# CFD Acceleration with FPGA

1st Krzysztof Rojek
*byteLAKE*
Wroclaw, Poland
*Czestochowa University of Technology*
Czestochowa, Poland
krojek@bytelake.com

2nd Marcin Rojek
*byteLAKE*
Wroclaw, Poland
mrojek@bytelake.com

3rd Viraj R. Paropkari
*Xilinx*
San Jose, CA, USA
virajp@xilinx.com

***Abstract*—The goal of this work is to adapt 4 CFD kernels to the Xilinx ALVEO U250 FPGA, including first-order step of the non-linear iterative upwind advection MPDATA schemes (non-oscillatory forward in time), the divergence part of the matrix-free linear operator formulation in the iterative Krylov scheme, tridiagonal Thomas algorithm for vertical matrix inversion inside preconditioner for the iterative solver, and computation of the psuedovelocity for the second pass of upwind algorithm in MPDATA. All the kernels use 3-dimensional compute domain consisted from 7 to 11 arrays. Since all the kernels belong to the group of memory bound algorithms, our main challenge is to provide the highest utilization of global memory bandwidth. In this work we present FPGA as a powerful device to accelerate HPC codes. The proposed adaptation is compared with a CPU implementation that was strongly optimized in order to provide realistic and objective benchmarks. Our adaptation allows us to reduce the execution time up to 40% and the energy consumption up to 80% compared to the CPU processors.**

***Index Terms*—CFD, FPGA, Energy efficient, code adaptation, performance, benchmark**

## I. Introduction

Computational Fluid Dynamics (CFD) tools combine numerical analysis and algorithms to solve fluid flows problems. A range of industries such as automotive, chemical, aerospace, biomedical, power and energy, and construction rely on fast CFD analysis turnaround time. It is a key part of their design workflow to understand and design how liquids and gases flow and interact with surfaces. Typical applications include weather simulations, aerodynamic characteristics modelling and optimization, and petroleum mass flow rate assessment.

The Field-Programmable Gate Array (FPGA) architecture has a proven achievements in many research and business areas including cryptography, network routing algorithms, machine and deep learning or video transmission. The goal of this paper is to design and develop a proof of concept that allows users to evaluate the performance acceleration and energy efficiency of the FPGA card for CFD codes [1].

The FPGA devices are mainly design to fully exploit a high performance of a single precision arithmetic [2]. Many research show that a single precision gives more than expected accuracy. A great example, where we can see that "less is more" is the Swiss National Supercomputing Centre (CSCS), where the geophysical CFD codes for a weather and climate simulations were ported to the GPU architecture using a single precision arithmetic. In it possible in the weather simulations, since the computations are error resistant. The other example is deep learning, where even a half-precision gives sufficient accuracy to solve many problems as classification, regression, image detection or recognition.

byteLAKE wish to thank Viraj R. Paropkari and Xilinx team for access to the latest Alveo accelerator cards. This work is supported in part by the National Science Centre, Poland under grant no. UMO-2017/26/D/ST6/00687.

## II. Solution overview

A lot of benchmarks show an idealized academic cases but in real usage they need to be strongly customized. Many of kernel algorithms are presented as a highly efficient adopted to a given compute architecture but in a real scenario there appear a special conditions that significantly decrease the final performance. The advantage of our approach is to support a real-life scientific scenario that is not an academic solution adjusted to the compute architecture, but it is a ready to use software compatible with a geophysical model like Eulerian/semi-Lagrangian fluid solver (EULAG).

### A. Key benefits

Highly optimized CFD kernels for Xilinx® Alveo™ Datacenter accelerator cards, compatible with geophysical models like EULAG. Fully configurable code. Simpler, lower cost server infrastructure.

### B. Scope of Work

The ever-increasing demand for accuracy and capabilities of the CFD workloads produces an exponential growth of the required computational resources.

Moving to heterogeneous HPC (High Performance Computing) configurations powered by Xilinx Alveo helps significantly improve performance within radically reduced energy budgets.

byteLAKE has created a set of highly optimized CFD kernels that leverage the speed and energy efficiency of Xilinx Alveo FPGA accelerator cards to create a high-performance platform for complex engineering analysis.

