Quantum chemistry in dataflow: Density-Fitting MP2

Bridgette Cooper,[*,†] Stephen Girdlestone,[‡] Pavel Burovskiy,[‡] Georgi Gaydadjiev,[‡]
Vitali Averbukh,[†] Peter J. Knowles,[¶] and Wayne Luk[§]

[†]*Department of Physics, Blackett Laboratory, Imperial College London, Prince Consort Road, SW7 2AZ London, United Kingdom*
[‡]*Maxeler Technologies Ltd., UK*
[¶]*School of Chemistry, Cardiff University, Main Building, Park Place, Cardiff, CF10 3AT, United Kingdom*
[§]*Department of Computing, Imperial College London, 180 Queen's Gate, SW7 2AZ, UK*

E-mail: b.cooper@ic.ac.uk

Abstract

We demonstrate the use of dataflow technology in the computation of the correlation energy in molecules at the Møller Plesset perturbation theory (MP2) level. Specifically, we benchmark density fitting (DF) - MP2 for as many as 168 atoms (in valinomycin) and show that speed ups between 3 to 3.8 times can be achieved when compared to the MOLPRO package run on a single CPU. Acceleration is achieved by offloading the matrix multiplications steps in DF-MP2 to Dataflow Engines (DFEs). We project that the acceleration factor could be as much as 24 with the next generation of DFEs.

Keywords

Dataflow computing, DFE, DF-MP2, density fitting, Møller Plesset perturbation theory

1 Introduction

Recently advances in dataflow computing have enabled significant acceleration of many computationally demanding applications.[1] Moreover, dataflow computing is becoming available from cloud service providers, such as the Amazon EC2 F1 instances. This paper investigates the potential of dataflow computing for speeding up calculations in quantum chemistry.

Ab-initio quantum chemical methods for determining the electronic structure of molecules are powerful tools to elucidate electronic states, molecular structures, properties and reaction mechanisms. Accurately accounting for the electron correlation in a molecular system is essential to robustly recover properties relevant in chemistry and biochemistry. In particular dispersion type effects (arising from long range electron correlation) which can strongly affect the molecular structure of a large biomolecule cannot be accounted for in simple mean-field approaches such as Hartree-Fock (HF) or most density functional theories. This highlights the importance of post HF methods such as those based on correcting for the electron correlation perturbatively or with a configuration interaction (CI) expansion of the wavefunction.

Second order Møller Plesset perturbation theory (MP2) is the simplest post Hartree-Fock method that can account for dynamic correlation in molecules, and is used extensively. MP2 is not the most accurate method for accounting for electron correlation effects in molecules, however it provides suitable estimates as it recovers 80-120% of the basis set limit correlation energy. However, the application of MP2 to larger molecules becomes quickly prohibitive due to the steep scaling of the computational effort with molecular size. The computational bottleneck is the need to transform two electron Coulomb repulsion integrals from an atomic orbital (AO) basis to a molecular orbital (MO) basis. This operation scales as $\mathcal{O}(N^5)$ with the number of basis functions N.

To reduce the computational effort in calculating second order MP2 energies and thus access larger molecules, a multitude of approaches have been previously explored, reviewed by Cremer.[2] Much of the previous work has been to formulate linear-scaling approximations for example by using internal localised orbitals for occupied and atomic orbitals for the virtual[3,4] classifying the electron pair interactions and thus neglecting very small interactions between very distance pairs[5,6] or neglecting integrals without one atom center in common in an atoms in molecules approach.[7,8] These methods have been extremely successful in the calculation of local correlations. However, when the selection of integrals is distance based, often the resulting MP2 energies can be adversely affected, for example dependent on the system size, or the calculated potential energy surfaces may contain discontinuities.[9] Stochastic sampling can be employed to efficiently calculate MP2 energies on very large nanocrystals,[10–12] however at the cost of accuracy as statistical error is introduced.

Other methods with reduced scaling behaviour and/or reduced prefactors include Laplace- transformed techniques,[13–17] divide and conquer approaches,[18,19] Cholesky decomposition,[20] pseudospectral approaches,[21] scaled opposite spin Møller Plesset method (SOS-MP2)[22] or resolution of the identity (RI) also called density-fitting (DF) approaches.[23–25]

As well as developing robust alternative theories, one can also take advantage of the developments in computer architecture to accelerate quantum chemical calculations. Much effort has been placed in using Graphical Processing Units (GPUs) to speed up computational chemistry calculations[26,27] particularly strategies for acceleration of the calculation of electron integrals[28] and MP2.[29–31] Interest in using GPUs as co-processors for quantum chemistry calculations has increased in recent years as double precision arithmetic is now supported and programming has become easier with CUDA and OpenCL. There are also highly parallel implementations[32–35] allowing one to calculate MP2 energies on systems as large nanographene sheets[32] and tetrapeptides,[33] as well as combining both highly parallel CPU and GPU heterogeneous architectures to achieve significant speed ups[36,37] on nano-sized molecules.

In this paper we exploit dataflow computing through the 4-th generation Dataflow Engine (DFE) MAX4 from Maxeler Technologies[38] to accelerate Møller Plesset perturbation theory to second order (MP2), specifically the density-fitting (DF-MP2) algorithm. In the following section, we will detail dataflow technology, and its computational advantages. As will be shown in Section 3, the computational effort of the DF-MP2 algorithm is dominated by linear algebra, in particular matrix multiplication. We will describe how matrix multiplication is optimised for dataflow computing in Section 4. By accelerating just this part of the algorithm we will demonstrate up to 3.8 times better performance compared to a single CPU core, as well as showing how the algorithm compares for parallel executions in Section 5.

2 Dataflow Computing

Maxeler Dataflow Engines (DFEs) are FPGA-based accelerators that are designed to deliver high performance and energy efficiency for large-scale HPC applications.[38] This is achieved by adopting a dataflow-oriented highly customisable computing model, an evolution of the dataflow and systolic array concepts,[39,40] which fundamentally differs from the classical Von Neumann control-flow oriented paradigm used in general purpose computing systems. On DFEs, data is streamed from memory (or the CPU) to the reconfigurable compute chip (FPGAs are just an example of such chips) where it passes through ultra-deep pipelines with hundreds or thousands of specialised dataflow cores, enabling throughputs and energy efficiency levels impossible with control-flow oriented systems.

