Proceedings of the 2023 International Conference on Mechatronics and Smart Systems
DOI: 10.54254/2755-2721/9/20230111

Specialized fluid estimation implementation on the FPGA

Haowen Yang

Herbert Wertheim College of Engineering, University of Florida, Gainesville, Florida, 32611, USA

haowenyang@ufl.edu

Abstract. This paper studies the possibility of exploring Field Programmable Gate Array (FPGA) in the acceleration of Computational Fluid Dynamics (CFD). CFD is an industrial analysis tool to estimate the flow of matter. In the previous experience, CFD are most implemented on the conventional CPU, and may accelerate with GPUs in a high-performance computing center. This paper studies the architecture of the FPGA and compared the FPGA to CPU and the application and algorithm of CFD. We studied the previous works from different research and found that CFD occupies a huge advantage on the efficiency of the overall system. FPGA utilize less hardware resources, and usually presents a less computing time and higher throughput of data. So, FPGA is a viable and cost-effective solution for future alternative of the CFD computation with the same cost of constructing a high-performance computing center.

Keywords: Computational Fluid Dynamics, fluid estimation, FPGA.

1. Introduction

We see the dawn of FPGA replacing CPU and GPU utilization in other calculation fields, such as Artificial Intelligence, and obtaining a considerable improvement in efficiency from the FPGA implementation. [6] FPGA sees its future when implementing CFD calculation onto the platform. CFD is a widely applied area of study with many essential applications in the industry. CFD can perform many different types of estimation simulation, such as flow transfer, heat transfer, etc. [1] CFD offers the possibility to research and develop in an efficient and environmental-friendly manner since many simulations can be done on a computer. We briefly discussed the basic theoretical knowledge of CFD. FPGAs are the prefabricated chips, equipped with configurable blocks. And the blocks can be programmed to be interconnected to suite the types of programs required for the calculation. And thus, FPGA offer an extraordinary flexibility compared to the CPU and GPU. This study offers an overview to discuss the future and feasibility of utilizing FPGA as a potential replacement of CPU and GPU in CFD calculation.

2. Computational fluid dynamics – basic knowledge analysis

2.1. Background of computational fluid dynamics

Computational Fluid Dynamics, known widely as CFD, is an analysis tool to predict fluid behaviour in heat transfer, fluid flows or chemical reactions, and many other areas of industry. CFD solves the mathematical equation to estimate the behaviour of the flow in the numerical process. There are three

© 2023 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

general approaches when calculating fluid dynamics: the Finite Difference Method, Finite Element Method, and Finite Volume Method [1]. All these three approaches involve the partial differential equation to calculate within a finite volume. In a complete CFD process, the geometry will be prepared for mesh generation, which is a step to generate the grid as a finite volume or finite surface to apply the solvers. Then, the boundary condition will be specified. The boundary condition aims to define a stopping layer where the solver no longer needs to be applied. Then, each grid will be processed, and flow will be solved in each grid. At this point, there will be a cluster of data generated. So, the data will be post-processed to visualize the results of the solution [2]. Sometimes, the post-processing step was done with an external tool to visualize the data. Sometimes, one more step is added before determining the boundary condition: Domain decomposition. Since most CFD software runs under multiple calculating cores, an automatic step is to assign the entity into small segments and perform operations under each small grid.

CFD has become a dependable tool for engineers and researchers to research and develop new models of products without the continuous production of prototypes, which is highly costly; instead, performing simulations provide an excellent physical estimation of the object [3]. CFD tools were a significant assistant for many aerodynamics works in motorsport and aerospace engineering. CFD tools allow the engineer to simulate the aerodynamic effect of a designed geometry. For example, the B777 aircraft’s design process benefits from the maturing CFD tools, and such advantages of CFD tools are also applied to the design of the B787 aircraft. In motorsports, specifically in Formula 1, a motorsport class in which each race team designs the aerodynamics feature of their cars based on the published regulation by FIA and competes in every year’s championship, it is a heavily aerodynamics-reliant industry. CFD tool was introduced in the 1990s to assist in the design of the racing car.

2.2. CFD-calculation method

CFD calculation involves the non-linear “Navier Stokes Equations” as the governing equation of the CFD calculation. The equation describes the conservation of mass, momentum, and energy of the entire system. The tensor form of the expression of the equation is as follows table 1[4].

