

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

**FPGA BASED HARDWARE ACCELERATOR
FOR EULER EQUATIONS WITH
FINITE VOLUME METHOD**

M.Sc. THESIS

Emine Elif YİĞİT

Department of Defence Technologies

Defence Technologies Programme

JUN 2024

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

**FPGA BASED HARDWARE ACCELERATOR
FOR EULER EQUATIONS WITH
FINITE VOLUME METHOD**

M.Sc. THESIS

**Emine Elif YİĞİT
(514211005)**

Department of Defence Technologies

Defence Technologies Programme

Thesis Advisor: Asst. Prof. Dr. Ramazan YENİÇERİ

JUN 2024

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

**EULER DENKLEMLERİ İÇİN SONLU HACİMLER YÖNTEMİ İLE
FPGA TABANLI DONANIM HIZLANDIRICI**

YÜKSEK LİSANS TEZİ

**Emine Elif YİĞİT
(514211005)**

Savunma Teknolojileri Anabilim Dalı

Savunma Teknolojileri Programı

Tez Danışmanı: Asst. Prof. Dr. Ramazan YENİÇERİ

HAZİRAN 2024

Emine Elif YİĞİT, a M.Sc. student of ITU Graduate School student ID 514211005 successfully defended the thesis entitled “FPGA BASED HARDWARE ACCELERATOR FOR EULER EQUATIONS WITH FINITE VOLUME METHOD”, which she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : **Asst. Prof. Dr. Ramazan YENİÇERİ**
Istanbul Technical University

Jury Members : **Prof. Dr. İbrahim Özkol**
Istanbul Technical University

Asst. Prof. Dr. Barış KARAKAYA
Fırat University

Date of Submission : **23 May 2024**
Date of Defense : **25 Jun 2024**





Anneme, Babama ve İlhan'a



FOREWORD

I would like to express my deepest gratitude to my advisor, Dr. Ramazan YENICERI, for the invaluable opportunities he has provided me and the knowledge he has imparted. His guidance has been a cornerstone of my academic journey, illuminating the path with his expertise and insight.

I also extend my heartfelt thanks to my family —my mother Betul YIGIT, father Omer YIGIT, and brother Mehmet YIGIT— for their unwavering support throughout this endeavor. Their belief in me and constant encouragement have been my strength and motivation, enabling me to pursue my aspirations with confidence.

Lastly, I must acknowledge my dear friend Ilhan PALA, who has been an unwavering source of support. His patience in answering every question, and his ability to lift me up whenever I faltered, have been invaluable. Ilhan's friendship and guidance have been a constant throughout my academic life, for which I am eternally grateful.

This thesis would not have been possible without the combined support, encouragement, and guidance of these remarkable individuals. To each of them, I offer my sincerest thanks for their significant contributions to my academic journey and personal growth.

Sincerely,

Jun 2024

Emine Elif YİĞİT
(Aeronautical Engineer)

TABLE OF CONTENTS

	<u>Page</u>
FOREWORD	ix
TABLE OF CONTENTS	xi
ABBREVIATIONS	xiii
SYMBOLS	xv
LIST OF TABLES	xvii
LIST OF FIGURES	xix
SUMMARY	xxi
ÖZET	xxv
1. INTRODUCTION	1
1.1 Motivation	3
1.2 Aim of This Thesis	4
1.3 Accelerating CFD	4
1.3.1 FPGA and MBD Approaches	4
1.3.2 Literature Review	7
1.4 Thesis Outline	9
2. HARDWARE DESIGN METHODOLOGY	11
3. INITIAL MODELLING	15
3.1 Finite Volume Method	15
3.2 Second-Order Upwind Scheme	16
3.3 Mathematical Expressions	19
3.3.1 Continuity equation	20
3.3.2 Momentum equations	22
3.3.3 Energy equations	27
3.3.4 Temperature	31
3.3.5 Pressure	31
3.4 Initial Model Algorithm	32
4. ASSESSMENT OF INITIAL MODEL ACCURACY	35
4.1 Ansys Modelling	35
4.2 Assessment	37
5. MODEL ELABORATION AND ASSESSMENT	41
6. CONCLUSION and FUTURE WORK	45
6.1 Conclusion	45
6.2 Future Work	46
REFERENCES	47
APPENDICES	51
APPENDIX A : Elaboration of Continuity Equation	53
APPENDIX B : Elaboration of Momentum Equation	55
APPENDIX C : Elaboration of Energy Equation	59
APPENDIX D : Elaboration of Temperature and Pressure	63
CURRICULUM VITAE	65



ABBREVIATIONS

FPGA	: Field Programmable Gate Array
GPU	: Graphics Processing Unit
CPU	: Central Processing Unit
FVM	: Finite Volume Method
CFD	: Computational Fluid Dynamics
MBD	: Model Based Design
HDL	: Hardware Description Language
HLS	: High Level Synthesis
LUT	: Look-Up Tables
LUTRAM	: LUT Random Access Memory
DSP	: Digital Signal Processors
FFs	: Flip-Flops
BUFGs	: Globally Routed Buffers



SYMBOLS

u	: Velocity component in x direction
v	: Velocity component in y direction
V	: Velocity
U	: Any property of the flow
ρ	: Density
T	: Temperature
p	: Pressure
K	: Kinetic energy
cp	: Specific heat constant
E	: Internal energy
R	: Gas constant
i	: x -directional coordinate
j	: x -directional coordinate
e	: East face of the cell
w	: West face of the cell
n	: North face of the cell
n	: East face of the cell
N	: Cell number in x -direction
M	: Cell number in y -direction



LIST OF TABLES

	<u>Page</u>
Table 4.1 : Set-up Parameters	37
Table 5.1 : Resource Utilization for Equations	42





LIST OF FIGURES

	<u>Page</u>
Figure 2.1 : Idia of the proposed model-based hardware design.	12
Figure 2.2 : Flowchart of the model-based hardware design methodology.	13
Figure 3.1 : Schematic representation of finite volume discretization.	16
Figure 3.2 : Schematic representation of second-order upwind scheme.	18
Figure 4.1 : Mesh structure and velocity inlet and outlet.	36
Figure 4.2 : Comparison of <i>density</i> results between ANSYS Fluent and MATLAB script.	38
Figure 4.3 : Comparison of <i>pressure</i> results between ANSYS Fluent and MATLAB script.	38
Figure 4.4 : Comparison of <i>x-velocity (u)</i> results between ANSYS Fluent and MATLAB script.	39
Figure 4.5 : Comparison of <i>y-velocity (v)</i> results between ANSYS Fluent and MATLAB script.	39
Figure 4.6 : Comparison of <i>energy</i> results between ANSYS Fluent and MATLAB script.	40
Figure 4.7 : Comparison of <i>temperataure</i> results between ANSYS Fluent and MATLAB script.	40



FPGA BASED HARDWARE ACCELERATOR FOR EULER EQUATIONS WITH FINITE VOLUME METHOD

SUMMARY

This thesis presents a detailed investigation into the acceleration of Computational Fluid Dynamics (CFD) simulations through the innovative use of Field-Programmable Gate Arrays (FPGAs). This research is motivated by the need to overcome the limitations imposed by traditional theoretical and experimental approaches in flow analysis, to accelerate analyses, and to reduce costs.