A CPU typically consists of a few microprocessors, that can perform complex tasks by control flow of arithmetic, control and input/output instructions. This is an extremely flexible model of operation, however inherently sequential. GPUs have hundreds of microprocessors and specialize in massively parallel scalar processing, performing the same operation set on large data sets. An alternative to control flow architectures is to create a customized data flow machine that exactly represents the algorithm and then simply stream the data through the resulting machine. In this model, data flows directly from one functional unit (dataflow kernel) to the next without the need for complex control mechanisms. DFEs are highly efficient for carrying out the large-scale, compute-intensive parts of an application and algorithms that have a high level of complex arithmetic and data reuse stand the most chance of benefiting from this technology.

The Multiscale Dataflow computing paradigm[39] developed by Maxeler Technologies as a combination of hardware and software components has been successfully demonstrated to provide substantial acceleration of real applications in variety of domains. These include computational finance - accelerating stochastic algorithms for Monte Carlo simulations for credit models[41] and interest rate derivative payoff evaluations based on the Heath-Jarrow-Morton model,[42] computational biology - accelerating short read alignment,[43,44] geoscience - accelerating finite difference algorithm for modelling wave propagation,[45–47] atmospheric modelling - solving Euler atmospheric equations,[48 49] and more.

Maxeler provides several heterogeneous dataflow platforms that combine DFEs with off-the-shelf CPUs, networking and storage. These include the Maxeler MPC-C series systems that combine dual Xeon processors in an industry-standard server chassis with up to 4 DFEs as PCI express (PCIe) cards. Maxeler MPC-X nodes are another pure dataflow appliance where 8 DFEs are integrated into a dense 1U industry-standard chassis connected to one or multiple conventional x86 CPU servers via an Infiniband network.

A Maxeler DFE uses a large FPGA as the main processing device. The programmable substrate of the FPGA is used to instantiate efficient and highly customised dataflow pipelines for application processing. The current generation MAX4 DFEs are based on a large Altera Stratix-V FPGA that includes 1,963 digital signal processing units (DSPs) each capable of multiplying two 27 bit integer numbers and 256K adaptive logic modules (ALMs; sometimes also referred to as lookup tables, LUTs). The dataflow cores are implemented as a combination of various on-chip resources such as DSPs, ALMs, and on-chip memory.

DFE cards provide the FPGA with access to large amounts of DRAM memory (currently between 48–96 GB) with about 60–65 GB/s of sustainable throughput. In addition, the FPGA itself also provides about 6 MB of embedded on-chip memories which are spread throughout the chip's fabric and can be used for low-latency buffering of local compute values which can be accessed with an aggregated bandwidth of several terabytes/second.

DFEs are highly efficient for carrying out the large-scale, compute-intensive parts of an application while conventional CPUs are more suitable for control-intensive tasks. Porting an application onto a dataflow system therefore requires the application code to be split into a host application and a dataflow part, often referred to as control plane and data plane. The CPU is responsible for setting up and controlling the computation on the DFE as well as for performing pre-computations or various control-oriented tasks.

Developing a practical dataflow application typically starts from identifying the performance-critical parts of the conventional CPU implementation. These parts are ported to the DFE by describing their dataflow models as a collection of compute kernels in a Java-based meta-language called MaxJ. In addition, a dedicated MaxJ object called the manager describes the data orchestration between compute kernels and external interfaces (such as on-board DRAM or PCIe bus). Compiling MaxJ code does not produce a Java application; instead, it leads to the generation of the binary object files that can be linked together with the CPU application.

3 DF-MP2 algorithm

We focus on reducing the computational time of the denisty-fitting second-order Moller- Plesset perturbation theory (DF-MP2),[23–25] as implemented in Molpro.[50] The MP2 method is one of the most widely used correlation treatments for electronic structure calculations, which involves evaluating two-electron repulsion integrals of the form:

$$(\mu\nu|\lambda\sigma) = \int\int \phi_\mu^*(r_1)\phi_\nu(r_1)r^{-1}\phi_\lambda(r_2)^*\phi_\sigma(r_2)dr_1dr_2 \quad (1)$$

where μ,ν, λ, and σ are orbital basis function (ϕ) indices. The calculation of the energy (E) is dependent on:

$$E^{(\mathrm{MP2})} = \sum_{ijab} \frac{(ia|jb)^2 + \frac{1}{2}[(ia|jb) - (ib|ja)]^2}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b} \quad (2)$$

$$(ia|jb) = \sum_{\mu\nu\lambda\sigma} C_{\mu i}C_{\nu a}C_{\lambda b}C_{\sigma b}(\mu\nu|\lambda\sigma) \quad (3)$$

In order to reduce the complexity of the computation, the four-center two electron integrals are approximated by two- and three- centred integrals. To achieve this, products of atomic orbitals are expressed in terms of linear combinations of auxiliary basis functions (χ):

$$\mu(r)\nu(r) = \rho_{\mu\nu}(r) \approx \widetilde{\rho_{\mu\nu}}(r) = \sum_A C_{\mu\nu,A}\chi_A \quad (4)$$

This leads to the following expression for the four-centre two electron integrals:

$$(\mu\nu|\lambda\sigma) \approx \sum_{AB}(\mu\nu|A)(A|B)^{-1}(B|\lambda\sigma) \quad (5)$$

where the metric $(A|B)$ arises as a consequence of robust fitting, i.e. minimising the Coulomb energy of the error in the fitted density.[51–53]

DF-MP2 is known to accurately predict structures and energies of molecules, and is particularly useful for systems that are weakly bound by dispersive forces, such as van der Waals clusters. DF-MP2 can reduce the computational costs as well as the required memory and disk sizes considerably while maintaining reliable accuracies for practical chemical applications. The development of standardised auxiliary basis sets,[24,54,55] as well as efficient algorithms for the energy and gradients,[24,56,57] has made the DF-MP2 method a widely used approach.

