.510 max*11.010 mean 10.505	1275	Row-Premute		
1202		H min 11.328 max*11.333 mean 11.331	1276		GPU 64 COO min 3.450 max 3.580 mean 3.521	
1203	Row-Column-Permute		1277		CSR min 3.610 max 4.150 mean 3.785	
1204		GPU 64 COO min 4.350 max 4.640 mean 4.511	1278		H min 10.738 max 10.742 mean 10.740	
1205		CSR min 10.040 max 10.790 mean 10.468	1279	Row-Gradient		
1206		H min 11.097 max 11.106 mean 11.101	1280		GPU 64 COO min 3.510 max* 3.660 mean 3.579	
1207	lp_osa_07.mtx		1281		CSR min 3.650 max 4.160 mean 3.806	
1208	Regular		1282		H min 10.576 max 10.585 mean 10.580	
1209		GPU 64 COO min 0.460 max* 3.640 mean 3.456	1283	Column-Gradient		
1210		CSR min 5.570 max* 8.530 mean 8.106	1284		GPU 64 COO min 3.460 max 3.650 mean 3.584	
1211		H min 8.412 max 8.412 mean 8.412	1285		CSR min 3.660 max* 4.240 mean 3.799	
1212	Row-Premute		1286		H min 10.826 max*10.842 mean 10.836	
1213		GPU 64 COO min 3.140 max 3.450 mean 3.367	1287	Row-Column-Permute		
1214		CSR min 7.600 max 8.070 mean 7.853	1288		GPU 64 COO min 3.470 max 3.580 mean 3.532	
1215		H min 9.255 max 9.258 mean 9.256	1289		CSR min 3.600 max 3.980 mean 3.743	
1216	Row-Gradient		1290		H min 10.738 max 10.742 mean 10.740	
1217		GPU 64 COO min 3.190 max 3.610 mean 3.509	1291	mult_dcop_02.mtx		
1218		CSR min 0.000 max 8.260 mean 7.597	1292	Regular		
1219		H min 8.583 max 8.678 mean 8.670	1293		GPU 64 COO min 3.390 max 3.660 mean 3.585	
1220	Column-Gradient		1294		CSR min 0.960 max 4.330 mean 4.162	
1221		GPU 64 COO min 3.330 max 3.500 mean 3.416	1295		H min 9.689 max 9.689 mean 9.689	
1222		CSR min 6.730 max 7.540 mean 7.199	1296	Row-Premute		
1223		H min 9.542 max* 9.604 mean 9.581	1297		GPU 64 COO min 3.310 max 3.600 mean 3.488	
1224	Row-Column-Permute		1298		CSR min 0.620 max 4.290 mean 4.132	
1225		GPU 64 COO min 3.290 max 3.430 mean 3.365	1299		H min 10.738 max 10.743 mean 10.740	
1226		CSR min 7.390 max 8.060 mean 7.832	1300	Row-Gradient		
1227		H min 9.255 max 9.258 mean 9.256	1301		GPU 64 COO min 3.310 max* 3.670 mean 3.593	
1228	Maragal_6.mtx		1302		CSR min 4.130 max* 4.430 mean 4.331	
1229	Regular		1303		H min 10.576 max 10.584 mean 10.580	
1230		GPU 64 COO min 4.160 max 4.310 mean 4.217	1304	Column-Gradient		
1231		CSR min 4.940 max 4.960 mean 4.956	1305		GPU 64 COO min 0.550 max 3.660 mean 3.486	
1232		H min 9.930 max 9.930 mean 9.930	1306		CSR min 3.890 max 4.410 mean 4.275	
1233	Row-Premute		1307		H min 10.831 max*10.843 mean 10.836	
1234		GPU 64 COO min 4.220 max 4.240 mean 4.225	1308	Row-Column-Permute		
1235		CSR min 4.750 max*13.040 mean 5.133	1309		GPU 64 COO min 3.470 max 3.590 mean 3.542	
1236		H min 10.776 max 10.778 mean 10.777	1310		CSR min 4.190 max 4.290 mean 4.242	
1237	Row-Gradient		1311		H min 10.738 max 10.742 mean 10.740	
1238		GPU 64 COO min 4.180 max* 4.450 mean 4.245	1312	mult_dcop_03.mtx		
1239		CSR min 4.880 max 4.940 mean 4.915	1313	Regular		
1240		H min 11.259 max*11.302 mean 11.281	1314		GPU 64 COO min 3.360 max* 3.660 mean 3.550	
1241	Column-Gradient		1315		CSR min 3.650 max 4.090 mean 3.813	
1242		GPU 64 COO min 4.200 max 4.250 mean 4.236	1316		H min 9.689 max 9.689 mean 9.689	
1243		CSR min 4.800 max 4.890 mean 4.859	1317	Row-Premute		
1244		H min 12.022 max 12.073 mean 12.051	1318		GPU 64 COO min 3.450 max 3.580 mean 3.521	
1245	Row-Column-Permute		1319		CSR min 3.610 max 4.160 mean 3.784	
1246		GPU 64 COO min 4.210 max 4.230 mean 4.222	1320		H min 10.738 max 10.743 mean 10.740	
1247		CSR min 4.860 max 4.890 mean 4.887	1321	Row-Gradient		
1248		H min 10.776 max 10.778 mean 10.778	1322		GPU 64 COO min 3.470 max 3.660 mean 3.572	
1249	mhd4800a.mtx		1323		CSR min 3.640 max 4.190 mean 3.809	
1250	Regular		1324		H min 10.572 max 10.584 mean 10.580	