Theoretical models have made significant strides in understanding fluid dynamics; however, they often falter when faced with the computational challenges of solving complex equations known for their difficulty, such as the Navier-Stokes equations, which are listed among the "Millennium Prize Problems". On the other hand, experimental methods provide valuable real-world insights through wind tunnel experiments, but are limited due to high operational costs, restricted control over test conditions, and the difficulty of obtaining precise measurements.

CFD emerges as a third and promising approach, advocating for a more accurate and cost-effective methodology to study fluid dynamics. Utilizing numerical methods and algorithms to simulate fluid motion, CFD exhibits distinct advantages such as applicability to complex geometries, the capacity to effectively complement physical experiments, and significant potential for cost and time reduction. Nevertheless, the reliability of CFD simulations heavily relies on the quality of the model and the chosen solution strategies, requiring extensive computational power to accurately model complex geometries and physical phenomena.

In previous studies, solutions for CFD have been significantly accelerated with the assistance of hardware such as Central Processing Units (CPU), Graphics Processing Units (GPU), and FPGAs. While CPUs are designed for general-purpose computations, GPUs offer advantages in tasks requiring high parallelism. FPGAs are programmable logic chips that can be configured by users to perform specific tasks. By allowing the functionality of the hardware to be altered through software, FPGAs become flexible and re-programmable for various applications. This study suggests the use of FPGAs, recognized for their re-programmable structures and superior performance capabilities, to meet the demanding computational requirements of CFD simulations. Thanks to their customizable architecture, FPGAs can be optimized for specific tasks, presenting a promising solution for the challenges associated with CFD computations.

For this study, a Model-Based Design (MBD) approach is adopted, detailing the process of generating synthesizable Hardware Description Language (HDL) code from MATLAB scripts via an HDL coding module. This methodology aims not

only to improve the design process but also to enhance the reliability of FPGA programming and to establish a robust framework for developing an FPGA-based hardware accelerator designed specifically to speed up CFD solutions. The research focuses particularly on developing a solver for the Euler Equations, which are crucial for modeling high-speed fluid flow. After the design is carried out according to this methodology, its accuracy will be checked against the commercial software Ansys Fluent. The first model, once its accuracy is confirmed, will be detailed and transferred to the MATLAB HDL Coder tool for generating the HDL code. After the HDL code is generated, an analysis of FPGA resource usage and the time it takes to solve for an element is planned. Thus, the performance of the designed hardware accelerator will be evaluated against previous studies.

The thesis initially discusses hardware design methodology and provides information about the proposed design method. This design methodology begins with the discretization of the Euler Equations, obtained using the Finite Volume Method (FVM) and a second-order upwind method. These equations, fundamental for modeling inviscid, steady, and compressible flow conditions, are then converted into MATLAB script form. This process results in the creation of core code files that form the backbone of the MATLAB simulator developed for this thesis. The simulator represents the culmination of the initial modeling phase, encompassing the entire simulation process by iteratively solving the discretized equations.

The simulation algorithm covers in detail the steps required to perform CFD analysis in a computational domain of 100 cells. The computation starts with the determination of initial conditions for various flow properties, including velocity components, energy, temperature, and pressure, which lay the groundwork for the simulation. The density is initialized using the ideal gas equation. The algorithm then solves the Continuity, Momentum, and Energy equations in parallel to update the fluid's density, velocity components, and energy, respectively. Additional equations related to temperature and pressure are solved next, and the updated values reflect the new properties of the fluid, leading to the transition to the next iteration. After implementing the written algorithm in MATLAB and running 100 iterations, the MATLAB simulator is created. To validate the accuracy of this simulator, an analysis is performed in ANSYS Fluent using the same parameters.

The validation process is crucial as it ensures that the initial model is consistent with the desired results. Since the differences between the analysis results logged in the MATLAB simulation diary and those conducted in the commercial software ANSYS Fluent are acceptably small, the validation and evaluation process of the initial model is considered to be successfully completed.

Subsequently, the initial model is transferred to MATLAB HDL Coder for the generation of HDL code. The initial model undergoes a detailing phase where it is broken down into atomic operations such as addition, subtraction, division, multiplication, and comparison, facilitating the creation of synthesizable HDL code suitable for FPGA implementation. At this stage, the discretized equations for a single finite volume are written line by line. After the codes are written to include only a single mathematical operation per line, they are transferred to the MATLAB HDL Coder tool. The 'xczu7cg-ffvf1517-2L-e' device from the Zynq UltraScale+ MPSoCs

family is selected for the synthesis process. The evaluation of synthesis results conducted in the Vivado environment and generated by MATLAB HDLcoder provides an in-depth overview of FPGA usage. This analysis includes a detailed breakdown of essential resources such as Look-Up Tables (LUTs), RAM, Flip-Flops, Digital Signal Processors (DSPs), and Buffers, which are crucial for performing calculations of continuity, momentum, energy, temperature, and pressure within the FPGA.

Remarkably, the iteration time recorded for solving the Euler Equations in a single cell through FPGA-based design is 606.8 ns, demonstrating a significant acceleration compared to traditional CPU-based methods. This performance is 1.17 times faster than previous studies focused on one-dimensional Euler Equation solutions on CPUs and 3.18 times faster than two-dimensional solutions on CPUs. These results not only confirm the efficacy of the FPGA approach in increasing computational speed but also signify a major advancement in the field of CFD simulations. This thesis showcases the potential for significant improvements in simulation times by leveraging FPGA technology, offering a faster and more efficient route for aerodynamic analysis and design iterations.

Looking forward, the thesis identifies future research areas, including expanding the hardware architecture to support a larger number of cells within a single FPGA, enhancing discretization techniques, and converting the solver into a three-dimensional model. These initiatives represent a critical step towards handling more complex simulations and enhancing the utility of FPGA-based accelerators in computational fluid dynamics and beyond, promising to extend the scope and applicability of this cutting-edge technology. Through this comprehensive research, the thesis not only demonstrates the transformative potential of FPGA-based accelerators for CFD simulations but also contributes to a broader discussion on the use of advanced computational tools to overcome the challenges presented by high-speed aerodynamic studies.



EULER DENKLEMLERİ İÇİN SONLU HACİMLER YÖNTEMİ İLE FPGA TABANLI DONANIM HIZLANDIRICI

ÖZET

Bu tezde, Alan Programlanabilir Kapı Dizilerinin (FPGA) yenilikçi kullanımı yoluyla Hesaplamalı Akışkanlar Dinamiği (CFD) simülasyonlarının hızlandırılmasına ilişkin ayrıntılı bir araştırma sunulmaktadır. Bu araştırma, akış analizinde geleneksel teorik ve deneysel yaklaşımların getirdiği sınırlamaları aşma, analizleri hızlandırma ve maliyeti düşürme ihtiyacıyla ortaya çıkmıştır.