Kernels can be directly adapted to the geophysical models such as EULAG (Eulerian/semi-Lagrangian) fluid solver, which is an established computational model developed for simulating thermo-fluid flows across a wide range of scales

and physical scenarios. Currently, the model is being implemented as the new dynamic core of the COSMO (Consortium for Small-scale Modeling) weather prediction framework.

The algorithms have been extended by additional quantities as forces (implosion, explosion) and density vectors. In addition, they allow users to fully configure the border conditions (periodic, open).

## III. List of kernels

The goal of this work is to adapt 4 CFD kernels to Alveo U250 FPGA. All the kernels use 3-dimensional compute domain consisting of 7 (Thomas) to 11 (pseudovelocity) arrays. The computations are performed with a stencil fashion (to compute a single element of a compute domain it is required to access the neighboring elements). Since all the kernels belong to a group of memory bound algorithms, our main challenge was to provide the highest utilization of the global memory bandwidth.

- Advection (movement of some material, dissolved or suspended in the fluid). First-order step of the non-linear iterative upwind advection MPDATA [3] (Multidimensional Positive Definite Advection Transport Algorithm) schemes.
- Pseudo velocity [4] (approximation of the relative velocity) Computation of the pseudo velocity for the second pass of upwind algorithm in MPDATA. This kernel is the most compute intensive.
- Divergence [6] (measures how much of fluid is flowing into/out of a certain point in a vector field). Divergence part of the matrix-free linear operator formulation in the iterative Krylov scheme.
- Thomas algorithm (simplified form of Gaussian elimination for tridiagonal system of equations). Tridiagonal Thomas algorithm [6] for vertical matrix inversion inside preconditioner for the iterative solver. Preconditioner operates on the diagonal part of the full linear problem. Effective preconditioning lies at the heart of multiscale flow simulation, including a broad range of geoscientific applications.

## IV. Alveo optimized CFD codes

The main goal of this work is to provide an effective adaptation of the CFD kernels to the Xilinx Alveo U250 FPGA [7] built on the 16nm UltraScale architecture. The premiere of this card was on October 02, 2018. The key device features that requires a special effort during the development are the data clock frequency that can vary from 60MHz to 300MHz depending on the instruction flow, off-chip memory bandwidth that in theory is 77GB/s (about 47GB/s in practice - based on the Xilinx Xbutil tool). The off-chip memory is organized into four DDR4 memory banks, each of size 16GB (64GB of off-chip memory capacity), connected to one Super Logic Region (SLR). All the SLRs consists of 1341K Look-Up Tables (LUTs) responsible for arithmetic and logic operations, 2749K registers and 2K block RAMs (BRAM), each of size 36 Kb. Four SLRs creates the dynamic region available for creating custom accelerators. The remaining part of the FPGA includes the static region containing a deployment shell that handles device bring-up and configuration over Peripheral Component Interconnect Express (PCIe). The maximum total power of this card is 225W, however the FPGA allows user to significantly reduce this metrics for memory-bound algorithms.

As a development environment we selected the Open Computing Language (OpenCL) framework, as a most portable across different platform, including CPU, GPU, FPGA, and DSP. Our research contain performance and energy efficiency benchmarks of FPGA and CPU platforms based on a group of kernels characterized by different complexity and parallel strategy. Our goal is to provide a deep dive on the Xilinx Alveo U250 FPGA as a future candidate to accelerate current HPC codes.

To address 4 memory banks feature of the U250 card a compute domain of the kernels is divided into 4 sub-domains, where each of them is assigned to a separate memory bank. Each kernel is distributed across 4 compute units assigned to a different SLR. In this way, the memory transfers between the global memory and compute units occurred only between connected pairs of SLR and memory bank. The kernels belongs to a group of iterative algorithms, where each iteration represents a single time step of simulation. After each time step it is required to synchronize compute units and exchange halo areas (borders of sub-domain) between neighbouring sub-domains. The high level design of the kernels is shown in Fig. 1. Here we have a host (CPU) and device (FPGA) layer. The host layer is responsible for sending data to the FPGA global memory, calling kernels to execute them on the device, and receive the output from the kernels. The device layer is responsible for kernel processing:

- Kernel is distributed into 4 SLRs;
- Each sub-domain is allocated in different memory bank;
- Data transfer occurs between neighboring memory banks.