The DF-MP2 algorithm has been accelerated using GPUs, where the GPU algorithm has focused on utilising cuBLAS, an NVIDIA CUDA library of standard linear algebra subroutines which is a GPU accelerated version of the standard BLAS library, for optimising the matrix multiplication steps, which for low effort gives up to 7.8 times speed up for double precision compared with a single CPU.[31]

Within Molpro, the DF-MP2 algorithm proceeds as follows:

Step 1: Generate the matrix $(A|B)$ involving evaluation of two-centre integrals of the auxiliary basis functions. This step scales as $\mathcal{O}(M^2)$, where M is the number of auxiliary functions.

Step 2: Evaluation of $(\mu\nu|A)$ integrals. This scales as $\mathcal{O}(N^2M)$ where N is the total number of orbital basis functions.

Step 3: Transformation to generate $(i\nu|A)$. This scales as $\mathcal{O}(ONM)$ where O is the number of occupied molecular orbitals.

Step 4: Second transformation to generate $(ia|A)$. This scales as $\mathcal{O}(OVM)$, where V is the number of virtual orbitals.

Step 5: Calculate the inverse of $(A|B)$. This scales as $\mathcal{O}(M^3)$.

Step 6: Use the inverse of the matrix $(A|B)$ and the three centre integrals $(ia|A)$ to generate the intermediate:

$$d_A^{ia} = \sum_B (ia|B)(B|A)^{-1} \quad (6)$$

Step 7: Use the intermediate result from step 3 and the three center integrals from step 2 to generate the intermediate K:

$$K_{ab}^{ij} = \sum_A d_A^{ia}(A|jb) \tag{7}$$

By splitting into the orbital transformation into two steps, one avoids scaling of $\mathcal{O}(O^2V^2M^2)$, to have scaling of $\mathcal{O}(OVM^2)$ for the intermediate **d** and scaling of $\mathcal{O}(O^2V^2M)$for the intermediate **K**. Molpro also generates a matrix containing $(K_{ab}^{ij} - K_{ba}^{ij})$ again utilising standard linear algebra routines in order to optimise sequential accessing of memory in the sum. Evaluation of the DF-MP2 energy which can be expressed in terms of K as:

$$E^{(\mathrm{MP2})} = \sum_{ijab} \frac{{K_{ab}^{ij}}^2 + \frac{1}{2}(K_{ab}^{ij} - K_{ba}^{ij})^2}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b} \tag{8}$$

with scaling $\mathcal{O}(O^2V^2)$.

In general, the number of occupied orbitals is small comparatively to the number of virtual orbitals, $V \approx O \times 10$, and the number of auxiliary functions, $M \approx (O + V) \times 3$.

This application has been profiled previously[30] with the bottleneck shown to be the matrix multiplication steps to create **d** and **K** matrices. In table 1, we show an example of the timing data for the $C_{28}H_{58}$ hydrocarbon in cc-pVDZ basis. From this it can easily be seen that the final step takes 78% of the runtime, with the next rate limiting step being the smaller matrix multiplication step 6 taking just 7%.

It is this matrix multiply kernel that is transferred to the dataflow accelerator, in a similar fashion to the GPU implementation of Watson et al.[30] We will use the matrix multiplication kernel, the details of which are contained in the next section, to accelerate both steps 6 and step 7.

4 Matrix multiplication in dataflow

Matrix multiplication of two matrices $A(m \times r)$ and $B(r \times n)$ to produce a matrix $C(m \times n)$ can be expressed as:

$$C_{i,j} = \sum_k A_{i,k} * B_{k,j} \tag{9}$$

Table 1: CPU (Intel Xeon E5-2650 v3) timings from a serial execution of the DF-MP2 algorithm in Molpro for the linear $C_{28}H_{58}$ molecule in cc-pVDZ basis set.

Task	Time (s)
Step 1	0.18
Step 2	10.56
Step 3	5.08
Step 4	12.45
Step 5	0.74
Step 6	13.64
Step 7	154.57
Total	197.56

This can be viewed as $n \times m$ dot products, with each dot product requiring an entire row of one matrix and a column of the second to obtain a single final value. Conventional CPU implementations minimize data movement by adopting a blocking technique[58–60] over the innermost loop over k to reduce multiple references to the result and minimize transfers from memory to cache. Analogously, in dataflow, to take advantage of all the multipliers (DSPs) available on the FPGA one would similarly employ a blocking algorithm[61] to to exploit the large amount of on-chip fast access memory (FMem) on the DFE as well as maximising the data reuse. The scaling of the algorithm remains the same, N^3 operations for N^2 data for matrix multiplication, however the size of the block reduces the prefactor so that the scaling becomes $\mathcal{O}(OVM^2/B)$ where B is the block size for **d** and $\mathcal{O}(O^2V^2M/B)$ for the intermediate **K**. For the MAX4 card from Maxeler, the design is limited only by the number of DSPs available, which currently allows for 960 double precision floating point operations to be carried out simultaneously (480 multiplications and 480 additions). Additionally, the data are arranged such that rectangular blocks of the matrices are worked upon, where the sum dimension of the matrices is 480, but the independent direction is only 120 to reduce the on chip memory requirement. Additionally, before initiating the matrix multiplication run, the data is transferred to the off-chip large memory (LMem). As this memory can be accessed at a rate of 60-65 GB/s rather than the comparatively low bandwidth (3 GB/s) of PCIe, this prevents the design from being interconnect bandwidth limited.

In order to take advantage of the available symmetry, unique ij pairs whilst full number of ab pairs are needed in the MP2 sum, the intermediate $\mathbf{K}$ is calculated in lower (or upper) triangle blocks of the matrix. This means that there is still some inefficiency of calculating more of $\mathbf{K}$ than is needed, but this is balanced by reducing the overhead of calling the DFE multiple times. This also allows a more balanced computing system approach of calculating partial MP2 sums on blocks of $\mathbf{K}$ on the CPU, whilst the DFE is creating new blocks in parallel. It also reduces the storage requirement on the host computer for storing all of $\mathbf{K}$, which can easily exceed 50 GB for a large molecule such as valinomycin.