Table 1. The tensor form of the expression of the equation

ρ	Density ($kg \cdot m/s$)	V	Velocity (m/s)
p	Pressure (Pa)	g	Gravitational force (N)
k	Thermal conductivity ($W/m \cdot k$)	h	Enthalpy (J)
C_p	Specific heat (constant pressure) ($J/(kg \cdot k)$)	μ	Viscosity ($kg/(m \cdot s)$)
C_v	Specific heat (constant volume) ($J/(kg \cdot k)$)	R	Gas constant ($J/mol \cdot K$)
T	Temperature (K)	∇	Gradient operator

The equation applies to a control volume as an enclosed system. As an enclosed system, the total mass of the system should remain unchanged. Since the Navier Stoke Equation describes the relationship between all the above parameters: density, pressure, temperature, velocity, enthalpy, and the viscosity of the fluid. So, we replace mass with density because density is a unique property of matter.

So, the total partial differential of the density respective to time (mass flux) and mass flux respective to the velocity of each direction (described as $\nabla \cdot \underset{V}{\rightarrow}$) is zero.

The conservation of mass:

$$\frac{D\rho}{Dt} + \rho\left(\nabla \cdot \underset{V}{\rightarrow}\right) = 0 \quad (1)$$

The conservation of momentum can be derived from the second Newton's law. The overall change of the momentum is equal to the change of momentum on the surface. As we derived from the second Newton's Law and taking the derivative with respect to time, we get follow:

The Conservation of Momentum:

$$\rho\frac{DV}{Dt} = \rho g - \nabla p + \partial/\partial x_i[\mu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) - 2/3\frac{\partial v_r}{\partial x_r}\delta_{ij}] \quad (2)$$

The energy conservation law state that the total amount of energy is equal to the sum of the work and energy applied to the system. The equation can be expressed as follow:

The Conservation of Energy:

$$\rho\left[\frac{\partial h}{\partial t} + \nabla \cdot (hV)\right] = -\frac{\partial p}{\partial t} + \nabla \cdot (k\nabla T) + \emptyset \quad (3)$$

$$\emptyset = \ [\mu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) - 2/3\frac{\partial v_r}{\partial x_r}\delta_{ij}]\frac{\partial v_i}{\partial x_j} \quad (4)$$

2.3. Implementation of CFD on computer:

CFD requires tremendous computer power in its overall computing process since the process has a lot of floating points operating on the differential equation solver. The most widely applied method to accelerate CFD is hardware acceleration. Previous work has been done to implement the CFD on the combination of CPU and GPU processing to maximize the hardware on the computing device [5]. Also, GPU has an overall advantage in floating point and parallel calculation, so GPU is a solid solution for CFD acceleration.

The previous work implemented an FPGA-CPU combination to perform the CFD calculation [6]. The previous work showed a significantly lower solving time than the generic CPU implementation method.

3. FPGA basic knowledge analysis of FPGA

3.1. Basic FPGA architecture

FPGA, or Field Programmable Gate Arrays, is a pre-fabricated silicon device that can be programmed to implement a specific system based on the user's needs in almost any form. The FPGA comprises configurable logic blocks (CLBs), interconnecting routing resources, and configurable I/O blocks. These blocks are mapped in a 2D geometry, and the interconnecting routing resources are responsible for connections between the CLBs and I/O blocks on the chip. For improved performance of FPGA, Look Up Table (LUT) was introduced in the FPGA [7].

LUT is made of SRAM bits that store the LUT masks, and the input selects the bits from the LUT masks through sets of multiplexers. And advantages of implementing the LUTs on FPGA include that LUT is programmable, which leads to the great flexibility of the input-output combination when implementing some complicated mathematical operation. LUT is an effective method to reduce the delay in the process, as any logic combination can be implemented with LUTs. For n numbers of input, the LUT required 2^n bits of memory to store the logic values. To further improve the performance of FPGA, some manufacturers implement DSP blocks [8]. DSP block is a pre-designed block containing many multipliers, registers, and adders to perform floating point calculations. These blocks are even advantageous when performing some particular type of math calculation. As the hardware resources became more in demand to perform some tasks, FPGA manufacturers also included an internal memory block to increase data access efficiency.

3.2. Basic FPGA architecture

Estimating the cost efficiency of FPGA, CPU, and General Purpose GPU (GPGPU) is hard. To estimate the cost of an FPGA, there is a difference in cost for the customized and non-customized board. Also, the development cost and operation costs have to be considered. In addition, the CPU and GPU would also require a host device to operate, while FPGA can run stand-alone [9]. In the previous work, FPGA had over 100 times faster computing time for the same acquisition price than the CPU.