1251		GPU 64 COO min 4.570 max* 4.710 mean 4.608	1325	Column-Gradient		
1252		CSR min 12.690 max*13.940 mean 13.369	1326		GPU 64 COO min 3.430 max 3.650 mean 3.562	
1253		H min 7.132 max 7.132 mean 7.132	1327		CSR min 3.670 max* 4.290 mean 3.793	

1328		H	min 10.828 max*10.840 mean 10.834	1402		GPU 64 COO min 4.540 max 4.940 mean 4.874	
1329	Row-Column-Permute			1403		CSR min 6.280 max 6.520 mean 6.403	
1330		GPU 64 COO min 3.370 max 3.610 mean 3.502		1404	H	min 10.042 max 10.047 mean 10.044	
1331		CSR min 3.610 max 3.970 mean 3.744		1405	Row-Gradient		
1332		H	min 10.738 max 10.741 mean 10.740	1406		GPU 64 COO min 4.830 max 4.930 mean 4.875	
1333	OPF_3754.mtx			1407		CSR min 5.790 max* 6.560 mean 6.289	
1334	Regular			1408	H	min 9.675 max 9.706 mean 9.692	
1335		GPU 64 COO min 4.700 max* 4.930 mean 4.842		1409	Column-Gradient		
1336		CSR min 6.230 max* 6.600 mean 6.411		1410		GPU 64 COO min 4.790 max* 4.960 mean 4.880	
1337		H	min 8.393 max 8.393 mean 8.393	1411		CSR min 5.760 max 6.450 mean 6.204	
1338	Row-Premute			1412	H	min 10.601 max*10.661 mean 10.626	
1339		GPU 64 COO min 4.620 max 4.890 mean 4.787		1413	Row-Column-Permute		
1340		CSR min 5.780 max 6.310 mean 6.192		1414		GPU 64 COO min 4.330 max 4.950 mean 4.845	
1341		H	min 11.265 max 11.272 mean 11.269	1415		CSR min 5.740 max 6.500 mean 6.375	
1342	Row-Gradient			1416	H	min 10.041 max 10.046 mean 10.044	
1343		GPU 64 COO min 4.570 max 4.870 mean 4.776		1417	TSOPF_RS_b39_c7.mtx		
1344		CSR min 5.770 max 6.510 mean 6.302		1418	Regular		
1345		H	min 10.464 max 10.473 mean 10.468	1419		GPU 64 COO min 4.300 max* 4.430 mean 4.364	
1346	Column-Gradient			1420		CSR min 4.480 max 4.750 mean 4.716	
1347		GPU 64 COO min 4.580 max 4.870 mean 4.756		1421	H	min 7.304 max 7.304 mean 7.304	
1348		CSR min 5.630 max 6.180 mean 6.055		1422	Row-Premute		
1349		H	min 11.394 max*11.401 mean 11.397	1423		GPU 64 COO min 4.260 max 4.400 mean 4.353	
1350	Row-Column-Permute			1424		CSR min 4.490 max 4.770 mean 4.734	
1351		GPU 64 COO min 4.610 max 4.900 mean 4.780		1425	H	min 10.536 max 10.541 mean 10.539	
1352		CSR min 5.010 max 6.300 mean 6.113		1426	Row-Gradient		
1353		H	min 11.268 max 11.272 mean 11.270	1427		GPU 64 COO min 3.970 max 4.420 mean 4.338	
1354	OPF_6000.mtx			1428		CSR min 4.620 max* 4.820 mean 4.789	
1355	Regular			1429		H	min 9.638 max 9.644 mean 9.641
1356		GPU 64 COO min 3.780 max* 3.920 mean 3.864		1430	Column-Gradient		
1357		CSR min 4.270 max 4.360 mean 4.332		1431		GPU 64 COO min 4.240 max 4.430 mean 4.368	
1358		H	min 8.799 max 8.799 mean 8.799	1432		CSR min 4.710 max 4.770 mean 4.736	
1359	Row-Premute			1433	H	min 11.129 max*11.222 mean 11.205	
1360		GPU 64 COO min 3.770 max 3.870 mean 3.821		1434	Row-Column-Permute		
1361		CSR min 3.970 max*11.050 mean 4.439		1435		GPU 64 COO min 4.260 max 4.410 mean 4.359	
1362		H	min 11.872 max 11.877 mean 11.875	1436		CSR min 4.660 max 4.760 mean 4.738	
1363	Row-Gradient			1437	H	min 10.537 max 10.541 mean 10.539	
1364		GPU 64 COO min 3.700 max 3.870 mean 3.795					
1365		CSR min 4.330 max 4.440 mean 4.403					
1366		H	min 11.109 max 11.116 mean 11.113				
1367	Column-Gradient			1438	11 FIJI		
1368		GPU 64 COO min 3.690 max 3.870 mean 3.804		1439	mult_dcop_03.mtx		
1369		CSR min 4.260 max 4.340 mean 4.308		1440	Regular		
1370		H	min 12.041 max*12.045 mean 12.043	1441		GPU 64 COO min 5.140 max* 5.140 mean 5.140	
1371	Row-Column-Permute			1442		CSR min 10.340 max*10.390 mean 10.365	
1372		GPU 64 COO min 3.780 max 3.860 mean 3.819		1443	H	min 9.689 max 9.689 mean 9.689	
1373		CSR min 4.090 max 4.290 mean 4.259		1444	Row-Premute		
1374		H	min 11.873 max 11.877 mean 11.876	1445		GPU 64 COO min 4.970 max 4.990 mean 4.980	