Teorik modeller akışkanlar dinamiğini anlamada önemli adımlar atmıştır, ancak Navier-Stokes denklemleri gibi zorluğuyla bilinen ve "Milenyum Ödülü Problemleri" arasında listelenen karmaşık denklemleri çözmenin hesaplama zorluklarıyla karşılaştıklarında genellikle bocalarlar. Öte yandan deneysel yöntemler, rüzgar tüneli deneyleri aracılığıyla gerçek dünyaya ilişkin değerli bilgiler sağlar, ancak yüksek işletme maliyetleri, test koşulları üzerindeki sınırlı kontrol ve hassas ölçümler elde etmenin zorluğu nedeniyle kısıtlanırlar.

CFD, akışkanların dinamiğini incelemek için daha hassas ve uygun maliyetli bir metodolojiyi savunan üçüncü ve umut verici bir yaklaşım olarak öne çıkmaktadır. Akışkan hareketini simüle etmek için sayısal yöntemler ve algoritmalar kullanan CFD, karmaşık geometrilere uygulanabilirliği, fiziksel deneyleri etkili bir şekilde tamamlama kapasitesi ve önemli ölçüde maliyet ve zaman azaltma potansiyeli gibi belirgin avantajlar sergiler. Bununla birlikte, CFD simülasyonlarının güvenilirliği büyük ölçüde modelin kalitesine ve seçilen çözüm stratejilerine bağlıdır. Karmaşık geometrileri ve fiziksel olayları doğru bir şekilde modellemek için kapsamlı hesaplama gücüne ihtiyaç duyar.

CFD çözümleri daha önce yapılan çalışmalarda Merkezi İşlem Birim (CPU), Grafik İşlem Birimi (GPU) ve FPGA gibi donanımların yardımıyla önemli ölçüde hızlandırılmıştır. CPU'lar genel amaçlı hesaplamalar için tasarlanırken, GPU'lar özellikle yüksek paralellik gerektiren görevlerde avantajlar sunar. FPGA'lar, kullanıcılar tarafından belli görevleri yerine getirmek üzere yapılandırılabilen programlanabilir mantık çipleridir. FPGA'lar, donanımın işlevselliğinin yazılım aracılığıyla değiştirilmesine izin vererek onları çeşitli uygulamalar için esnek ve yeniden programlanabilmesine imkan sağlar. Bu çalışma, özelleştirilebilir yapıları sayesinde özel görevler için optimize edilebildiğinden CFD simülasyonlarının zorlu hesaplama gereksinimlerini karşılamak amacıyla, yeniden programlanabilir yapıları ve üstün performans yetenekleri ile tanınan FPGA'ların kullanımını önermektedir.

Bu çalışma için Model Tabanlı Tasarım (MBD) yaklaşımı benimsenerek, HDL kodlayıcı modülü aracılığıyla MATLAB komut dosyalarından sentezlenebilir Donanım Tanımlama Dili (HDL) kodu üretme süreci detaylandırılmıştır. Bu metodoloji

sadece tasarım sürecini iyileştirmeyi değil, aynı zamanda FPGA programlamanın güvenilirliğini artırmayı ve özellikle CFD çözümlerini hızlandırmak için tasarlanmış FPGA tabanlı bir donanım hızlandırıcısının geliştirilmesi için sağlam bir çerçeve oluşturmayı amaçlamaktadır. Araştırma özellikle, yüksek hızlı akışkan akışının modellenmesi için çok önemli olan Euler Denklemleri için bir çözücü geliştirmeye odaklanmaktadır. Tasarım bu metodolojiye göre gerçekleştirildikten sonra doğruluğu Ansys Fluent ticari yazılımı ile kontrol edilecektir. Doğruluğu kesinleşen ilk model detaylandırılacak ve HDL kodu üretilmesi için MATLAB HDL Coder aracına aktararak HDL kodu üretilcektir. HDL kod üretildikten sonra bir sonlu hacim elemanı için FPGA kaynak kullanımına ve çözüm süresine bakılması planlanmıştır. Böylelikle önceki çalışmalar ile tasarlanan donanım hızlandırıcısı arasındaki çözüm süresi performansı değerlendirilecektir.

Tezde öncelikle donanım tasarım metodolojisinden bahsedilmiş ve önerilen tasarım metodu hakkında bilgi verilmiştir. Bu tasarım metodolojisi, Sonlu Hacimler Yöntemi (FVM) ve ikinci dereceden yukarı rüzgar yöntemi ile elde edilen Euler Denklemlerinin ayrıştırılmasıyla başlar. Viskoz olmayan, zamana bağlı olarak değişmeyen ve sıkıştırılabilir akış koşullarının modellenmesi için temel olan bu denklemler daha sonra MATLAB komut dosyası formuna dönüştürülür. Bu süreç, bu tez için geliştirilen MATLAB simülatörünün omurgasını oluşturan çekirdek kod dosyalarının oluşturulmasıyla sonuçlanır. Simülatör, ilk modelleme aşamasının doruk noktasını temsil eder ve ayrıştırılmış denklemleri yinelemeli olarak çözerek tüm simülasyon sürecini kapsar.

Simülasyon algoritması, 100 hücrelik bir hesaplama alanında CFD analizi yapmak için gereken tüm adımları ayrıntılı bir şekilde kapsamaktadır. Hesaplama, çeşitli akış özellikleri için başlangıç koşullarının belirlenmesiyle başlar. Bu özellikler arasında hız bileşenleri, enerji, sıcaklık ve basınç yer alır ve simülasyonun temelini oluşturur. İlk yoğunluk hesaplaması, ideal gaz denklemi kullanılarak yapılır. Algoritma akışkanın yoğunluğunu, hız bileşenlerini ve enerjisini güncellemek için sırasıyla Süreklilik, Momentum ve Enerji denklemlerini paralel olarak çözer. Ardından sıcaklık ve basınçla ilgili ek denklemler çözülür ve edilen değerler akışkanın yeni özelliklerini yansıtacak şekilde güncellenir ve bir sonraki iterasyona geçiş yapılır. Yazılan algoritmanın MATLAB ortamında uygulanması ve 100 iterasyon koşturulmasından sonra MATLAB simülatörü oluşturulmuş olur. Bu simülatörün doğruluğunu karşılaştırma yoluyla doğrulamak için aynı parametreler kullanılarak ANSYS Fluent'te bir analiz gerçekleştirilir.

Doğrulama süreci, ilk modelin istenen sonuçlarla uyumlu olmasını sağladığı için çok önemlidir. MATLAB simülasyon günlüğünde tutulan analiz sonuçları ile Ansys Fluent ticari yazılımında yapılan analiz sonuçları arasındaki farklar kabul edilebilir ölçüde küçük oldukları için ilk modelin doğrulama ve değerlendirme süreci başarıyla tamamlanmış olarak kabul edilmiştir.

