# CPU and FPGA Performance Comparison of a Conjugate Gradient Solver Extracted from a Molecular Dynamics Code

Charles Prouveur[1], Matthieu Haefele [2], Nils Voss[3]

1: Maison de la Simulation, CEA, CNRS, France ; 2: Universite de Pau et des Pays de l'Adour, E2S UPPA, CNRS,LMAP ; 3: Maxeler Technologies, Imperial College London

## Abstract

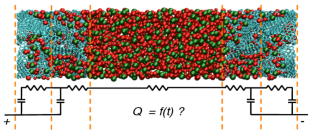
FPGA devices used in the HPC context promise an increased energy efficiency, enhancing the computing systems Flop/W rate. This work compares an FPGA, a GPU implementation and a CPU implementation of a conjugate gradient solver in terms of both time to solution and energy to solution metrics.

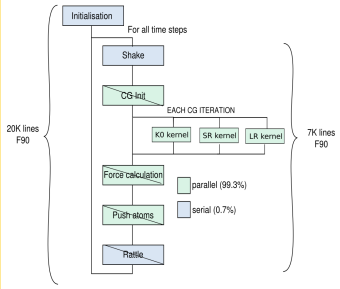
The starting point is MetalWalls, a molecular dynamics code developed at Sorbonne University in Pr. M. Salanne's team, capable of computing accurately the charge and discharge cycles of supercapacitors (energy storing devices) [1]. In the context of the H2020 EXA2PRO project, a miniapp has been derived from the F90 pure MPI production code, extracting the core of the electrostatic computation.

The FPGA version has been implemented with the Data Flow Engine (DFE) software toolchain developed by Maxeler. Additionally, since FPGAs can perform arithmetic operations with any number of bits instead of the standardized IEEE 32 and 64 bits floating point format, the miniapp could be further accelerated using optimised custom number formats. Thanks to an accuracy analysis based on comparisons with quadruple precision runs, this acceleration could be achieved without decreasing the computed solution accuracy.
Finally, the original CPU, the original GPU and the developed FPGA implementations could be compared on Juelich Computing Centre's computing systems and the Piz daint system from CSCS.

## Context

METALWALLS is a molecular dynamic production code [2] dedicated to electrochemical systems simulation. In this contexte electrostatic forces play an important role and their computations are based on an ewald summation. The heart of the code is the computation of a matrix-free conjugate gradient that finds the charge distribution on electrodes such their respective potentials remain constant.

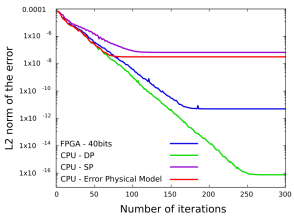



Most of the execution time is spent in 3 computing kernels of different complexity : K0 and SR are in O(N²), and LR is in O(NxM) with N the number of atoms and M the number of Modes. At each iteration of the conjugate gradient, every kernel contribution are summed to compute the electrostatic potential. This process is repeated until the target residue is reached. This application is mostly CPU bound with a 60MB memory footprint. A miniapp was extracted from this production code, keeping only the computing kernels and the conjugate gradient algorithm. The CPU implementation uses MPI while the GPU implementation uses OpenACC.

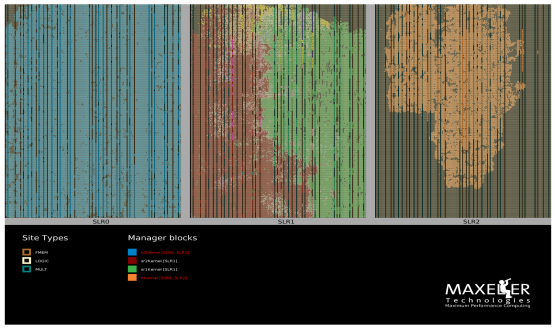
## Numerical accuracy analysis

The physical model used in Metalwalls uses a parameter to determine which computations are done in the Fourier space or in the real space. Theoretically it should not impact the result but in practice it does. In order to estimate the numerical accuracy required by the physical model, a quadruple precision run was done at the reference parameter. The solutions computed in Double Precision (DP) and single precision (SP) were then compared to it. An average of the error at different parameter values was also done to give an estimate of the physical model numerical accuracy.
As expected SP is not accurate enough whereas the accuracy from DP is more than what is needed by the model which is around 1e-7. With the FPGA implementation, the number of bits used to represent floating numbers could be trimmed from 64 (11 bits exponent ,53 mantissa) down to 40 (8,32) while leaving a large margin of error even if it plateaus around 1e-11. This raises an interesting problem as to what bit size is optimal and how much the target residue, the stop criterion of the CG, should be changed since with the FPGA, one can optimize this parameter while on a CPU one would be limited to SP and DP.