1375	shermanACb.mtx			1446		CSR min 9.420 max 9.430 mean 9.425	
1376	Regular			1447	H	min 10.739 max 10.739 mean 10.739	
1377		GPU 64 COO min 2.920 max* 3.140 mean 3.048		1448	Row-Gradient		
1378		CSR min 5.550 max 5.980 mean 5.803		1449		GPU 64 COO min 5.080 max 5.090 mean 5.085	
1379		H	min 8.600 max 8.600 mean 8.600	1450		CSR min 9.720 max 10.300 mean 10.010	
1380	Row-Premute			1451	H	min 10.579 max 10.582 mean 10.580	
1381		GPU 64 COO min 2.760 max 3.020 mean 2.898		1452	Column-Gradient		
1382		CSR min 2.660 max 5.830 mean 5.632		1453		GPU 64 COO min 5.030 max 5.120 mean 5.075	
1383		H	min 10.377 max 10.381 mean 10.379	1454		CSR min 9.330 max 9.770 mean 9.550	
1384	Row-Gradient			1455	H	min 10.835 max*10.838 mean 10.836	
1385		GPU 64 COO min 2.800 max 3.040 mean 2.944		1456	Row-Column-Permute		
1386		CSR min 5.330 max* 6.020 mean 5.742		1457		GPU 64 COO min 5.000 max 5.010 mean 5.005	
1387		H	min 9.919 max 9.925 mean 9.922	1458		CSR min 7.580 max 9.460 mean 8.520	
1388	Column-Gradient			1459	H	min 10.739 max 10.741 mean 10.740	
1389		GPU 64 COO min 2.720 max 3.010 mean 2.926		1460	mult_dcop_03.mtx		
1390		CSR min 0.000 max 5.840 mean 5.513		1461	Regular		
1391		H	min 10.587 max*10.596 mean 10.591	1462		GPU 64 COO min 5.140 max* 5.140 mean 5.140	
1392	Row-Column-Permute			1463		CSR min 10.340 max*10.390 mean 10.365	
1393		GPU 64 COO min 2.780 max 3.030 mean 2.939		1464	H	min 9.689 max 9.689 mean 9.689	
1394		CSR min 4.860 max 5.810 mean 5.667		1465	Row-Premute		
1395		H	min 10.376 max 10.382 mean 10.379	1466		GPU 64 COO min 4.970 max 4.990 mean 4.980	
1396	TSOPF_FS_b9_c6.mtx			1467		CSR min 9.420 max 9.430 mean 9.425	
1397	Regular			1468	H	min 10.739 max 10.739 mean 10.739	
1398		GPU 64 COO min 0.000 max 0.000 mean 0.000		1469	Row-Gradient		
1399		CSR min 0.000 max 0.000 mean 0.000		1470		GPU 64 COO min 5.080 max 5.090 mean 5.085	
1400		H	min 7.380 max 7.380 mean 7.380	1471		CSR min 9.720 max 10.300 mean 10.010	
1401	Row-Premute			1472	H	min 10.579 max 10.582 mean 10.580	

1473	Column-Gradient					1547			CSR min 6.360 max 7.450 mean 6.711
1474		GPU 64 COO min 5.030 max 5.120 mean 5.075				1548	H		min 11.109 max 11.109 mean 11.109
1475		CSR min 9.330 max 9.770 mean 9.550				1549	Row-Premute		
1476		H min 10.835 max*10.838 mean 10.836				1550		GPU 64 COO min 3.950 max* 3.980 mean 3.953	
1477	Row-Column-Permute					1551		CSR min 6.330 max 7.400 mean 6.661	
1478		GPU 64 COO min 5.000 max 5.010 mean 5.005				1552	H		min 11.098 max 11.104 mean 11.101
1479		CSR min 7.580 max 9.460 mean 8.520				1553	Row-Gradient		
1480		H min 10.739 max 10.741 mean 10.740				1554		GPU 64 COO min 3.960 max 3.980 mean 3.961	
1481	mult_dcop_03.mtx					1555		CSR min 6.270 max*10.770 mean 7.017	
1482	Regular					1556	H		min 11.109 max 11.109 mean 11.109
1483		GPU 64 COO min 5.130 max* 5.220 mean 5.142				1557	Column-Gradient		
1484		CSR min 7.250 max* 9.320 mean 7.722				1558		GPU 64 COO min 3.940 max 3.960 mean 3.950	
1485		H min 9.689 max 9.689 mean 9.689				1559		CSR min 6.270 max 7.370 mean 6.696	
1486	Row-Premute					1560	H		min 11.329 max*11.334 mean 11.331
1487		GPU 64 COO min 4.980 max 5.030 mean 4.999				1561	Row-Column-Permute		
1488		CSR min 6.460 max 8.470 mean 6.950				1562		GPU 64 COO min 3.950 max 3.960 mean 3.952	
1489		H min 10.738 max 10.742 mean 10.740				1563		CSR min 6.180 max 7.420 mean 6.641	
1490	Row-Gradient					1564	H		min 11.098 max 11.105 mean 11.101
1491		GPU 64 COO min 5.070 max 5.140 mean 5.088				1565	bloweya.mtx		
1492		CSR min 6.780 max 8.700 mean 7.268				1566	Regular		
1493		H min 10.572 max 10.584 mean 10.580				1567		GPU 64 COO min 0.000 max 0.000 mean 0.000	