Daha sonra İlk model HDL kodunun oluşturulması için MATLAB HDL Coder'a aktarılır. İlk model toplama, çıkarma, bölme, çarpma ve karşılaştırma gibi atomik işlemlere ayrıldığı ve FPGA uygulaması için uygun sentezlenebilir HDL kodunun oluşturulmasını kolaylaştıran bir detaylandırma aşamasından geçer. Bu adımda tek bir sonlu hacim için tüm ayrıştırılmış denklemler satır satır yazılır.

Kodlar her satırda tek bir matematiksel işlem olacak şekilde yazıldıktan sonra, MATLAB HDL Coder aracına aktarılır. Sentez işlemi için Zynq UltraScale+ MPSoCs yonga ailesinden 'xczu7cg-ffvf1517-2L-e' cihazı seçilmiştir. MATLAB HDLcoder tarafından oluşturulan ve Vivado ortamında gerçekleştirilen sentez sonuçlarının değerlendirilmesi, FPGA kullanımına ilişkin derinlemesine bir genel bakış sunmaktadır. Bu analiz, FPGA içinde süreklilik, momentum, enerji, sıcaklık ve basınç hesaplamalarını gerçekleştirmek için çok önemli olan Look-Up Tabloları (LUT'lar), RAM, Flip-Flop'lar, Dijital Sinyal İşlemcileri (DSP'ler) ve Tampon Hafıza Bölgesi (Buffer'lar) gibi temel kaynakların ayrıntılı bir dökümünü kapsar.

Dikkat çekici bir şekilde, FPGA tabanlı tasarım aracılığıyla Euler Denklemlerini tek bir hücrede çözmek için kaydedilen yineleme süresi 606,8 ns'dir ve geleneksel CPU tabanlı yöntemlere kıyasla önemli bir hızlanma sergilemektedir. Bu performans, CPU'larda tek boyutlu Euler denklemi çözümlerine odaklanan önceki çalışmalardan 1,17 kat, CPU'larda iki boyutlu çözümlerden ise 3,18 kat daha hızlıdır. Bu sonuçlar FPGA yaklaşımının hesaplama hızını artırmadaki etkinliğini teyit etmekle kalmayıp, aynı zamanda CFD simülasyonları alanında önemli bir ilerlemeye işaret eder. Bu tez, FPGA teknolojisinden yararlanarak simülasyon sürelerinde belirgin iyileşme potansiyelini ortaya koymakta ve aerodinamik analiz ve tasarım yinelemeleri için daha hızlı, daha verimli bir yol sunmaktadır.

Bu araştırma, tek bir FPGA içinde daha fazla sayıda hücreyi desteklemek için donanım mimarisinin genişletilmesini, ayrıklaştırma tekniklerinin iyileştirilmesi ve çözücünün üç boyutlu bir modele dönüştürülmesini dahil olmak üzere gelecekteki bir çok araştırmaya ışık tutmaktadır. Bu girişimler, daha karmaşık simülasyonları ele alma ve FPGA tabanlı hızlandırıcıların hesaplamalı akışkanlar dinamiği ve ötesindeki faydasını artırma yönünde kritik bir adım anlamına gelerek, bu alandaki son teknolojinin kapsamını ve uygulanabilirliğini genişletmeyi vaat etmektedir. Bu kapsamlı araştırma sayesinde tez, FPGA tabanlı hızlandırıcıların CFD simülasyonlarını dönüştürme potansiyelini sergilemekle kalmaz, aynı zamanda yüksek hızlı aerodinamik çalışmaların sunduğu zorlukların üstesinden gelmek için gelişmiş hesaplama araçlarının kullanılmasına ilişkin daha geniş bir tartışmaya da katkıda bulunur.



1. INTRODUCTION

Researchers have demonstrated an intense curiosity in understanding the flow around aircraft and utilizing this knowledge to enhance flight since the earliest flights. Particularly after hypersonic flights towards the end of the 20th century, this interest has further increased. At that time, two approaches were employed to examine the flow: theoretical and experimental [1]. The theoretical approach relies on the mathematical modeling of equations necessary for understanding flow behavior, such as the Navier-Stokes equations. Despite significant theoretical advancements, the analytic solution of these equations is complex and difficult to solve directly. For instance, the Navier-Stokes equations are considered among the most challenging equations in mathematics today and are listed among the "Millennium Prize Problems" [2]. When reaching the limits of the theoretical approach, experimental studies become critically important. The experimental approach relies on collecting real-world data through wind tunnel experiments to study the flow. Experimental data is crucial for validating and refining CFD models, but faces obstacles such as high costs, limited control over parameters, and measurement precision. Especially towards the end of the 20th century, there were no wind tunnels capable of testing at both high speeds and high temperatures simultaneously. As a result of these two approaches, a third approach, CFD, is proposed to study flow more accurately and within broader limits at a lower cost. CFD uses numerical methods and algorithms to study the motion and interactions of fluids. A problem that cannot be solved analytically can be solved numerically, and a problem difficult to test in wind tunnels can be simulated with appropriate boundary conditions. It has advantages such as applicability to complex geometries and boundary conditions, the ability to replace or complement physical experiments, and savings in cost and time. However, the accuracy of the results depends on the quality of the model and solution strategy used. Moreover, solving such complex problems, accurately modeling detailed geometries, and physical phenomena require a high amount of computational power [3]. For example, assume a simple geometry

analyzing external flow is divided into 100,000 equal parts within a two-dimensional flow field. Assuming that continuity, energy, momentum, and turbulence equations are calculated for each element and that the computation converges in 1000 iterations, at least 400 million differential equations must be solved for this analysis. Even in a simple example, calculating so many differential equations requires much more computation for complex geometries, under different conditions (e.g., supersonic or transonic flows).

The process of solving these partial differential equations, which play a fundamental role in science and engineering, has been significantly accelerated with the help of hardware such as CPUs, GPUs, and FPGAs [4,5]. The CPU acts as the brain of the computer, processing program instructions and executing the computer's basic computation and control functions. This enables computers to perform various tasks and respond to the user. GPUs, on the other hand, are optimized specifically for graphics and image processing tasks. Unlike CPUs, GPUs possess thousands of smaller processing units, affording them very high parallel processing capabilities. This feature makes them highly effective for tasks requiring substantial computational power. While CPUs are designed for general-purpose computations, GPUs offer advantages particularly in tasks that demand high parallelism. FPGAs are programmable logic chips that can be configured by users to perform specific tasks. FPGAs allow for the functionality of the hardware to be altered through software, making them flexible and reprogrammable for various applications. FPGAs are often preferred in specialized operations requiring high performance, signal processing, cryptography, and artificial intelligence projects because their customizable structure can be optimized for such tasks.