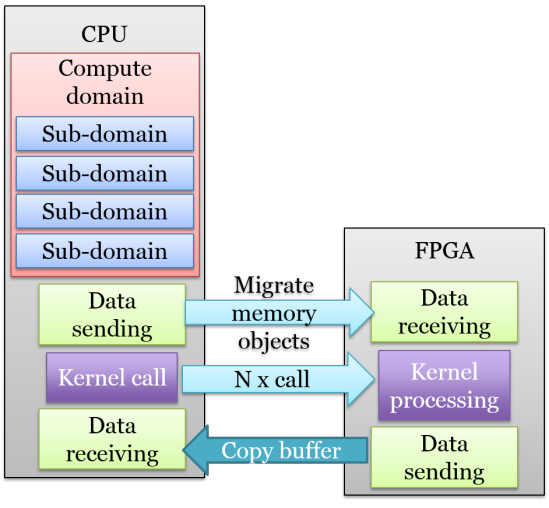


Fig. 1. The main layer of algorithm design.

Since all the kernels belong to a group of memory bound

algorithms, our main challenge is to provide the highest utilization of the global memory bandwidth. To provide an efficient data transfer between memory banks, we utilize a new memory object called pipe. A pipe stores data organized as a FIFO. Pipes can be used to stream data from one kernel to another inside the FPGA without having to use the external memory, which greatly improves the overall system latency.

To minimize global memory traffic, we use a BRAM memory. The characteristic of stencil computation requires to access a single array many times. Since there is not enough memory space to store 3D blocks of compute domains, we applied the 2.5D blocking technique to provide data locality. For this purpose, we stored only a small set of planes for each domain, which is stored as a queue of planes in BRAM. After each iteration, only a single plane is transferred from the global memory, while others migrate across the queue. In this way the global memory traffic is minimized. The overview of this method is shown in Fig. 2.

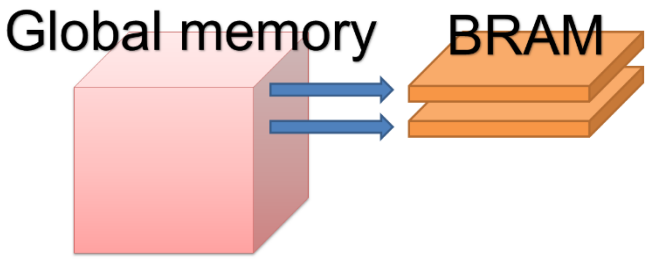


Fig. 2. 2.5D blocking technique between the global memory and BRAM.

- Each array is transferred from the global memory to the fast BRAM memory;
- To minimize the data traffic, we use a memory queue across iterations.

Another key optimization is to organize the computation in a SIMD fashion by utilizing vector data types of size 16. It allows us to utilize a 512-bit AXI4 memory interface for global memory access.

Lastly, here goes a summary of how each optimization let us speed up the execution of the advection kernel, ultimately cutting the time of execution from almost 600s to roughly 1s.

- O1: 1 SLR basic implementation, loop pipelining;
- O2: memory bank assignment, memory alignment;
- O3 :2 SLRs and 2 memory banks, using pipes to kernel communication, loop tiling;
- O4: vectorization added, kernel communication thru data copying;
- O5: BRAM used;
- O6: critical path optimized;
- O7: queue on BRAM memory;
- O8: optimization flags;
- O9: memory pins reduction (3SLRs), border conditions optimization and global memory traffic reduced;
- O10+: 4 SLRs.

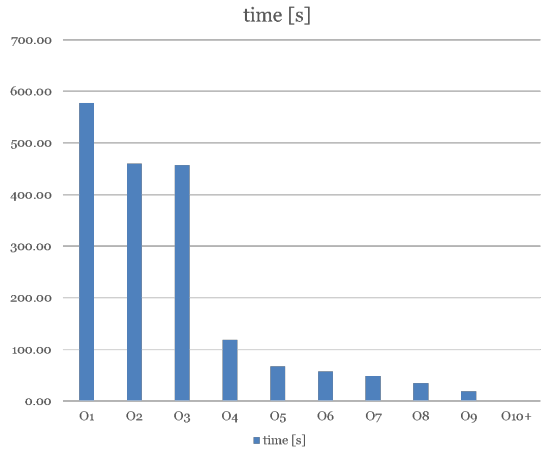


Fig. 3. Algorithm optimization results (FPGA).