1494	Column-Gradient					1568		CSR min 0.000 max 0.000 mean 0.000	
1495		GPU 64 COO min 4.980 max 5.030 mean 5.010				1569	H		min 7.205 max 7.205 mean 7.205
1496		CSR min 6.390 max 7.640 mean 6.982				1570	Row-Premute		
1497		H min 10.825 max*10.845 mean 10.836				1571		GPU 64 COO min 4.020 max 4.030 mean 4.023	
1498	Row-Column-Permute					1572		CSR min 6.070 max 6.750 mean 6.340	
1499		GPU 64 COO min 4.990 max 5.010 mean 4.997				1573	H		min 11.025 max 11.031 mean 11.028
1500		CSR min 6.300 max 7.160 mean 6.636				1574	Row-Gradient		
1501		H min 10.738 max 10.743 mean 10.740				1575		GPU 64 COO min 4.090 max* 4.160 mean 4.111	
1502	mult_dcop_01.mtx					1576		CSR min 5.980 max* 7.370 mean 6.678	
1503	Regular					1577	H		min 10.295 max 10.304 mean 10.300
1504		GPU 64 COO min 5.120 max* 5.140 mean 5.134				1578	Column-Gradient		
1505		CSR min 6.990 max* 9.230 mean 7.546				1579		GPU 64 COO min 3.980 max 4.010 mean 3.995	
1506		H min 9.689 max 9.689 mean 9.689				1580		CSR min 5.880 max 6.780 mean 6.295	
1507	Row-Premute					1581	H		min 10.881 max*10.887 mean 10.883
1508		GPU 64 COO min 4.990 max 5.020 mean 5.004				1582	Row-Column-Permute		
1509		CSR min 6.370 max 7.220 mean 6.771				1583		GPU 64 COO min 4.020 max 4.030 mean 4.023	
1510		H min 10.738 max 10.743 mean 10.740				1584		CSR min 5.970 max 6.420 mean 6.183	
1511	Row-Gradient					1585	H		min 11.025 max 11.033 mean 11.028
1512		GPU 64 COO min 5.060 max 5.100 mean 5.082				1586	lp_osa_07.mtx		
1513		CSR min 6.730 max 7.720 mean 7.317				1587	Regular		
1514		H min 10.574 max 10.585 mean 10.580				1588		GPU 64 COO min 4.260 max* 4.270 mean 4.261	
1515	Column-Gradient					1589		CSR min 6.440 max 7.640 mean 6.863	
1516		GPU 64 COO min 4.980 max 5.100 mean 5.012				1590	H		min 8.412 max 8.412 mean 8.412
1517		CSR min 6.580 max 7.510 mean 7.054				1591	Row-Premute		
1518		H min 10.828 max*10.842 mean 10.835				1592		GPU 64 COO min 4.200 max 4.200 mean 4.200	
1519	Row-Column-Permute					1593		CSR min 6.020 max 7.030 mean 6.418	
1520		GPU 64 COO min 4.970 max 5.000 mean 4.986				1594	H		min 9.255 max 9.257 mean 9.256
1521		CSR min 6.390 max 7.050 mean 6.677				1595	Row-Gradient		
1522		H min 10.738 max 10.742 mean 10.740				1596		GPU 64 COO min 4.210 max 4.240 mean 4.226	
1523	mult_dcop_02.mtx					1597		CSR min 6.070 max*10.050 mean 6.498	
1524	Regular					1598	H		min 8.607 max 8.678 mean 8.671
1525		GPU 64 COO min 5.120 max 5.140 mean 5.133				1599	Column-Gradient		
1526		CSR min 6.950 max 7.590 mean 7.336				1600		GPU 64 COO min 4.170 max 4.190 mean 4.180	
1527		H min 9.689 max 9.689 mean 9.689				1601		CSR min 5.610 max 7.300 mean 5.988	
1528	Row-Premute					1602	H		min 9.534 max* 9.601 mean 9.585
1529		GPU 64 COO min 4.970 max 4.990 mean 4.984				1603	Row-Column-Permute		
1530		CSR min 6.440 max 7.110 mean 6.719				1604		GPU 64 COO min 4.190 max 4.190 mean 4.190	
1531		H min 10.738 max 10.742 mean 10.740				1605		CSR min 6.070 max 7.000 mean 6.386	
1532	Row-Gradient					1606	H		min 9.255 max 9.257 mean 9.256
1533		GPU 64 COO min 5.070 max* 5.150 mean 5.086				1607	ex19.mtx		
1534		CSR min 6.650 max* 7.930 mean 7.304				1608	Regular		
1535		H min 10.574 max 10.587 mean 10.580				1609		GPU 64 COO min 6.140 max* 6.180 mean 6.159	
1536	Column-Gradient					1610		CSR min 12.780 max*14.400 mean 13.328	
1537		GPU 64 COO min 4.980 max 5.040 mean 5.012				1611	H		min 8.228 max 8.228 mean 8.228
1538		CSR min 6.520 max 7.650 mean 7.139				1612	Row-Premute		
1539		H min 10.829 max*10.846 mean 10.836				1613		GPU 64 COO min 5.820 max 5.850 mean 5.833	
1540	Row-Column-Permute					1614		CSR min 9.870 max 11.070 mean 10.372	