FPGAs are hardware that can be challenging to program. Here, utilizing the MBD approach can offer a more straightforward solution. According to the methodology of model-based design, the design's main scripts are written in the MATLAB environment, and hardware definition codes are automatically generated using the HDL coder module. The design process with MBD becomes automated and standardized. This reduces the potential for errors that could be made while manually coding and ensures the design is more reliable [6].

In contrast to traditional solution methods [7,8], this thesis designs an alternative hardware accelerator to speed up CFD analysis [9]. A FPGA-based hardware accelerator has been developed due to their re-programmable and configurable nature and their use in applications requiring high performance. Also MBD was used during this design process.

1.1 Motivation

Theoretical approaches used in flow analysis are not applicable for every problem. In the experimental approach, wind tunnel tests are time-consuming and costly. Accelerated CFD simulations significantly speed up these processes, enabling engineers and researchers to innovate faster and perform design iterations. This presents a substantial advantage, especially for industries aiming to respond to rapid market changes and gain a competitive edge.

The design of FPGA-based hardware accelerators offers significant innovation for CFD solutions. The motivation for this technology stems from the intensive computational demands of CFD simulations. Traditional CPU and GPU-based systems may encounter limitations in terms of time and energy when solving large-scale or complex CFD problems. FPGA-based accelerators provide a significant performance increase in these computations, offering faster simulation processes and energy efficiency. This has the potential to revolutionize engineering design and analysis processes in industries such as aviation, automotive, and energy. The programmable nature of FPGAs allows for the optimization of specific CFD algorithms, facilitating the development of customized solutions and easing the standardization of CFD solution processes in general. This technology is a valuable tool for engineers as it accelerates simulation processes and saves time.

Additionally, MBD reduces manual coding errors and increases the overall reliability of the design while implementing FPGA designs. Given the complexity of FPGA-based systems, this enables more effective implementation of CFD algorithms. The integration of MBD simplifies the customization and optimization of FPGAs, allowing CFD simulations to be carried out more quickly and accurately. This

approach provides time and resource savings in engineering and research projects, paving the way for more innovative and effective solutions.

1.2 Aim of This Thesis

The aim of this thesis is to design an FPGA-based hardware accelerator to accelerate CFD solutions and develop a solver for Euler Equations for high velocity flow simulations [9]. The research will utilize a model-based hardware design methodology to validate the FPGA solution in terms of performance, accuracy, and power consumption against reference CFD tools (such as the commercial CFD analysis software ANSYS Fluent). HDL code generation will be carried out via MATLAB HDL-Coder. The purpose of this work is to assess the effectiveness of the FPGA implementation as a potential alternative to traditional CFD tools in terms of computational performance, solution accuracy, and energy efficiency.

1.3 Accelerating CFD

1.3.1 FPGA and MBD Approaches

CFD utilizes numerical methods and computer algorithms to simulate the movements of fluids and gases. There are several key reasons for preferring CFD over purely experimental and theoretical approaches. Firstly, CFD is a cost-effective solution. Physical experiments and wind tunnel tests, especially in large-scale projects or under challenging conditions, can be expensive. CFD can offer significant cost savings for such projects. Secondly, CFD is time-efficient. While preparing and conducting physical experiments takes time, CFD simulations allow for much quicker results. This accelerates the development of projects. Thirdly, CFD facilitates modifications across different physical conditions, boundary conditions, and geometries. This flexibility allows engineers and researchers to quickly test various scenarios. Fourthly, CFD offers the possibility of detailed analysis. It provides access to hard-to-obtain information such as detailed examination of parameters like pressure, temperature, and velocity in the flow field, which might be challenging to obtain through wind tunnels or theoretical analyses. Lastly, CFD enables the modeling of complex geometries and

boundary conditions, offering a significant advantage for systems involving complex surface geometries or intense physical interactions.

CFD simulations typically require a large number of computations, heavily utilizing computer resources. Acceleration processes enable these computations to be performed more quickly and efficiently. Therefore, there is a special interest in reducing the computation time for CFD problems. Particularly, intense research on hardware-specific solutions has shown that GPUs and FPGAs can solve problems faster than conventional CPUs found in ordinary computers [10]–[16].

When comparing GPUs with FPGAs, it is notable that FPGAs offer significant performance and energy efficiency advantages for specific applications due to their customizable hardware structure. FPGAs, being programmable according to specific processing requirements, can maximize processing power by providing workload-specific optimizations. This means higher efficiency and lower latency times for data-intensive and parallelism-required applications compared to GPUs. Additionally, the energy consumption of FPGAs is generally lower than that of GPUs, which offer extensive parallel processing capabilities. This makes FPGAs ideal for energy-constrained environments such as mobile devices and edge computing. These features make FPGAs superior for certain computational tasks compared to GPUs, underscoring the importance of including studies related to FPGAs within the scope of this work.

The challenges of FPGA design and development include a series of complex engineering and design challenges despite the flexibility and high-performance potential of these platforms. Overcoming these challenges with the use of the MBD approach can simplify and accelerate the design process.

One of the principal challenges of FPGA designs is the complexity of the hardware and the difficulty of programming. FPGAs contain a multitude of configurable logic blocks, I/O pins, and interconnections. This means that designers must understand how to best configure and combine logic blocks to create efficient and effective designs. Furthermore, FPGA programming differs from traditional software development languages and often involves the use of hardware description languages such as

VHDL or Verilog. These languages can be more challenging compared to high-level programming languages and may require a longer learning curve.

A second challenge involves the optimization of system performance. FPGA-based systems are often preferred for applications requiring high performance. However, maximizing system performance necessitates attention to factors such as data bus bandwidth, memory access, and the efficient use of logic blocks. This requires making complex design decisions and continuous optimization. A third challenge is testing and verification. FPGA designs require comprehensive testing and verification at every stage of the design. This is critical to ensure that both the hardware and the software running on it function correctly. However, the complexity of FPGA systems can make these testing and verification processes time-consuming and challenging.

The MBD approach offers an effective way to overcome these challenges. MBD is a development method where models are used to simulate, analyze, and design complex systems. It allows engineers to create virtual representations of systems, test them through simulations, and automatically generate code for implementation, making the design process faster, more efficient, and more accurate. MBD involves the use of mathematical models and simulations in the design process. This allows designers to analyze system performance and identify potential issues before conducting physical tests on the actual hardware. Moreover, MBD facilitates a better understanding and optimization of different aspects of the design, thereby speeding up the design process and reducing costs. The use of MBD makes the design and development of FPGA-based systems more accessible and manageable, enabling designers to fully leverage the advantages of this powerful technology.