Estimating the energy efficiency of FPGA, CPU, and GPGPUs is pretty accurate since the number of operations accomplished or the number of samples being processed per joule is a measurable figure. In the previous work, the FPGA also shows over 100 times more efficient than CPU and GPU compared to the sampled processed by a joule of energy [10]. Another study reveals that CPU clusters while consuming 950W of power, are longer at performing tasks than FPGA, which runs under 20W of power.

3.3. FPGA flexibility

FPGA has a natural advantage over flexibility because FPGA is a reconfigurable device. The FPGA contains the configurable logic block, I/O blocks, and configurable interconnections. So, FPGA can be programmed to favour the type of calculation. While the CPU and GPU are prefabricated, all the hardware on the board is fixed in connection and function. So, CPU and GPU are worse in flexibility compared to FPGA. Unlike CPU and GPU, which have a lot of black silicon, which is not activated when performing a specific task, the FPGA is implemented to fully utilize the logic gate on the chip to maximize the hardware resource.

As we can see, FPGA has been proven to show better performance and better efficiency when compared to CPUs and GPGPUs when performing the same tasks.

3.4. FPGA application in artificial intelligence

One field that has been widely applying FPGA is artificial intelligence. AI involves a very complex mathematical operation. For example, one of the object detection algorithms: HOG, involves much floating-point operation and a tangent operation, which involve the utilization of the adder and multiplier block. The previous work has shown that when implementing the FPGA on the computer vision algorithm model, FPGA shows a dominating efficiency compared to GPU and CPU. Even though the throughput of FPGA is less than GPU because GPU has a much higher frequency, FGPA was able to achieve over 5 times better energy efficiency.

FPGAs are also appearing in the data centre as the replacement of CPUs. In the test implementation, the FPGA also showing low latency, and high throughputs when not utilizing full hardware resources. Another study shows that the FPGA has a low latency advantage when implementing the Convolutional Neural Network.

With a plentiful of instances where FPGAs are implemented to increase the efficiency and reduce the latency. In this study, we try to implement CFD onto FPGA board and compared the performance to a CPU and GPUs.

4. CFD acceleration with FPGA

In the previous works, there are two instances where FPGA and CPU are making a direct comparison on the solving time and throughput of FPGA and CPU platform when performing a same CFD kernel or a same CFD solving equation.

With the Alveo U250 card, the previous study conducts development to implement 4 different CFD kernels on to the FPGA platform. The Advection, Pseudo velocity, Divergence, and Thomas algorithm were studied and implemented on the U250 FPGA. The U250 utilized a combination of external memory and internal memory. Four DDR4 memory block, in total of 64GB, are connect to the four Super Logic Region (SLR), where each region contains an 1341K LUTs, 2749k registers, and 2K block of RAM, each of size 36Kb [11]. Then the SLRs are connection to the static region to deal with the I/O port connection. One of the focuses of the work is to optimize the usage of the BRAMs.

Each Kernal has their unique characteristics what would affect the performance of the processing. Pseudo velocity are considered the most computing intensive kernel due to the logarithm itself. Divergence kernel is very memory-dependent because the algorithm would take nine inputs and return one output, which proposed a challenge on the memory bandwidth. The Thomas algorithm is extremely hard for the parallelization due to the algorithm type and data dependency. Since the kernels required an array data as an input and export another array of data as an output after the kernel processing. So, the high-bandwidth data transmission would become extremely beneficial to augment the performance of these kernel. So, the 2.5D blocking was developed: for data migrating between each layer of the BRAM, there is only one layer of data downloaded into the BRAM block from the global memory [12]. So, the overall demand on the memory bandwidth is decreased. From the result of the simulation, we see that FPGA overall gain a better performance compared to the CPU performing the same kernel. But it also reveals a weakness of FPGA when dealing with the algorithm that is difficult to parallelize. Since the CPU has a much higher frequency than FPGA, the CPU outperforms the FPGA.

Another study has been done when estimating the performance of the FPGA and CPU under different scenario: Ordinary 2nd Order Differential Equation, Laplace Equation, and Quasi-One-Dimensional Inviscid Compressible Flow. For the CPU-FPGA connection, there are several different forms of combination to one another. One way is CPU as host processor, and FPGA is programmed to execute the operation as needs from the CPU [13]. And the coprocessor is an enlarged version of host-processor formation, while the coprocessor would receive the initial instruction and data from the CPU, and then operation individually. Also, a joint compressor can form an extra processor under a multi-processor formation. FPGA can also choose to operation as an individual processor. This study, both CPU and FPGA are operating as an independent unit. For all three types of calculation, with the premise of the similar accuracy, FPGA are utilizing less power yet gaining much faster solution time.