The dataflow application is currently called from host C code. In this host code, the data for integrals and energies are current read from file, the data is rearranged to be in the optimal order for running the DFE application, the bitstream for the DFE is loaded, and the matrix multiply DFE kernel is called twice: once to create the matrix $\mathbf{d}$ and a second time inside of a loop to create blocks of the matrix $\mathbf{K}$. Whilst blocks of $\mathbf{K}$ are generated on the DFE, the C host code accumulates the sum for the energy, and finally unloads the bitstream. Overheads associated with rearranging and loading/offloading the bitstream are insignificant in comparison to the time spent in the matrix multiply kernel. Presently this is a stand alone application, however it can easily be integrated into host packages by simply including the MaxCompiler libraries, the matrix multiply library into the makefile and making minor modifications to the host C code.

Given the fully deterministic nature of DFEs, one can predict the maximum throughput of the device for a design as simply the number of operations per cycle multiplied by the clock frequency of the device. In our case the maximum clock speed achievable was 176 MHz, and the number of operations per cycle is 960, giving a throughput of 169 GFlop/s. The clock frequency is a parameter that can be fine tuned whilst compiling the DFE configuration. In general it is fixed by the longest path in the dataflow graph. In this case, because we are utilising almost all of the available compute units, the maximum clock frequency is lower than what can be achieved with simpler designs.

The measured throughput was found to be slightly lower than the 169 GFlop/s estimated above. This was determined by calculating the total number of operations required to perform the matrix multiplication for a given problem, divided by the total time taken. In the case of valinomycin reported below, where the matrices are sufficiently large as to be most optimal for benchmarking the throughput, the measured quantity was 162.76 GFlop/s or 96.33% of what was expected. The differences are mostly due to overhead of the time taken for the first block of the matrix to be read into the DFE.

This number can easily be compared to the maximum performance expected for matrix multiplication executed on a CPU, which again depends on the number of operations performed in a cycle and the clock speed. For the Intel CPUs used here (Intel Xeon E5-2650 v3 2.3 GHz) the maximum performance expected is 36.8-48 GFlop/s when a single CPU is used for double precision. The DFE implementation was benchmarked against Molpro on CPUs which uses the standard BLAS MKL dgemm algorithm for the matrix multiplication.

The final resource usage is given in table 2, which shows that this design uses the vast majority of the area available. The optimization of the mapping of the design to the available DFE resources preferentially uses spare block memory before other logic for routing. The annotated resource files indicate only 70% of the available FMem is used in the design. Therefore the design is limited by the number of multipliers available.

Table 2: Final resource usage when the bitstream is compiled for a clock speed of 176 MHz, for a matrix block size of 480.

Resource	Used (Available)	%
FMem (M20K)	2567 (2567)	100.00
Multipliers (18x18)	3848 (3926)	98.01
DSP blocks	1924 (1963)	98.01
Logic Utilization	229937 (262400)	87.63

5 Benchmark comparisons with MOLPRO

The benchmarks were executed on Intel Xeon E5-2650 v3 2.30GHz CPUs, using Molpro 2015.1. The hardware setup consists of a MAX4 DFE board from Maxeler Technologies which contains Altera Stratix V FPGAs and Intel CPU Dual E5-2640, and using MaxCompiler version 2015.2.[62]

Fig. 1 shows execution times of the matrix multiplication step to generate **d** required for step 6 in the DF-MP2 algorithm in the computation of MP2 energies for various sized alkane chains for serial and parallel CPU implementations as well as the DFE implementation. For simple alkane chains, the number of basis functions scales linearly with the chain length, and hence the number of integrals also varies systematically. For short chain lengths (< 16 carbon atoms), the speed up factor is small due to the overheads of reading in the first block of the matrix, and the padding ratio to fit the data to a block size of 480. The average speed up for chain lengths greater that 16 Carbon atoms is 3.26 (± 0.36) compared to a serial CPU execution, and minor variations from this value are due to how well the problem fits into the block size.

Similarly, Fig. 2 shows the execution times of the matrix multiplication step required to generate **K** combined with the times to calculate the MP2 energy. The average speed up for alkane chains greater than 16 C atoms long is 3.17(±0.21) compared to a serial CPU execution.

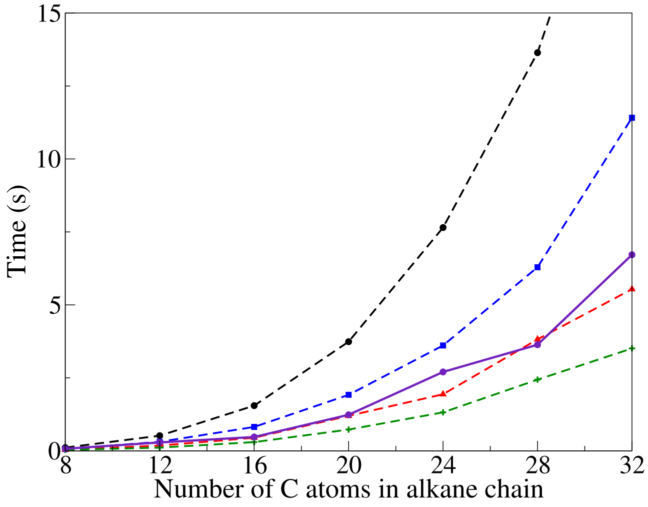


Figure 1: Average times taken to calculate the **d** using Molpro on CPUs in serial (black dash) parallel with 2 (blue dash), 4 (red dash), 8 (green dash) CPU cores compared to a single DFE (purple solid line), for alkane chains using cc-pVDZ basis set.

Tables 3 and 4 compare the DFE application to serial and parallel CPU applications for the much larger molecules of taxol and valinomycin using cc-pVDZ basis set. These show a slightly higher speed up compared to a single CPU core of 3.78 (± 0.19) for **d** and 3.58 (± 0.30) for **K** and E_{MP2}. This is because the matrix sizes are much larger, so the overheads due to calling the DFE, and reading in the first block of matrices are negligible in comparison to the over execution time of the matrix multiplication.