1541		GPU 64 COO min 4.970 max 5.050 mean 4.983				1615	H		min 11.836 max 11.840 mean 11.838
1542		CSR min 6.440 max 7.380 mean 6.779				1616	Row-Gradient		
1543		H min 10.738 max 10.743 mean 10.740				1617		GPU 64 COO min 6.070 max 6.120 mean 6.104	
1544	lp_fit2d.mtx					1618		CSR min 11.290 max 12.760 mean 12.088	
1545	Regular					1619	H		min 10.743 max 10.752 mean 10.748
1546		GPU 64 COO min 3.960 max 3.960 mean 3.960				1620	Column-Gradient		

1621		GPU 64 COO min 5.760 max 5.840 mean 5.813	1695		H min 7.380 max 7.380 mean 7.380
1622		CSR min 9.710 max 14.220 mean 10.376	1696	Row-Premute	
1623		H min 11.873 max*11.882 mean 11.878	1697		GPU 64 COO min 4.130 max 4.170 mean 4.134
1624	Row-Column-Permute		1698		CSR min 6.180 max* 9.200 mean 6.796
1625		GPU 64 COO min 5.810 max 5.860 mean 5.838	1699		H min 10.041 max 10.046 mean 10.044
1626		CSR min 9.920 max 10.820 mean 10.240	1700	Row-Gradient	
1627		H min 11.836 max 11.841 mean 11.838	1701		GPU 64 COO min 4.150 max* 4.220 mean 4.163
1628	brainpc2.mtx		1702		CSR min 6.410 max 7.500 mean 6.816
1629	Regular		1703		H min 9.682 max 9.706 mean 9.693
1630		GPU 64 COO min 0.000 max 0.000 mean 0.000	1704	Column-Gradient	
1631		CSR min 0.000 max 0.000 mean 0.000	1705		GPU 64 COO min 4.080 max 4.110 mean 4.096
1632		H min 7.478 max 7.478 mean 7.478	1706		CSR min 6.020 max 7.220 mean 6.309
1633	Row-Premute		1707		H min 10.597 max*10.658 mean 10.631
1634		GPU 64 COO min 4.760 max 4.790 mean 4.773	1708	Row-Column-Permute	
1635		CSR min 6.930 max 7.780 mean 7.310	1709		GPU 64 COO min 4.120 max 4.140 mean 4.130
1636		H min 9.810 max 9.813 mean 9.811	1710		CSR min 6.210 max 7.200 mean 6.609
1637	Row-Gradient		1711		H min 10.041 max 10.046 mean 10.044
1638		GPU 64 COO min 4.820 max* 4.840 mean 4.831	1712	TSOPF_FS_b9_c6.mtx	
1639		CSR min 7.220 max 8.290 mean 7.583	1713	Regular	
1640		H min 9.721 max 9.725 mean 9.723	1714		GPU 64 COO min 0.000 max 0.000 mean 0.000
1641	Column-Gradient		1715		CSR min 0.000 max 0.000 mean 0.000
1642		GPU 64 COO min 4.760 max 4.820 mean 4.779	1716		H min 7.380 max 7.380 mean 7.380
1643		CSR min 6.870 max* 8.300 mean 7.393	1717	Row-Premute	
1644		H min 10.368 max*10.373 mean 10.370	1718		GPU 64 COO min 4.120 max 4.140 mean 4.129
1645	Row-Column-Permute		1719		CSR min 6.170 max 7.160 mean 6.664
1646		GPU 64 COO min 4.750 max 4.780 mean 4.765	1720		H min 10.041 max 10.045 mean 10.043
1647		CSR min 6.940 max 7.580 mean 7.298	1721	Row-Gradient	
1648		H min 9.809 max 9.814 mean 9.811	1722		GPU 64 COO min 4.150 max* 4.180 mean 4.162
1649	shermanACb.mtx		1723		CSR min 6.420 max 7.360 mean 6.723
1650	Regular		1724		H min 9.682 max 9.706 mean 9.693
1651		GPU 64 COO min 4.090 max* 4.130 mean 4.112	1725	Column-Gradient	
1652		CSR min 6.320 max* 7.200 mean 6.779	1726		GPU 64 COO min 4.080 max 4.120 mean 4.096
1653		H min 8.600 max 8.600 mean 8.600	1727		CSR min 5.880 max 7.090 mean 6.403
1654	Row-Premute		1728		H min 10.611 max*10.660 mean 10.637
1655		GPU 64 COO min 4.020 max 4.050 mean 4.036	1729	Row-Column-Permute	
1656		CSR min 5.670 max 6.460 mean 6.014	1730		GPU 64 COO min 4.130 max 4.140 mean 4.130
1657		H min 10.376 max 10.382 mean 10.379	1731		CSR min 6.330 max* 7.390 mean 6.695
1658	Row-Gradient		1732		H min 10.042 max 10.047 mean 10.044
1659		GPU 64 COO min 4.050 max 4.100 mean 4.074	1733	OPF_6000.mtx	
1660		CSR min 5.580 max 6.420 mean 5.996	1734	Regular	
1661		H min 9.918 max 9.924 mean 9.921	1735		GPU 64 COO min 7.270 max* 7.370 mean 7.293
1662	Column-Gradient		1736		CSR min 12.890 max*14.500 mean 13.566
1663		GPU 64 COO min 4.010 max 4.080 mean 4.033	1737		H min 8.799 max 8.799 mean 8.799
1664		CSR min 0.000 max 6.320 mean 5.527	1738	Row-Premute	