Taking the advantage of the flexibility and high area efficiency of FPGA, previous work has made an attempt to implement a real time CFD estimation tool, to simulate the fuel system of a diesel engine. One of the challenges of implementing a CFD as real-time machine is the calculation amount and memory. The previous notes that even though GPU has multiple computing cores and is excellent at homogeneous parallel calculations, GPU is showing high latency when accessing the memory block from the computing core. Also, CPU is overall not optimum for real-time CFD due to its limitation on core numbers and overall throughput of data. And the researchers utilize Precision Timed (PRET) architecture and an overall distribution circuit to reduce the latency of overall computation and memory access.

5. Conclusion

From the previous work, FPGA proved to be a great solution for accelerating CFD computation. In previous work where FPGA are directly compared to server level conventional CPU, FPGA can reach over three to five times of improvement on performance when implementing the same solver. FPGA show its great potential on the shorter computing time and higher energy efficiency. FPGA has a massive future on applying in the industry such as vehicle development, aeronautical analysis. However, current work on implementation of FPGA in CFD calculation are still limited, and not yet widely applied in the industry. Also, CFD is a high compacted calculation tool with different distributors. Each distributor optimizes their software based on different solver and different preference on the CPU and GPU usage. So, more future work can be done to study the real-world possibility of applying FPGA in the CFD tool.

References

[1] M. Mani and A. J. Dorgan, “A Perspective on the State of Aerospace Computational Fluid Dynamics Technology,” Annual Review of Fluid Mechanics, vol. 55, no. 1, pp. 431–457, Jan. 2023.
[2] M. H. Zawawi et al., “A review: Fundamentals of computational fluid dynamics (CFD),” Green Design and Manufacture: Advanced and Emerging Applications, vol. 2030, no. 1, 2018.
[3] J. F. Wendt, J. D, and E. Al, Computational fluid dynamics: an introduction. Berlin: Springer, 2010.
[4] M. L. Hosain and R. B. Fdhila, “Literature Review of Accelerated CFD Simulation Methods towards Online Application,” Energy Procedia, vol. 75, pp. 3307–3314, Aug. 2015.

[5] K. Fujii, “Progress and future prospects of CFD in aerospace—Wind tunnel and beyond,” Progress in Aerospace Sciences, vol. 41, no. 6, pp. 455–470, Aug. 2005.
[6] D. B. Thomas, L. Howes, and W. Luk, “A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation,” Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays, Feb. 2009,.
[7] C. Du, I. Firmansyah, and Y. Yamaguchi, “FPGA-Based Computational Fluid Dynamics Simulation Architecture via High-Level Synthesis Design Method,” Applied Reconfigurable Computing. Architectures, Tools, and Applications, pp. 232–246, 2020.
[8] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li, “Computer vision algorithms and hardware implementations: A survey,” Integration, vol. 69, pp. 309–320, Nov. 2019.
[9] S. Gandhare and B. Karthikeyan, “Survey on FPGA Architecture and Recent Applications,” IEEE Xplore, Mar. 01, 2019. https://ieeexplore.ieee.org/abstract/document/8899550 (accessed Mar. 28, 2023).
[10] U. Farooq, Z. Marrakchi, and H. Mehrez, “FPGA Architectures: An Overview,” Tree-based Heterogeneous FPGA Architectures, pp. 7–48, 2012.
[11] I. Liu, E. A. Lee, M. Viele, G. Wang, and H. Andrade, “A Heterogeneous Architecture for Evaluating Real-Time One-Dimensional Computational Fluid Dynamics on FPGAs,” 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines, Apr. 2012.
[12] K. Rojek, K. Halbiniak, and L. Kuczynski, “CFD code adaptation to the FPGA architecture,” The International Journal of High-Performance Computing Applications, vol. 35, no. 1, pp. 33–46, Nov. 2020.
[13] A. Ebrahimi and M. Zandsalimy, “Evaluation of FPGA Hardware as a New Approach for Accelerating the Numerical Solution of CFD Problems,” IEEE Access, vol. 5, pp. 9717–9727, 2017.