Our taxol and valinomycin calculations can be directly compared with GPU results from Olivares-Amaya 2016,[63] as they are in the same basis set. In this paper they report GPU timings faster than those performed on Intel Xeon(R) CPU E5-2650 cpus by a factor of 6.0 for taxol and 7.6 for valinomycin, benchmarking CUBLAS dgemm with blas dgemm. The dataflow calculations presented here are roughly par with these calculations. The CPUs benchmarked in our work are a factor of 2.3 to 3 times faster than those used in the GPU benchmark, so we estimate that the GPU results would be 2.6x for taxol and 3.3 times faster for valinomycin compared to avx2 CPUs. How-

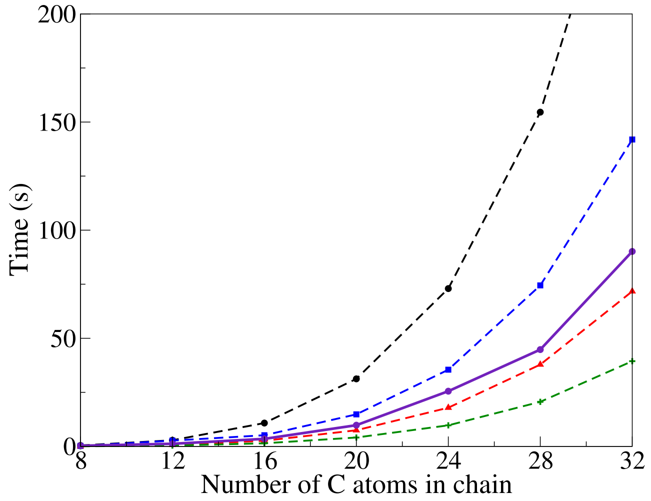


Figure 2: Average times taken to calculate the **K** and $E^{(MP2)}$ using Molpro on CPUs in serial (black dash), parallel with 2 (blue dash), 4 (red dash), 8 (green dash) CPU cores compared to a single DFE (purple solid line), for alkane chains using cc-pVDZ basis set.

ever, more recent GPUs could be as much as 3 times faster than the results of that paper, and those GPU timings should be compared to the next generation of dataflow engines.

Table 3: DFE timings for the calculation of **d** for taxol and valinomycin with comparison to CPU timings using Molpro in serial and parallel executions in cc-pVDZ basis set.

Times (s)	Taxol	Valinomycin
DFE	33.12	112.59
1 CPU core	120.69	439.77
2 CPU cores	53.24	181.90
4 CPU cores	28.10	136.99
8 CPU cores	18.78	152.18

Table 4: Table of DFE timings for the calculation of **K** for taxol and valinomycin with comparison to CPU timings using Molpro in serial and parallel executions in cc-pVDZ basis set.

Times (s)	Taxol	Valinomycin
DFE	638.95	2917.79
1 CPU core	2154.89	11074.15
2 CPU cores	1064.81	5399.13
4 CPU cores	540.02	2738.12
8 CPU cores	295.63	1485.56

6 Discussion

The results presented above, indicate that only modest acceleration can be achieved by simply pushing linear algebra routines onto the DFE. Linear algebra is efficient on CPUs and standard library routines are optimised to operate at very close to the maximum throughput. This is because the data ordering is predictable and the compute is not sufficiently complex, making this the real worst case senario for DFEs. Given the principle that the best dataflow designs are highly customised for the specific application, prediction of the specific performance for other algorithms used in standard quantum chemistry codes becomes difficult. However, in general for problems with significantly more complex compute and data re-use DFEs will perform significantly better than demonstrated here.

Additional speed up can be achieved for any non-linear function $f(x)$ in a hardware efficient way, for example by constructing a piece-wise polynomial fit for the function of interest rather than explicit evaluation.[64] This can be done by optimising the order of the polynomial expansion and the minimum precision for a given tolerance in the numerical error relative to the CPU equivalent expression. This methodology can typically reduce by orders of magnitude the required compute resources (the number of DSPs and LUTs) for a suitably non-trival non-linear function. However there is an increase in the memory requirements as the fitting coefficients should be stored in FMem on chip for fast access for function evaluation in hardware. This works well on DFE architecture because of the comparative size of BRAM on the chip (6MB in the case of MAX4) compared to the L1 cache on a CPU (which is on the order of KBs), and the fine grained access to this memory. For example this methodology has been used to accelerate computational finance problems such as calculating the value of risk in a portfolio by Curran's algorithm,[1] and has shown 278 times speed up using MAX4 compared to a single Intel Xeon CPU E5-2697 v2 core.

Another opportunity for processing speed enhancement is through a technique called multipumping,[65] which runs the DSP blocks N times faster than the surrounding fine-grained logic so that only (1/N) DSP blocks would be needed. Assuming that all the DSP blocks can be used effectively, the throughput of a multi-pumped design can be up to N times faster than the original version. We are exploring and evaluating the use of multi- pumping for our quantum chemistry calculations.

There is further potential for increasing the speed-up by varying the precision to which the integrals are stored. Previous work has shown that single precision (32 bit) does not have sufficient accuracy for MP2 energies. Since GPUs initially were only single precision, there has been much effort placed in adopting some form of mixed precision representation,[31,66,67] mostly by categorising the two electron integrals on magnitude, and keeping the large integrals at double precision, whilst reducing to single precision the smaller valued integrals. The GPU is then used to perform computation on the low precision integrals, whilst the CPU is used to perform calculations on the sparse double precision integrals. In this way, a greater speed up could be achieved for smaller reduction in the overall precision of the calculation. For example, Olivares-Amaya et al.[31] have seen up to 10.1 times speed up for valinomycin in their mixed precision approach with only 1.16 kcal/-mol error compared to double precision, which is much reduced from the 9.99 kcal/mol error in single precision.

The advantage of the dataflow engine architectures in comparison to GPUs is that they support variable precision for both floating point and fixed point. The matrix multiplication design presented here is compute bound, limited by the number of DSPs. Therefore the choice of precision if the aim is to improve acceleration is limited by the details of the DSPs. Given the deterministic nature of the DFE implementation it is possible to project the speed up. For an intermediate precision between single and double, for example for a 32 bit mantissa and 11 bit exponent which takes only half the compute resources, means doubling the speed up achieved here on a MAX4, as the block size could be doubled to 960 giving a throughput of 338 Gflop/s assuming a similar clock frequency.