1665		H min 10.543 max*10.595 mean 10.589	1739		GPU 64 COO min 6.640 max 6.720 mean 6.678
1666	Row-Column-Permute		1740		CSR min 9.680 max 11.600 mean 10.040
1667		GPU 64 COO min 4.020 max 4.050 mean 4.036	1741		H min 11.873 max 11.877 mean 11.875
1668		CSR min 5.670 max 6.510 mean 6.092	1742	Row-Gradient	
1669		H min 10.377 max 10.381 mean 10.379	1743		GPU 64 COO min 7.090 max 7.140 mean 7.122
1670	cvxqp3.mtx		1744		CSR min 11.250 max 13.030 mean 12.142
1671	Regular		1745		H min 11.110 max 11.117 mean 11.114
1672		GPU 64 COO min 3.500 max* 3.540 mean 3.501	1746	Column-Gradient	
1673		CSR min 11.860 max*13.100 mean 12.694	1747		GPU 64 COO min 6.590 max 6.710 mean 6.644
1674		H min 8.646 max 8.646 mean 8.646	1748		CSR min 9.400 max 13.140 mean 9.991
1675	Row-Premute		1749		H min 12.040 max*12.046 mean 12.043
1676		GPU 64 COO min 3.360 max 3.370 mean 3.365	1750	Row-Column-Permute	
1677		CSR min 6.210 max 7.610 mean 6.631	1751		GPU 64 COO min 6.640 max 6.710 mean 6.679
1678		H min 11.027 max 11.032 mean 11.030	1752		CSR min 9.690 max 10.740 mean 10.050
1679	Row-Gradient		1753		H min 11.874 max 11.877 mean 11.875
1680		GPU 64 COO min 3.370 max 3.380 mean 3.376	1754	OPF_3754.mtx	
1681		CSR min 6.170 max 7.070 mean 6.499	1755	Regular	
1682		H min 11.059 max 11.068 mean 11.064	1756		GPU 64 COO min 4.430 max* 4.450 mean 4.443
1683	Column-Gradient		1757		CSR min 9.710 max*13.000 mean 11.377
1684		GPU 64 COO min 3.350 max 3.390 mean 3.371	1758		H min 8.393 max 8.393 mean 8.393
1685		CSR min 6.150 max 7.180 mean 6.531	1759	Row-Premute	
1686		H min 11.125 max*11.133 mean 11.130	1760		GPU 64 COO min 4.230 max 4.250 mean 4.240
1687	Row-Column-Permute		1761		CSR min 7.430 max 8.750 mean 7.986
1688		GPU 64 COO min 3.350 max 3.380 mean 3.364	1762		H min 11.266 max 11.272 mean 11.269
1689		CSR min 6.040 max 7.440 mean 6.603	1763	Row-Gradient	
1690		H min 11.028 max 11.033 mean 11.030	1764		GPU 64 COO min 4.370 max 4.420 mean 4.382
1691	case9.mtx		1765		CSR min 8.160 max 9.470 mean 8.682
1692	Regular		1766		H min 10.462 max 10.473 mean 10.468
1693		GPU 64 COO min 0.000 max 0.000 mean 0.000	1767	Column-Gradient	
1694		CSR min 0.000 max 0.000 mean 0.000	1768		GPU 64 COO min 4.210 max 4.240 mean 4.227

1769		CSR min 7.160 max 8.080 mean 7.595	1843	Row-Premute	
1770		H min 11.394 max*11.401 mean 11.398	1844		GPU 64 COO min 10.340 max 10.430 mean 10.362
1771	Row-Column-Permute		1845		CSR min 12.880 max 13.340 mean 13.057
1772		GPU 64 COO min 4.230 max 4.250 mean 4.243	1846		H min 10.777 max 10.778 mean 10.777
1773		CSR min 7.230 max 8.940 mean 8.056	1847	Row-Gradient	
1774		H min 11.264 max 11.271 mean 11.269	1848		GPU 64 COO min 10.650 max*10.740 mean 10.688
1775	c-47.mtx		1849		CSR min 12.310 max 13.670 mean 12.562
1776	Regular		1850		H min 11.247 max 11.300 mean 11.281
1777		GPU 64 COO min 5.320 max* 5.340 mean 5.329	1851	Column-Gradient	
1778		CSR min 8.890 max* 9.590 mean 9.249	1852		GPU 64 COO min 10.340 max 10.440 mean 10.398
1779		H min 8.364 max 8.364 mean 8.364	1853		CSR min 9.480 max 10.110 mean 9.782
1780	Row-Premute		1854		H min 12.023 max*12.069 mean 12.047
1781		GPU 64 COO min 5.240 max 5.250 mean 5.241	1855	Row-Column-Permute	
1782		CSR min 7.790 max 8.890 mean 8.214	1856		GPU 64 COO min 10.330 max 10.380 mean 10.356
1783		H min 10.059 max 10.063 mean 10.061	1857		CSR min 12.840 max 13.530 mean 13.119
1784	Row-Gradient		1858		H min 10.776 max 10.778 mean 10.777
1785		GPU 64 COO min 5.230 max 5.260 mean 5.242	1859	aft01.mtx	
1786		CSR min 7.080 max 8.050 mean 7.673	1860	Regular	
1787		H min 10.206 max 10.226 mean 10.218	1861		GPU 64 COO min 3.680 max* 3.690 mean 3.688
1788	Column-Gradient		1862		CSR min 13.860 max*14.830 mean 14.560