## V. Benchmark Preparation

Our initial CPU implementation utilized 2 CPU processors: Intel® Xeon® CPU E5-2695 v2 2.40 – 3.2 GHz (2x12 cores). Then we compare the results with several other configurations, including: 1 x Intel Xeon E5-2695 CPU 2.4GHz - IvyBridge (Ivy), Intel Xeon Gold 6148 CPU 2.4GHZ - SkyLake (Gold) and Intel Xeon Platinum 8168 CPU 2.7GHZ - SkyLake (Platinum).

Our environment includes a server equipped with 128GB of RAM memory, Ubuntu 16.04 OS, Xilinx Runtime Alveo driver v.2018.3, Xilinx deployment Shell v.2018.3 – the communication layer physically implemented and flashed into the card, the Xilinx SDAccel IDE (Integrated Development Environment) v.2018.3 – framework for development.

To optimize the CPU code, we implemented several techniques including:

- all the available cores utilization;
- loop transformations;
- memory alignment;
- thread affinity;
- data locality within nested loops;
- compiler optimizations.

Depending on a kernel, the above techniques translated into an almost 92x speedup.

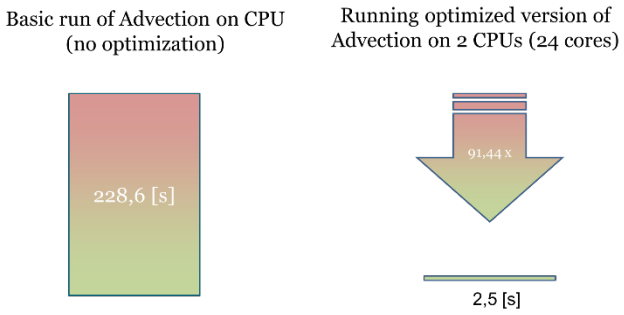


Fig. 4. Comparison of the basic CPU version with the optimized CPU version.

## VI. Systematic performance evaluation of FPGA

For the configuration with 2 CPUs (Ivy) we reached the maximum throughput of: 3.7 GB/s. Corresponding power dissipation was: 142 Watts.

The configuration with FPGA resulted in almost 6 GB/s throughput and the power dissipation of slightly above 100 W.

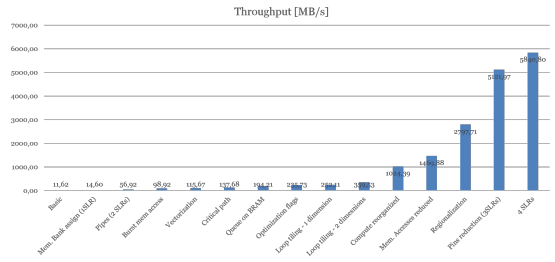


Fig. 5. Throughput results for each optimization applied to FPGA code.

Also, still speaking of FPGA, the results for the advection kernel were as follows:

- Read Data: 467.92704 GB (domain 1020x510x64; time steps: 500);
- Execution time: 9.96 s;
- Achieved memory throughput: 46.927 GB/s;
- Maximum throughput of Alveo U250 measured with Xbutil tool: 46.981 GB/s.

Based on the achieved results we conclude that the kernels execution time is equal to the data transfer time. It means that the computations are almost fully overlapped with memory access. It is very important to note that we reached 98.32% of the maximum attainable throughput. In that case, we optimize the performance to the level that the time of execution was completely hidden behind the time of the data transfer with a maximum possible throughput. The performance for various configurations are shown in Fig. 6, while the performance/W results are shown in Fig. 7. In Fig. 8 we show the output of the advection kernel after 250 time steps of the simulation.

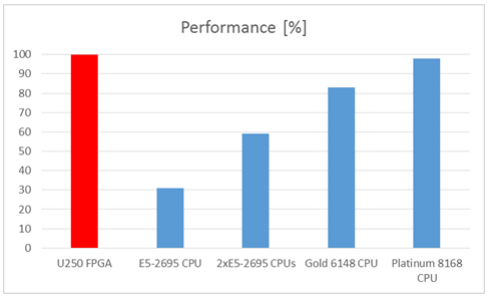


Fig. 6. Comparison of performance results between CPU and FPGA.

As we can see, a single Alveo U250 card is able to outperform even Intel Xeon Platinum 8168 CPU, delivering results slightly faster and at a significantly lower energy budget.