In addition to varying the precision, the next generation of dataflow engines (MAX5) are significantly more capable, with both more multipliers and more on chip BRAM. For the matrix multiplication design this would allow a block size of 960 compared to the current 480, with the application is still projected to be DSP bound. There is also potential to increase the clock speed further, for some designs to as much as 500 MHz, but more likely for this resource intensive design to 350 MHz. This gives an estimated throughput of 672 Gflop/s. If one also changes the precision to the (32 bit mantissa, 11 bit exponent) precision discussed above, it becomes possible to have a block size of 1,680, with an estimated throughput of 1.176 Tflop/s. Overall this would make the estimated speed up compared to the CPUs benchmarked here to be 24 times faster on the DFE.

The expected performance relative to a typical multicore CPU, possibly augmented with GPUs, is difficult to estimate, but this discussion demonstrates that the DFE has the potential to be competitive

Acknowledgement This work was supported in part by Maxeler University Programme, Altera, UK EPSRC Project EP/I012036/1, the European Union Horizon 2020 Research and Innovation Programme under grant agreement number 671653, and the HiPEAC NoE.

References

(1) Nestorov, A. M.; Reggiani, E.; Palikareva, H.; Burovskiy, P.; Becker, T.; Santambrogio, M. D. A scalable dataflow implementation of Curran's approximation algorithm. 31st Annual IEEE Interna-

tional Parallel and Distributed Processing Symposium (IEEE IPDPS 2017). 2017.

(2) Cremer, D. Møller-Plesset perturbation theory: from small molecule methods to methods for thousands of atoms. *WIREs Comput. Mol. Sci.* **2011**, *1*, 509–530.

(3) Pulay, P. Localizability of dynamic electron correlation. *Chem. Phys. Lett.* **1983**, *100*, 151–154.

(4) Pulay, P.; Saebø, S. Orbital-invariant formulation and second-order gradient evaluation in Møller-Plesset perturbation theory. *Theor. Chim. Acta.* **1986**, *69*, 357–368.

(5) Schütz, M.; Hetzer, G.; Werner, H. J. Low-order scaling local electron correlation methods. I. Linear scaling local MP2. *J. Chem. Phys.* **1999**, *111*, 5691–5705.

(6) Hetzer, G.; Schütz, M.; Stoll, H.; Werner, H.-J. Low-order scaling local correlation methods II: Splitting the Coulomb operator in linear scaling local second-order Møller-Plesset perturbation theory. *J. Chem. Phys.* **2000**, *113*, 9443–9455.

(7) Maslen, P. E.; Head-Gordon, M. Non-iterative local second order Møller-Plesset theory. *Chem. Phys. Lett.* **1998**, *283*, 102 – 108.

(8) Lee, M. S.; Maslen, P. E.; Head-Gordon, M. Closely approximating second-order Møller-Plesset perturbation theory with a local triatomics in molecules model. *J. Chem. Phys.* **2000**, *112*, 3592.

(9) Russ, N. J.; Crawford, T. D. Potential energy surface discontinuities in local correlation methods. *J. Chem. Phys.* **2004**, *121*, 691–696.

(10) Neuhauser, D.; Rabani, R.; Baer, R. Expeditious stochastic approach for MP2 energies in large electronic systems. *J. Chem. Theory Comput.* **2013**, *9*, 24–27.

(11) Neuhauser, D.; Baer, R.; Zgid, D. Stochastic self-consistent Green's function second-order perturbation theory (sGF2). *ArXiv e-prints* **2016**,

(12) Takeshita, T.-Y.; de Jong, W.-A.; Neuhauser, D.; Baer, R.; Rabani, E. A Stochastic formulation of the resolution of the identity: application to second Moller-Plesset Perturbation theory. *ArXiv e-prints* **2017**,

(13) Almlöf, J. Elimination of energy denominators in Møller-Plesset perturbation theory by a Laplace transform approach. *Chem. Phys. Lett.* **1991**, *181*, 319.

(14) Häser, M.; Almlöf, J. Laplace transform techniques in Møller-Plesset perturbation theory. *J. Chem. Phys.* **1992**, *96*, 489–494.

(15) Ayala, P. Y.; Scuseria, G. E. Linear scaling second-order Møller-Plesset theory in the atomic orbital basis for large molecular systems. *J. Chem. Phys.* **1999**, *110*, 3660–3671.

(16) (a) Lambrecht, D. S.; Doser, B.; Ochsenfeld, C. Rigorous integral screening for electron correlation methods. *J. Chem. Phys.* **2005**, *123*, 184102; (b) Doser, B.; Lambrecht, D. S.; Ochsenfeld, C. Tighter multipole-based integral estimates and parallel implementation of linear-scaling AO–MP2 theory. *Phys. Chem. Chem. Phys.* **2008**, *10*, 3335–3344.

(17) Doser, B.; Lambrecht, D. S.; Kussmann, J.; Ochsenfeld, C. Linear-scaling atomic orbital-based second-order Møller-Plesset perturbation theory by rigorous integral screening criteria. *J. Chem. Phys* **2009**, *130*, 064107.

(18) Surján, P. R. The MP2 energy as a functional of the Hartree-Fock density matrix. *Chem. Phys. Lett.* **2005**, *406*, 318–320.

(19) Kobayashi, M.; Nakai, H. Implementation of Surján's density matrix formulae for calculating second-order Møller-

Plesset energy. *Chem. Phys. Lett.* **2006**, *420*, 250–255.

(20) Koch, H.; Sánchez de Merás, A.; Pederson, T. B. Reduced scaling in electronic structure calculations using Cholesky decompositions. *J. Chem. Phys.* **2003**, *118*, 9481–9484.

(21) Martinez, T. J.; Carter, E. A. Pseudospectral Møller-Plesset perturbation theory through third order. *J. Chem. Phys.* **1994**, *100*, 3631–3638.

(22) (a) Jung, Y.; Shao, Y.; Head-Gordon, M. Fast evaluation of scaled opposite spin second-order Møller-Plesset correlation energies using auxiliary basis expansions and exploiting sparsity. *J. Comput. Chem.* **2007**, *28*, 1953–1964; (b) Lochan, R. C.; Shao, Y.; Head-Gordon, M. Quartic-Scaling Analytical Energy Gradient of Scaled Opposite-Spin Second-Order Møller-Plesset Perturbation Theory. *J. Chem. Theory Comput.* **2007**, *3*, 988–1003.