In this thesis, the focus is the design of an FPGA-based hardware accelerator for CFD problems, embracing a MBD methodology. By forging a solver for the Euler Equations on FPGA, this research aims to expedite high-speed aerodynamic simulations. The thesis extends the MBD paradigm, leveraging the robust, script-driven capabilities of MATLAB to generate HDL code, a strategy reflected in contemporary studies cited herein [17]–[19]. The employment of direct MATLAB scripting rather than traditional Simulink models illustrates a pragmatic and innovative

application of MBD, tailored to streamline the hardware design process. This approach spotlights the adaptive nature of MBD, showcasing its applicability through MATLAB scripting which is pivotal for performance evaluation, accuracy benchmarking, and power consumption analysis of FPGA-based implementations when contrasted with conventional CFD tools.

Due to the priorities set and the objectives of the thesis, FPGA designs made with MBD will be shared after summarizing the development and successes of FPGA-based CFD accelerators in the literature review.

1.3.2 Literature Review

In 2011, Talip et al. [20] focused on solving Euler's equations on FPGA, demonstrating that single-precision designs operate at higher frequencies, yet double-precision designs provide better accuracy. This finding underscores the importance of accuracy in CFD simulations. In the same year, Roman and his team [21] reported achieving up to 13.25 times acceleration with the implementation of a finite volume technique based on cell diagonals for solving Euler's equations governing inviscid flows on FPGA. These studies showcase the potential of FPGA-based accelerators in CFD applications.

Gan and colleagues [22] introduced a hybrid CPU-DFE (Data Flow Engine) design for solving three-dimensional Euler atmospheric equations on FPGA. This study highlights the significant performance enhancements provided by FPGA technology in solving complex atmospheric equations, with the FPGA-based design operating 18.5 times faster and 8.3 times more energy-efficiently compared to traditional CPU-based systems.

Ebrahimi and Zandsalimi [23] investigated the use of FPGA chips to accelerate the numerical solution of fluid dynamics equations. They applied typical Computational Fluid Dynamics (CFD) problems such as Laplace and 1-D Euler equations on both reconfigurable hardware and CPU, numerically solving them. The computation process on the FPGA was shown to be 20 times faster than that on a traditional CPU with the same data precision.

Rojek [11] adapted four different CFD cores to the Xilinx ALVEO U250 FPGA, stating that this adaptation provided up to 40% improvement in execution time and up to 80% reduction in energy consumption compared to CPU processors.

Kamalakkannan et al. [24] demonstrated that FPGA-based solutions offer significant advantages in applications requiring high performance, including CFD applications. Three representative applications were implemented on FPGA with nearly optimal performance, achieving over 85% model prediction accuracy. These findings suggest that FPGA-based solutions possess significant potential in high-performance applications, offering notable advantages in energy efficiency and performance compared to traditional CPU and GPU architectures.

Tomczak et al. [25] presented a hardware accelerator based on the modal discontinuous Galerkin method for the complete solution of Navier-Stokes equations. A hardware design containing simple and concurrently operating micro-programmable computing units was developed to achieve high performance and reduce energy consumption. This design was adapted to a mid-scale FPGA and provided reasonable hardware usage with performance exceeding 100,000 network elements per second in all tested cases. The results indicate that FPGA-based solutions could offer significant benefits in accelerating CFD codes in terms of energy efficiency and performance.

Memis's [19] thesis study thoroughly examines how the MBD approach can be used to design and implement complex systems like a quadrotor dynamic model on FPGA. The proposed refined model-based FPGA hardware design methodology integrates MBD with high-level synthesis (HLS) methods, incorporating concepts of Resource Sharing and Standard Processor Architecture. This integration ensures more efficient use of FPGA resources and enables the realization of complex dynamic models on hardware with high accuracy and low resource usage.

In a study by Şenel and Yeniçeri [18], an MBD hardware accelerator for converting quaternions to Euler angles was designed. This work introduces a design methodology that involves automatic code generation from a Simulink model using the HLS method, detailing the formulation of numerical representation, basic operators, and functions.

1.4 Thesis Outline

The following chapters of this thesis present a structured and detailed review of the model-based hardware design process. Chapter 2 provides a detailed description of the model-based hardware design methodology. This chapter summarizes the steps involved in the formulation of a hardware design methodology aimed at creating an FPGA-based hardware accelerator specific to the solver of the Euler Equations.

Advancing to Chapter 3, the discussion pivots to the foundational aspects of initial modeling where the essential equations and solution techniques required for CFD are presented in detail. This chapter aims to provide a clear understanding of how these mathematical models are derived, justified, and initially applied, setting a solid foundation for the advanced modeling and simulations to follow.

This thesis moves towards a rigorous comparative analysis in Chapter 4. This chapter is dedicated to evaluating the performance results obtained from ANSYS Fluent with those obtained from MATLAB scripts. Since the evaluation in this thesis is done by comparing ANSYS Fluent and MATLAB simulation results, the same initial model simulated in MATLAB is created and validated in ANSYS Fluent.

Chapter 5 delves into the outcomes of applying model elaboration formulated from discretized equations. This chapter showcases the comprehensive steps from mathematical modeling and simulation in MATLAB through to HDL code generation and FPGA synthesis, underlining the precision and efficiency of the model elaboration process.

This thesis culminates in Chapter 6 with a comprehensive discussion and a forward-looking outlook for future work. This final chapter synthesizes the collective findings and translates them into a coherent discussion that not only reflects the current work but also looks at potential improvements and areas for future research. This chapter aims to articulate the broader implications of the study and foster a dialog about the advances that can be pursued in the evolving field of hardware-accelerated computational fluid dynamics. The conclusion summarizes the essence of the research and offers insights and directions for future academic work.



2. HARDWARE DESIGN METHODOLOGY

This chapter examines the subtleties of hardware design methodology, laying the groundwork for understanding the proposed model-based hardware design approach and the steps meticulously planned for this thesis.

The methodology for hardware design pertains to a structured strategy applied in the conception and evolution of hardware systems, which include things like integrated circuits, computer parts, and various electronic apparatuses. This approach is generally segmented into multiple phases. In this chapter of the thesis, we will examine the model based design steps required to create an FPGA based hardware accelerator for solving the Euler Equations.

The Euler Equations are mathematical equations of critical importance in the aerodynamic study of objects moving at high speeds, including supersonic and hypersonic aircraft. The reason for choosing this set of equations and further explanations are discussed in more detail in Chapter 3 under "Mathematical Expressions" section.

Initially, the Euler Equations are derived for an inviscid, steady and compressible flow. Then, these equations are discretized over two dimensions using the FVM developed with a second order upwind approximation. These discretized equations represent the Final Equation set, which is the first output of the Idea of the proposed model-based hardware design shown in Figure 2.1.