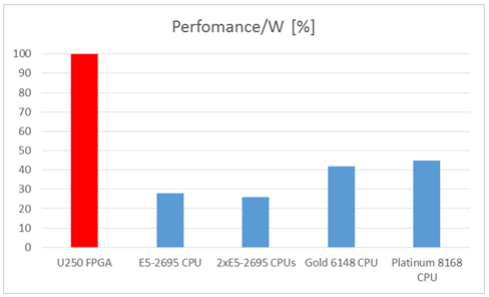


Fig. 7. Comparison of performance/W results between CPU and FPGA.

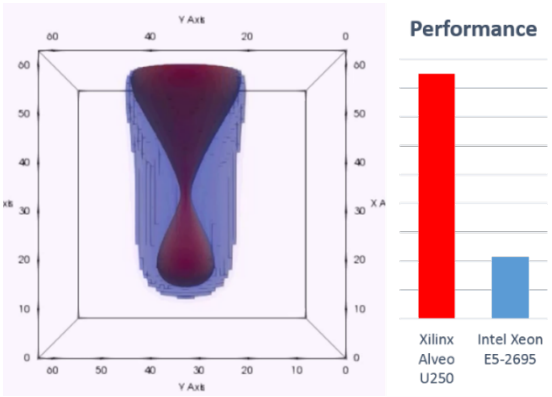


Fig. 8. Test-case simulation overview.

It is important to emphasize, that the presented results are for single kernels. Typical CFD applications consist of many kernels which execute various operations on the same data. Therefore, we should expect even better results due to further possibility to optimize the computations and reuse data across all the kernels.

## VII. Results summary

The proposed adaptation allows us to achieve up to 4x speedup over the CPU version of kernels, up to 80% less energy consumption, and up to 6x higher performance per watt. The developed kernels are fully compatible with EULAG geophysical model supporting all the required parameters including domain settings, border conditions, and hardware characteristics.

- 4x faster results;
- 80% lower energy consumption;
- Up to 6x better Performance per Watt;
- Compatible with geophysical models like EULAG;
- Configurable code (domain settings, border conditions, data; precision, and hardware characteristics).

## VIII. Early tests with U280

As the kernels are memory bound, we also performed initial tests with another Alveo card: U280. Therefore, for the advection kernel, we distributed computations across 3 SLRs and shared the computational domain (data) with 18 (out of

32 as using more led to significant frequency dropping) HBM banks. The results were as follows:

- 1.2 speedup vs. U250
- 42 vs 53 Watts consumed on U250
- 1.39 better performance / Watt vs. U250 (energy measurement includes: RAM, CPU and Alveo)
- 29% lower energy consumption

The additional speedup is the result of using much faster memory in U280 (410GB/s vs 77GB/s). As for U250, the algorithm has been developed at a 98.3% of its peak attainable performance. Therefore, further optimizations on the software level were not possible. And while moving to an HBM-based architecture, we redesigned the memory allocations and compute organization within the algorithm. However further study would be required to assess the maximum possible speedup for this particular algorithm. Energy consumption reduction is a result of various things: 3SLRs in U280 vs. 4 SLRs in U250, we do not use DDR and currently, we have not yet utilized it at its peak attainable performance.

It is also worth mentioning that the algorithm we used is relatively complex in terms of computations but also when it comes to the amount of data and its structure. Talking about the data, these types of algorithms usually require 64GB of memory to handle complex scenarios. Less memory is usually enough for small to mid-size simulations. Also, due to computations and algorithms design, it was hard to utilize all the 32 memory banks of HBM. When we tried to do so, the frequency of SLRs dramatically decreased. Hence the current speedup is currently based on utilizing only 18 out of 32 memory banks.

## IX. Brief TCO analysis

Alveo is an accelerator card designed to meet the constantly changing needs of the modern Data Center, supporting any workload type while reducing overall cost of ownership. To confirm the designer's assumptions, it is important to verify not only a raw performance and power metrics but also validate an attainable performance, platform limitations and estimate adaptive possibilities of CFD algorithms for this architecture. This goal is achieved by providing a deep analysis of the selected algorithms with the internal profile tools of the Alveo environment.

As mentioned, typical CFD applications utilize a number of various kernels which execute operations on the same data.