(23) Feyereisen, M.; Fizgerald, G.; Komornicki, A. Use of approximate integrals in ab initio theory. An application in MP2 energy calculations. *Chem. Phys. Lett.* **1993**, *208*, 359–363.

(24) Weigend, F.; Häser, M.; Pazelt, H.; Ahlrichs, R. RI-MP2: optimized auxiliary basis sets and demonstration of efficiency. *Chem. Phys. Lett.* **1998**, *294*, 143–152.

(25) Werner, H.-J.; Manby, F. R.; Knowles, P. J. Fast linear scaling second-order Møller-Plesset perturbation theory (MP2) using local and density fitting approximations. *J. Chem. Phys.* **2003**, *118*, 8149–8160.

(26) Stone, J. E.; Hardy, D. J.; Ufimtsev, I. S.; Schulten, K. GPU-accelerated molecular modeling coming of age. *J. Mol. Graph.* **2010**, *29*, 116–125.

(27) Maia, J. D. C.; Urquiza Carvalho, G. A.; Mangueira, C. P.; Santana, S. R.; Cabral, L. A. F.; Rocha, G. B. GPU Linear Algebra Libraries and GPGPU Programming for Accelerating MOPAC Semiempirical Quantum Chemistry Calculations. *J. Chem. Theory Comp.* **2012**, *8*, 3072–3081.

(28) (a) Ufimtsev, I. S.; Martinez, T. J. Quantum Chemistry on Graphical Processing Units. 1. Strategies for Two-Electron Integral Evaluation. *J. Chem. Theory Comput.* **2008**, *4*, 222–231; (b) Ufimtsev, I. S.; Martinez, T. J. Quantum Chemistry on Graphical Processing Units. 2. Direct Self-Consistent-Field Implementation. *J. Chem. Theory Comput.* **2009**, *5*, 1004–1015; (c) Ufimtsev, I. S.; Martinez, T. J. Quantum Chemistry on Graphical Processing Units. 3. Analytical Energy Gradients, Geometry Optimization, and First Principles Molecular Dynamics. *J. Chem. Theory Comput.* **2009**, *5*, 2619–2628.

(29) Vogt, L.; Olivares-Amaya, R.; Kermes, S.; Shao, Y.; Amardor-Bedolla, C.; Aspuru-Guzik, A. Accelerating Resolution-of-the-Identity Second-Order Møller-Plesset Quantum Chemistry Calculations with Graphical Processing Units. *J. Phys. Chem. A* **2008**, *112*, 2049–2057.

(30) Watson, M.; Olivares-Amaya, R.; Edgar, R. G.; Aspuru-Guzik, A. Accelerating Correlated Quantum Chemistry Calculations Using Graphical Processing Units. *Comput. Sci. Eng.* **2010**, *12*, 40–51.

(31) Olivares-Amaya, R.; Watson, M. A.; Edgar, R. G.; Vogt, L.; Shao, Y.; Aspuru-Guzik, A. Accelerating Correlated Quantum Chemistry Calculations Using Graphical Processing Units and a Mixed Precision Matrix Multiplication Library. *J. Chem. Theory Comput.* **2010**, *6*, 135.

(32) Katouda, M.; Nakajima, T. MPI/OpenMP Hybrid Parallel Algorithm of Resolution of Identity Second-Order Møller-Plesset Perturbation Calculation

for Massively Parallel Mulicore Supercomputers. *J. Chem. Theory Comput.* **2013**, *9*, 5373–5380.

(33) Goldey, M.; DiStasio Jr, R. A.; Shao, Y.; Head-Gordon, M. Shared memory multiprocessing implementation of resolution-of-the-identity second-order Møller-Plesset perturbation theory with attenuated and unattenuated results for intermolecular interactions between large molecules. *Mol. Phys.* **2014**, *112*, 836–843.

(34) Bernholdt, D. E.; Harrison, R. J. Large-scale correlated electronic structure calculations: the RI-MP2 method on parallel computers. *Chem. Phys. Lett.* **1996**, *250*, 477–484.

(35) Bernholdt, D. E. Scalability of correlated electronic structure calculations on parallel computers: A case study of the RI-MP2 method. *Parallel Comput.* **2000**, *26*, 945–963.

(36) Katouda, M.; Naruse, A.; Hirano, Y.; Nakajima, T. Massively parallel algorithm and implementation of RI-MP2 energy calculation for peta-scale many-core supercomputers. *J. Comp. Chem.* **2016**, *37*, 2623–2633.

(37) Doran, A. E.; Hirata, S. Monte Carlo MP2 on Many Graphical Processing Units. *J. Chem. Theory Comput.* **2016**, *12*, 4821–4832.

(38) Maxeler Technologies DFE, `https://www.maxeler.com/technology/dataflow-computing/`, [Accessed on 26-February-2017].

(39) Dennis, J. B. Data Flow Supercomputers. *Computer* **1980**, *13*, 48–56.

(40) Kung, H.-T. Why systolic architectures? *IEEE computer* **1982**, *15*, 37–46.

(41) Weston, S.; Spooner, J.; Racanière, S.; Mencer, O. Rapid computation of value and risk for derivatives portfolios. *Concurrency and Computation: Practice and Experience* **2012**, *24*, 880–894.

(42) Jin, Q.; Dong, D.; Tse, A. H. T.; Chow, G. C. T.; Thomas, D. B.; Luk, W.; Weston, S. Multi-level Customisation Framework for Curve Based Monte Carlo Financial Simulations. Reconfigurable Computing: Architectures, Tools and Applications - 8th International Symposium, ARC. 2012; pp 187–201.

(43) Arram, J.; Kaplan, T.; Luk, W.; Jiang, P. Leveraging FPGAs for Accelerating Short Read Alignment. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **2016**, *PP*, 1–1.

(44) Arram, J.; Luk, W.; Jiang, P. Ramethy: Reconfigurable Acceleration of Bisulfite Sequence Alignment. Proc. Int. Symp. on Field-Programmable Gate Arrays (FPGA). 2015; pp 250–259.

(45) Pell, O.; Bower, J.; Dimond, R.; Mencer, O.; Flynn, M. J. Finite-Difference Wave Propagation Modeling on Special-Purpose Dataflow Machines. *IEEE Transactions on Parallel Distributed Systems* **2013**, *24*, 906–915.