MATLAB script development of the solution is carried out using this discretized final equation set. In this step, the discretized equations are written as MATLAB code and the codes written are called core script files. After this stage, all adjustments are made by modifying or improving on this core script files. A simulation wrapper is created using the core script files and a script file is created to contain an entire simulation to iteratively solve the resulting equations. All the scripts generated at the

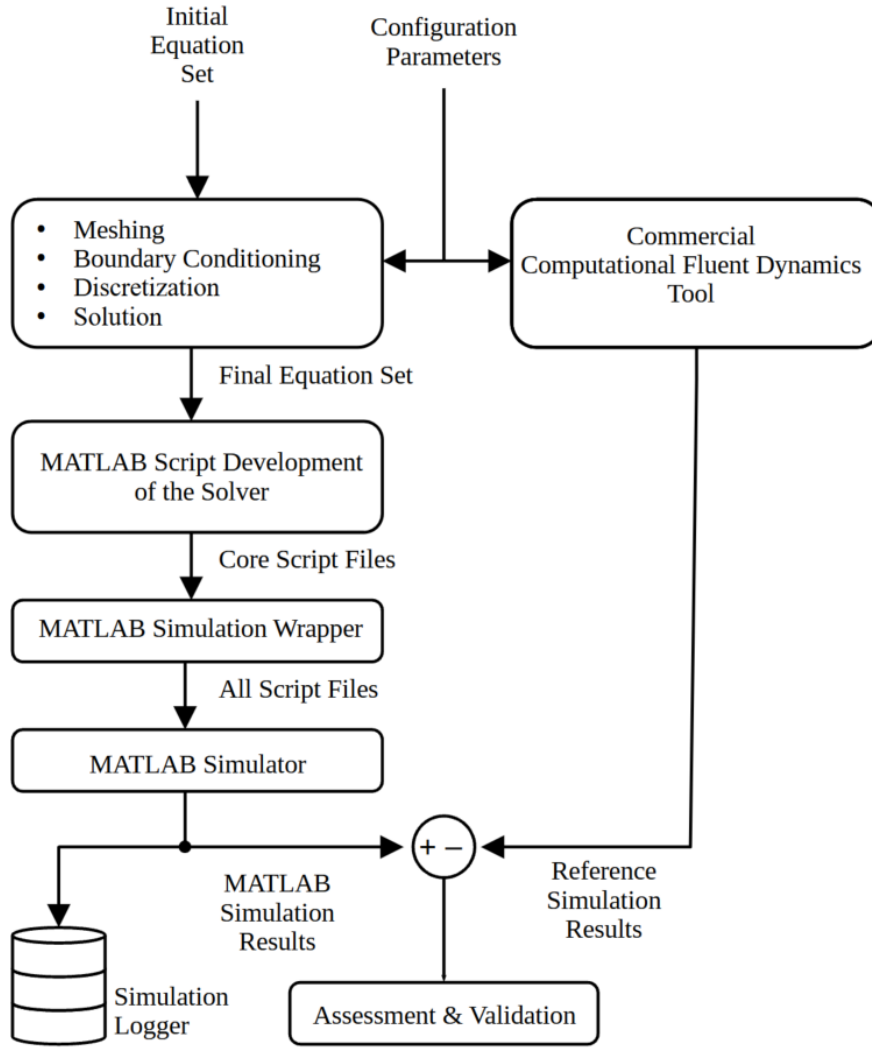


Figure 2.1 : Idia of the proposed model-based hardware design.

end of this process create the MATLAB Simulator developed for this thesis. Once the MATLAB Simulator is created, the analysis is performed in ANSYS Fluent using the same configuration parameters and the results are compared. After evaluation and assessment, if the results match the desired outcome, this is considered the initial model. The initial model is imported into MATLAB HDL Coder. Details on how this validation is done and whether acceptable results are obtained are given in Section 4.

The initial model was used to generate the first HDL code in MATLAB HDL Coder. In the next step, the model was refined to be transformed into atomic functions such as addition, subtraction, division, multiplication and comparison. Finally, the

synthesizable HDL code for the FPGA was generated from the MATLAB code. Throughout these steps, the newly generated HDL code needs to be validated and evaluated with the results stored in the simulation log (in Figure 2.2) obtained from the simulations performed in MATLAB. If the verification of the generated HDL code is positive, the implementation will be realized. The flow chart of the model-based hardware design methodology, the steps of which are explained, is given in Figure 2.2 [6].

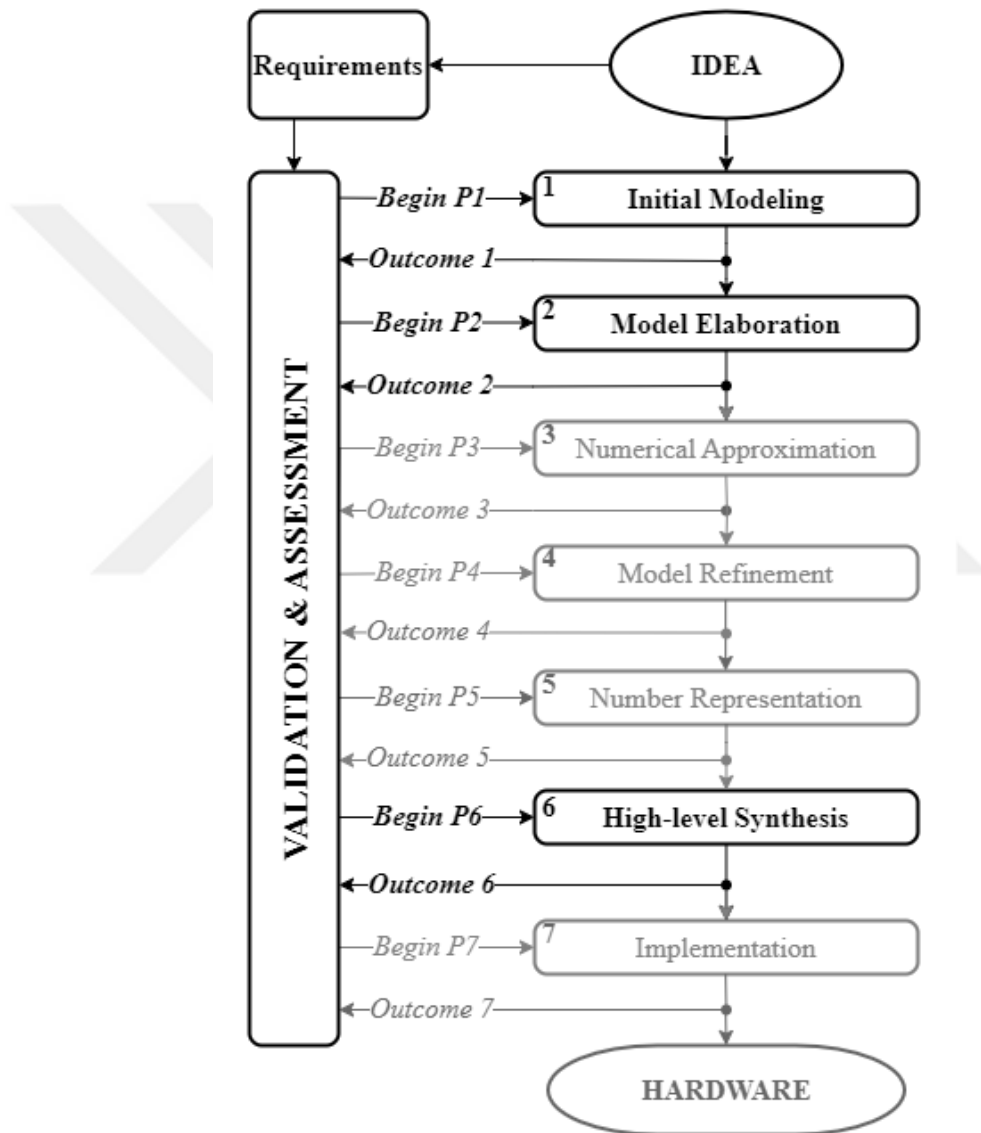


Figure 2.2 : Flowchart of the model-based hardware design methodology.