TABLE I
Nodes configuration for TCO calculations

| Setup | CPU [USD] | Accel. [USD] | Price [USD] | # of dev. per node | Tot. price [USD] |
|---|---|---|---|---|---|
| Intel Xeon Platinum 8168 | 8000 | 0 | 8000 | 2 | 16000 |
| Xilinx U250 +Intel Xeon E3 1220 | 200 | 5000 | 5200 | 1 | 5200 |

Moving from CPU-only nodes to CPU+FPGA nodes translates into a 10x better TCO when the speedup reaches roughly under 4x.

TABLE II
TCO calculations for algorithms speedup

| TCO CPU/FPGA | Required speedup over 2 CPUs |
|---|---|
| 1 | 0.325 |
| 2 | 0.65 |
| 3 | 0.975 |
| 4 | 1.3 |
| 10 | 3.25 |

## X. Conclusion

Moving fluid simulations to heterogeneous computing architectures powered by Alveo FPGA delivers faster results within significantly reduced energy budgets. For instance, nodes equipped with Alveo U250 deliver up to 4x speedup while reducing the energy consumption by almost 80% compared to CPU-only nodes. As these algorithms are memory bound, upgrading the configuration to U280 (equipped with HBM) gives additional speedup (1.2 when we utilized 18 out of 32 memory banks; further study required) and reduces the energy budgets further by additional 29%. CFD market needs speedup, heterogeneous architectures ready solutions and scalability. Alveo products family addresses these challenges very well. Moreover, CFD codes fit well into the Alveo architecture features such as multi banks, BRAM utilization, pipelining, vectorization etc. FPGAs introduce certain limitations like for example:

- the performance drops when we exceed the maximum supported amount of computations that can be executed at highest throughput. Beyond that point we need to share the resources;
- available bandwidth between the host memory and FPGA can limit the overall performance.

Therefore, FPGAs are good candidates for applications where:

- most of the computations are organized in small codes which are repeated on larger data portions;
- computational load is much higher than input/output operations, utilizing and gradually saturating the host memory – FPGA available bandwidth.

Based on this, CFD codes in general are very good candidates to benefit from the FPGA architectures. In general, all the CFD codes contains suitable properties for the Alveo card. In details, we distinguish three core differences between the code adaptation to CPU and FPGA. The first one is a large cache memory storage of CPU versus a relatively small but ultra fast BRAM of FPGA. It gives an advantage for kernels where the same memory space can be reused to store a different arrays of compute domain depending on the kernel stage. To this group belongs advection and pseudovelocity. Here a data locality is sufficiently provided by the 2.5D blocking

technique that allows us to reduce data traffic between the global and local memory. In contrast, the divergence kernel requires to keep many arrays of compute domain in the local memory through the entire process. This property makes the algorithm more convenient for implementation with using the CPU-based programming model. The second differentiator of FPGA is lower frequency but higher parallelism than CPU. A really large set of FPGA compute units allows us to efficiently overlap computations with data transfers by using an intensive pipelining with dual memory access.

## REFERENCES


[1] A. Ebrahimi, M. Zandsalimy, "Evaluation of FPGA Hardware as a New Approach for Accelerating the Numerical Solution of CFD Problems," in IEEE Access, vol. 5, pp. 9717–9727, May 2017.

[2] S. Kim, R. A. Rutenbar, "An Area-Efficient Iterative Single-Precision Floating-Point Multiplier Architecture for FPGA," Proceedings of the 2019 on Great Lakes Symposium on VLSI, pp. 87–92, May 2019.

[3] K. Rojek, "Machine learning method for energy reduction by utilizing dynamic mixed precision on GPU-based supercomputers," Concurrency and Computation: Practice and Experience, vol. 31(6), March 2019.

[4] K. Rojek, R. Wyrzykowski, "Performance modeling of 3D MPDATA simulations on GPU cluster," The Journal of Supercomputing, vol. 73(2), pp. 664–675, February 2017.

[5] K. Rojek, R. Wyrzykowski, L. Kuczynski, "Systematic adaptation of stencil-based 3D MPDATA to GPU architectures," Concurrency and Computation: Practice and Experience, vol. 29(9), May 2017.

[6] P. K. Smolarkiewicz, C. Kuhnlein, N. P. Wedi, "A consistent framework for discrete integrations of soundproof and compressible PDEs of atmospheric dynamics," Journal of Computational Physics, vol. 263, pp. 185–205, January 2014.

[7] Vivado Design Suite User Guide (UG908), Xilinx, 2019