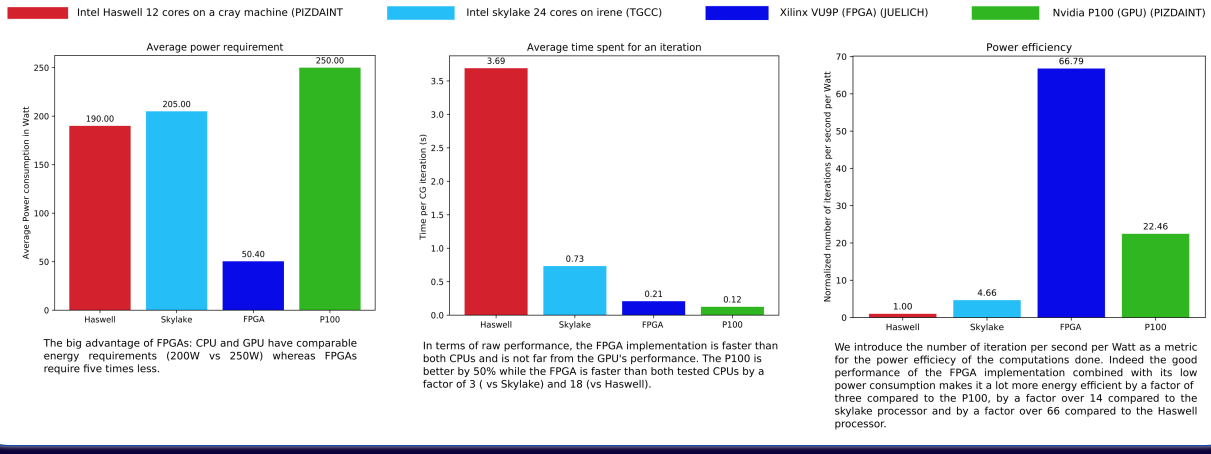



## FPGA Implementation

Each kernel takes one SLR (super logic region) in order for the design to fit on one chip
This implies a lower denominator approach where the most ressource hungry of the kernels will limit the other kernels ressource usage. Indeed since all kernels are running at the same time on the chip, they need to use a similar number of clock ticks to be synchronized.
On chip memory is heavily used in order to avoir using the external memory as it makes it harder to meet timings. A balance between the degree of parallelism and the frequency used is needed as a higher frequency makes meeting timing at compile time harder while increasing the parallelism increases the amount of chip resources used which also makes it harder for timings to be met.
The balance was found in our case at 300 MHz.




## Results




The big advantage of FPGAs: CPU and GPU have comparable energy requirements (200W vs 250W) whereas FPGAs require five times less.

In terms of raw performance, the FPGA implementation is faster than both CPUs and is not far from the GPU's performance. The P100 is better by 50% while the FPGA is faster than both tested CPUs by a factor of 3 ( vs Skylake) and 18 (vs Haswell).

We introduce the number of iteration per second per Watt as a metric for the power efficiecy of the computations done. Indeed the good performance of the FPGA implementation combined with its low power consumption makes it a lot more energy efficient by a factor of three compared to the P100, by a factor over 14 compared to the skylake processor and by a factor over 66 compared to the Haswell processor.

## Conclusion

This work compares the original CPU and GPU implementations of a matrix free conjugate gradient that minimises the total energy of a realistic electrochemical system with FPGA implementations.
The FPGA implementations use the Maxeler software environment and make extensive use of the on-chip memory such that the code is not limited by the memory bus between the FPGA and its attached DDR memory.
A numerical accuracy study has enabled the usage of an intermediate floating point number representation using 40 bits, lying between the standard single and double IEEE754 representations, without damaging the result of the algorithm.
The comparisons have been performed with a production test case (42508 atoms). Time and energy measurements have been performed for all CPU, GPU and FPGA runs. PizDaint, a Cray machine at CSCS in Switzerland, has been used for CPU and GPU runs. Jumax, a Maxeler machine at JSC in Germany, has been used for FGPA runs. These tests reveal that the FPGA is faster than both CPUs and since it also requires a lot less electrical power to deliver the same results, these two features leads to a better efficiency, here the number of iterations per second and per Watt metric. This better efficiency holds true compared to a GPU with the same transistor size technology. This factor is even more impressive taking into account it was compared against highly optimised production code. FPGA technology is, in our opinion, a clear candidate to be part of exascale systems.

## Future works and perspectives

- Publish the results from the multiple FPGAs implementations.
- New test case: Acceleration of a hydrodynamic code
- Hybrid (CPU + FPGA) implementation with STARPU
- Comparison to other platforms of development (oneAPI, VITIS)

## References

[1] Marin-Laflèche et al. 2020. MetalWalls: A classical molecular dynamics software dedicated to the simulation of electrochemical systems. Journal of Open Source Software 53, 5 (2020), 178–183. https://doi.org/10.21105/joss.02373.

[2] Trinidad Méndez-Morales, Nidhal Ganfoud, Zhujie Li, Matthieu Haefele, Benjamin Rotenberg, and Mathieu Salanne. 2019. Performance of microporous carbon electrodes for supercapacitors: Comparing graphene with disordered materials. Energy Storage Materials 17 (2019), 88 – 92. https://doi.org/10.1016/j.ensm.2018.11.022.

**Acknowledgements**: This work has been funded by the EU H2020 project EXA2PRO (801015). The authors gratefully acknowledge the access to the PRACE-3IP PCP pilot system at Jülich Supercomputing Centre, which has been partially been funded by the European Union FP7 programme under grant agreement no. RI-312763 (PRACE-3IP)
The authors gratefully acknowledge the access to the Piz Daint system through the preparatory access request (2010PA5523) approved by CSCS, Switzerland