1789		GPU 64 COO min 5.080 max 5.120 mean 5.105	1863		H min 7.811 max 7.811 mean 7.811
1790		CSR min 5.780 max 6.970 mean 6.359	1864	Row-Premute	
1791		H min 11.205 max*11.233 mean 11.222	1865		GPU 64 COO min 3.510 max 3.530 mean 3.513
1792	Row-Column-Permute		1866		CSR min 6.420 max 10.520 mean 7.265
1793		GPU 64 COO min 5.220 max 5.250 mean 5.227	1867		H min 11.161 max*11.170 mean 11.165
1794		CSR min 7.860 max 8.710 mean 8.247	1868	Row-Gradient	
1795		H min 10.059 max 10.064 mean 10.061	1869		GPU 64 COO min 3.630 max 3.670 mean 3.643
1796	mhd4800a.mtx		1870		CSR min 10.760 max 13.510 mean 12.199
1797	Regular		1871		H min 10.248 max 10.265 mean 10.258
1798		GPU 64 COO min 3.090 max* 3.100 mean 3.098	1872	Column-Gradient	
1799		CSR min 11.570 max*12.290 mean 12.092	1873		GPU 64 COO min 3.510 max 3.520 mean 3.519
1800		H min 7.132 max 7.132 mean 7.132	1874		CSR min 6.490 max 11.230 mean 7.645
1801	Row-Premute		1875		H min 11.112 max 11.121 mean 11.117
1802		GPU 64 COO min 3.020 max 3.020 mean 3.020	1876	Row-Column-Permute	
1803		CSR min 5.560 max 7.270 mean 6.007	1877		GPU 64 COO min 3.510 max 3.540 mean 3.515
1804		H min 10.959 max*10.968 mean 10.963	1878		CSR min 6.510 max 11.650 mean 7.311
1805	Row-Gradient		1879		H min 11.161 max 11.168 mean 11.165
1806		GPU 64 COO min 3.080 max 3.100 mean 3.088	1880	TSOPF_RS_b39_c7.mtx	
1807		CSR min 10.250 max 12.150 mean 11.340	1881	Regular	
1808		H min 9.509 max 9.528 mean 9.520	1882		GPU 64 COO min 5.970 max* 6.010 mean 5.988
1809	Column-Gradient		1883		CSR min 12.470 max*21.120 mean 13.816
1810		GPU 64 COO min 3.020 max 3.050 mean 3.026	1884		H min 7.304 max 7.304 mean 7.304
1811		CSR min 5.530 max 10.580 mean 6.432	1885	Row-Premute	
1812		H min 10.933 max 10.946 mean 10.939	1886		GPU 64 COO min 5.840 max 5.870 mean 5.856
1813	Row-Column-Permute		1887		CSR min 10.780 max 15.810 mean 11.425
1814		GPU 64 COO min 3.020 max 3.020 mean 3.020	1888		H min 10.537 max 10.540 mean 10.539
1815		CSR min 5.510 max 6.830 mean 6.136	1889	Row-Gradient	
1816		H min 10.959 max 10.967 mean 10.963	1890		GPU 64 COO min 5.950 max 6.000 mean 5.975
1817	gen4.mtx		1891		CSR min 11.520 max 17.250 mean 12.799
1818	Regular		1892		H min 9.638 max 9.646 mean 9.641
1819		GPU 64 COO min 3.300 max* 3.320 mean 3.308	1893	Column-Gradient	
1820		CSR min 5.250 max 6.340 mean 5.705	1894		GPU 64 COO min 5.790 max 5.860 mean 5.827
1821		H min 9.234 max 9.234 mean 9.234	1895		CSR min 10.500 max 14.080 mean 11.237
1822	Row-Premute		1896		H min 11.128 max*11.223 mean 11.209
1823		GPU 64 COO min 3.290 max 3.310 mean 3.299	1897	Row-Column-Permute	
1824		CSR min 5.190 max 7.420 mean 5.683	1898		GPU 64 COO min 5.850 max 5.870 mean 5.855
1825		H min 10.249 max 10.254 mean 10.252	1899		CSR min 10.790 max 15.250 mean 11.718
1826	Row-Gradient		1900		H min 10.537 max 10.541 mean 10.539
1827		GPU 64 COO min 3.300 max 3.310 mean 3.301	1901	mult_dcop_03.mtx	
1828		CSR min 5.370 max 6.310 mean 5.659	1902	Regular	
1829		H min 9.934 max 9.958 mean 9.948	1903		GPU 64 COO min 5.130 max* 5.220 mean 5.142
1830	Column-Gradient		1904		CSR min 7.250 max* 9.320 mean 7.722
1831		GPU 64 COO min 3.240 max 3.260 mean 3.249	1905		H min 9.689 max 9.689 mean 9.689
1832		CSR min 5.090 max* 8.660 mean 5.546	1906	Row-Premute	
1833		H min 10.853 max*10.873 mean 10.864	1907		GPU 64 COO min 4.980 max 5.030 mean 4.999
1834	Row-Column-Permute		1908		CSR min 6.460 max 8.470 mean 6.950
1835		GPU 64 COO min 3.290 max 3.320 mean 3.296	1909		H min 10.738 max 10.742 mean 10.740
1836		CSR min 5.190 max 7.550 mean 5.659	1910	Row-Gradient	
1837		H min 10.249 max 10.255 mean 10.252	1911		GPU 64 COO min 5.070 max 5.140 mean 5.088