(46) Lindtjorn, O.; Clapp, R.; Pell, O.; Fu, H.; Flynn, M.; Mencer, O. Beyond Traditional Microprocessors for Geoscience High-Performance Computing Applications. *IEEE Micro* **2011**, *31*, 41–49.

(47) Oriato, D.; Pell, O.; Andreoletti, C.; Bienati, N. FD modeling beyond 70Hz with FPGA acceleration. SEG 2010 HPC Workshop. 2010.

(48) Gan, L.; Fu, H.; Yang, C.; Luk, W.; Xue, W.; Mencer, O.; Huang, X.; Yang, G. A highly-efficient and green data flow engine for solving Euler atmospheric equations. Field Programmable Logic and Applications, (FPL 2014). 2014; pp 1–6.

(49) Gan, L.; Fu, H.; Mencer, O.; Luk, W.; Yang, G. Chapter Four - Data Flow Computing in Geoscience Applications. *Adv. Comput.* **2017**, *104*, 125–158.

(50) (a) Werner, H.-J.; Knowles, P. J.; Knizia, G.; Manby, F. R.; Schütz, M.; et. al., MOLPRO, version 2015.1, a package of ab initio programs. 2015; see http://www.molpro.net; (b) Werner, H.-J.; Knowles, P. J.; Knizia, G.; Manby, F. R.; Schütz, M. Molpro: a general purpose quantum chemistry program package. *WIREs Comput Mol Sci* **2012**, *2*, 242–253.

(51) Whitten, J. L. Coulombic potential energy integrals and approximations. *J. Chem. Phys.* **1973**, *58*, 4496–4501.

(52) Dunlap, B. I. Robust and variational fitting. *Phys. Chem. Chem. Phys.* **2000**, *2*, 2113–2116.

(53) Dunlap, B. I.; Connolly, J. W. D.; Sabin, J. R. On first-row diatomic molecules and local density models. *J. Chem. Phys.* **1979**, *71*, 4993–4999.

(54) Weigend, F.; Kohn, A.; Hättig, C. Efficient use of the correlation consistent basis sets in resolution of the identity MP2 calculations. *J. Chem. Phys.* **2002**, *116*, 3175–3183.

(55) Hättig, C. Optimization of auxiliary basis sets for RI-MP2 and RI-CC2 calculations: Core–valence and quintuple-ζ basis sets for H to Ar and QZVPP basis sets for Li to Kr. *Phys. Chem. Chem. Phys.* **2005**, *7*, 59–66.

(56) Rhee, Y. M.; DiStasio Jr., R. A.; Lochan, R. C.; Head-Gordon, M. Analytical gradient of restricted second-order Møller-Plesset correlation energy with the resolution of the identity approximation, applied to the TCNE dimer anion complex. *Chem. Phys. Lett.* **2006**, *426*, 197–203.

(57) DiStasio Jr., R. A.; Steele, R. P.; Rhee, Y. M.; Shao, Y.; Head-Gordon, M. An improved algorithm for analytical gradient evaluation in resolution-of-the-identity second-order Møller-Plesset perturbation theory: Application to alanine tetrapeptide conformational analysis. *J. Comput. Chem.* **2007**, *28*, 839.

(58) Dongarra, J. J.; Sorensen, D. C. Linear algebra on high performance computers. *Appl. Math. Comput.* **1986**, *20*, 57–58.

(59) Gallivan, K. A.; Plemmons, R. J.; Sameh, A. H. Parallel Algorithms for Dense Linear Algebra Computations. *SIAM Rev.* **1990**, *32*, 54–135.

(60) Schreiber, R. *Numerical Algorithms for Modern Parallel Computer Architectures*; Springer, 1988; pp 197–207.

(61) Dou, Y.; Vassiliadis, S.; Kuzmanov, G. K.; Gaydadjiev, G. N. 64-bit floating-point FPGA matrix multiplication. Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays. 2005; pp 86–95.

(62) Multiscale Dataflow Programming. Maxeler Technologies, 2015.

(63) Olivares-Amaya, R.; Jinich, A.; Watson, M. A.; Aspuru-Guzik, A. In *GPU Acceleration of Second-Order Møller-Plesset Perturbation Theory with Resolution of Identity, in Electronic Structure Calculations on Graphics Processing Units: From Quantum Chemistry to Condensed Matter Physics*; Walker, R. C., Götz, A. W., Eds.; 2016.

(64) Becker, T.; Burovskiy, P.; Nestrorov, A. M.; Palikareva, H.; Reggiani, E.; Gaydadjiev, G. From exaflop to exaflow. Proceedings of Design and Automation and Test in Europe (DATE). 2017; to appear.

(65) Canis, A.; Anderson, J. H.; Brown, S. D. Multi-pumping for resource reduction in FPGA high-level synthesis. 2013 Design,

Automation Test in Europe Conference Exhibition (DATE). 2013; pp 194–197.

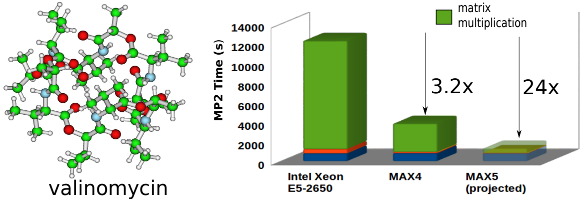
(66) Luehr, N.; Ufimtsev, I. S.; Martinez, T. J. Dynamic Precision for Electron Repulsion Integral Evaluation on Graphical Processing Units (GPUs). *J. Chem. Theory Comput.* **2011**, *7*, 949.

(67) Asadchev, A.; Gordon, M. S. Mixed-precision evaluation of two-electron integrals by Rys quadrature. *Comp. Phys. Comm.* **2012**, *183*, 1563–1567.

Graphical TOC Entry

Quantum chemistry in dataflow: Density-Fitting MP2

Bridgette Cooper, Stephen Girdlestone, Pavel Burovskiy, Georgi Gaydadjiev, Vitali Averbukh, Peter J. Knowles, Wayne Luk.


valinomycin
MP2 Time (s)
🟩 matrix multiplication
3.2x
24x
Intel Xeon E5-2650
MAX4
MAX5 (projected)