1838	Maragal_6.mtx		1912		CSR min 6.780 max 8.700 mean 7.268
1839	Regular		1913		H min 10.572 max 10.584 mean 10.580
1840		GPU 64 COO min 10.580 max 10.620 mean 10.599	1914	Column-Gradient	
1841		CSR min 15.620 max*16.470 mean 15.832	1915		GPU 64 COO min 4.980 max 5.030 mean 5.010
1842		H min 9.930 max 9.930 mean 9.930	1916		CSR min 6.390 max 7.640 mean 6.982

1917		H	min 10.825 max 10.845 mean 10.836	1972
1918	Row-Column-Permute			1973
1919		GPU 64 COO min 4.990 max 5.010 mean 4.997		1974
1920		CSR min 6.300 max 7.160 mean 6.636		1975
1921		H min 10.738 max 10.743 mean 10.740		1976

REFERENCES

- [1] Hartwig Anzt, Terry Cojean, Chen Yen-Chen, Jack J. Dongarra, Goran Flegar, Pratik Nayak, Stanimire Tomov, Yuhsiang M. Tsai, and Weichung Wang. 2020. Load-balancing Sparse Matrix Vector Product Kernels on GPUs. *ACM Trans. Parallel Comput.* 7, 1 (2020), 2:1–2:26. <https://doi.org/10.1145/3380930>
- [2] Paolo D'Alberto, Chris Drome, and Ali Dasdan. 2012. Non-Parametric Methods Applied to the N-Sample Series Comparison. (2012). arXiv:stat.CO/1205.1880
- [3] Jeremy Fowers, Kalin Ovtcharov, Karin Strauss, Eric S Chung, and Greg Stitt. 2014. A high memory bandwidth fpga accelerator for sparse matrix-vector multiplication. In *IEEE Symposium on FPGAs for Custom Computing Machines (FCCM)*. IEEE, 36–43.
- [4] Paul Grigoras, Pavel Burovskiy, and Wayne Luk. 2016. CASK: Open-Source Custom Architectures for Sparse Kernels. In *Proceedings of the International Symposium on Field Programmable Gate Arrays (FPGA)*. ACM, 179–184.
- [5] B. Huang, J. Gao, and X. Li. 2009. An Empirically Optimized Radix Sort for GPU. In *2009 IEEE International Symposium on Parallel and Distributed Processing with Applications*. 234–241.
- [6] Enver Kayaaslan, Cevdet Aykanat, and Bora Uçar. 2018. 1.5D Parallel Sparse Matrix-Vector Multiply. *SIAM J. Scientific Computing* 40, 1 (2018). <https://doi.org/10.1137/16M1105591>
- [7] Srinidhi Kestur, John D Davis, and Eric S Chung. 2012. Towards a universal FPGA matrix-vector multiplication architecture. In *IEEE Symposium on FPGAs for Custom Computing Machines (FCCM)*. IEEE, 9–16.
- [8] Sicheng Li, Yandan Wang, Wujie Wen, Yu Wang, Yiran Chen, and Hai Li. 2016. A data locality-aware design framework for reconfigurable sparse matrix-vector multiplication kernel. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–6.
- [9] X. Li, M. J. Garzaran, and D. Padua. 2004. A dynamically tuned sorting library. In *International Symposium on Code Generation and Optimization, 2004. CGO 2004*. 111–122.
- [10] Krishna K Nagar and Jason D Bakos. 2011. A sparse matrix personality for the convey HC-1. In *IEEE Symposium on FPGAs for Custom Computing Machines (FCCM)*. IEEE, 1–8.
- [11] Brian A. Page and Peter M. Kogge. 2018. Scalability of Hybrid Sparse Matrix Dense Vector (SpMV) Multiplication. In *2018 International Conference on High Performance Computing & Simulation, HPCS 2018, Orleans, France, July 16-20, 2018*. IEEE, 406–414. <https://doi.org/10.1109/HPCS.2018.00072>
- [12] Kevin Townsend and Joseph Zambreno. 2013. Reduce, reuse, recycle (r3): A design methodology for sparse matrix vector multiplication on reconfigurable platforms. In *Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*. IEEE, 185–191.
- [13] Shijie Zhou, Rajgopal Kannan, Viktor K Prasanna, Guna Seetharaman, and Qing Wu. 2019. HitGraph: High-throughput Graph Processing Framework on FPGA. *IEEE Transactions on Parallel and Distributed Systems* 30, 10 (2019), 2249–2264.
- [14] Ling Zhuo and Viktor K Prasanna. 2005. Sparse matrix-vector multiplication on FPGAs. In *Proceedings of the International Symposium on Field Programmable Gate Arrays (FPGA)*. ACM, 63–74.