In the HDL coder framework, the hardware design is performed exactly for a single finite volume. The required data from adjacent volumes (interconnected nodes) is obtained from the record kept in the simulation log during each iterative cycle. This single finite volume is used in the calculation for a cluster of 100 cells strategically positioned within a square region. This region is the product of a meticulously structured network consisting of a 10 by 10 array of volumes.

As shown in Figure 2.2, the foundation of this hardware design approach is based on a conceptual step. Following this, an initial model is created by taking cues from the initial concept, which is then imported into the MATLAB HDL Coder. This prototype is subjected to a rigorous verification and validation process, utilizing information from the simulation logging module, before progress is confirmed. Then, in the model refinement phase, the prototype is refined into a design consisting of basic elements (such as multipliers, dividers, adders and subtractors). The output of this polished model is once again subjected to a thorough review and validation using data from the simulation log. Once this review phase is complete, the procedure moves on to the next phase. The HLS step requires the analysis, evaluation and assessment of each of the predecessor stages in turn and results in the generation of synthesizable HDL code suitable for the FPGA. This code generation step clears the way for the transformation of the conceptual model into a hardware description ready for synthesis and deployment on the FPGA architecture.

3. INITIAL MODELLING

In the initial modeling, the method and scheme for solving the Euler Equations on a hardware accelerator are first outlined. Following this, the inviscid, steady and compressible Euler Equations are introduced and then discretized for computation. This chapter is of great importance for the understanding and background of the flow problem to be solved. By providing insights into the theoretical and practical aspects that underpin the research, this chapter provides a crucial foundation for grasping the intricacies of the fluid dynamics in question. This foundational knowledge is essential for navigating the subsequent analysis and interpretations and enriches the overall thesis with a deeper understanding of the nature of the flow problem and its importance in the field of CFD.

3.1 Finite Volume Method

FVM is a powerful numerical technique widely used in CFD to simulate and analyze how fluid flows behave within various domains. Unlike other numerical methods that might focus on the differential form of the governing equations, FVM is particularly adept at conserving fluxes through a control volume. This makes it exceptionally suitable for simulations where the conservation of mass, momentum, and energy is critical.

At its core, FVM discretizes the space into a multitude of small, finite volumes or cells. These cells cover the entire computational domain, ensuring that the analysis encompasses all spatial aspects of the flow. For each of these cells, the fluxes entering and exiting the control volume are precisely calculated, and the governing physical laws (such as the Navier-Stokes equations or Euler Equations) are applied in their integral form

One of the distinctive features of FVM is the way it handles information and calculations. The method gathers and processes data at the centers of the cells, while

the interfaces between these cells (the cell faces) serve as the critical points for flux computations. The vertices of the cells, known as nodes, play a pivotal role in defining the geometry of the control volume but do not directly participate in the primary FVM calculations. Schematic representation of finite volume discretization as shown in Figure 3.1

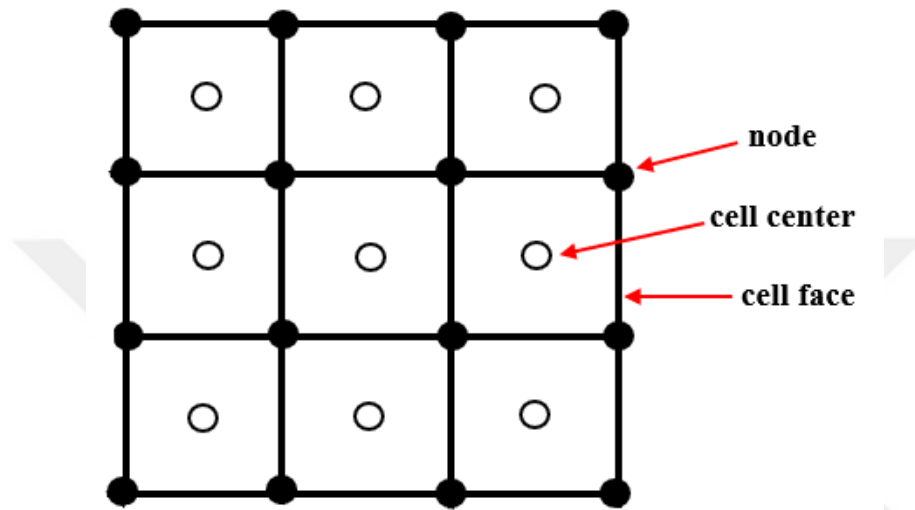


Figure 3.1 : Schematic representation of finite volume discretization.

This spatial arrangement and methodological approach allow FVM to ensure the conservation of physical quantities not just locally, within each small volume, but also globally across the entire computational domain. It's this rigorous adherence to conservation principles that sets FVM apart and makes it particularly valuable for solving real-world engineering and physical problems. The inherent characteristic of the FVM to ensure conservation has significantly contributed to its growing application in addressing practical physical challenges [26,27]. Its application spans across various fields, from aerodynamics and weather prediction to the design of complex engineering systems, where accurate simulation of fluid dynamics is essential.

3.2 Second-Order Upwind Scheme

The Second Order Upwind scheme represents a refined numerical approach designed to improve the accuracy of CFD simulations, especially in the context of solving

transport equations. This scheme is an evolution of the first-order upwind method and offers a higher degree of precision by incorporating gradient information from neighboring cells, thus significantly reducing the numerical diffusion usually associated with low-order methods.

Taking into account the fact that the information moves in the direction of the flow, the appropriate neighboring cell value is chosen to calculate the flux at the cell surface. The upwind scheme accurately captures the advection and allows for accurate calculation of sudden changes in velocity and pressure. The ability to produce more refined results in situations where flow fields exhibit sharp gradients or discontinuities, such as shock waves in compressible flows or sharp interfaces in multiphase flows. Leveraging additional information from the upstream and downstream conditions of the flow, the scheme skillfully minimizes the propagation of such features and thus provides a more physically accurate representation of flow phenomena. It allows for accurate observation of shock waves occurring at high velocities, as discussed in this thesis.

In this thesis, the second-order upwind method is utilized to enhance accuracy. With the first-order upwind method, terms are expanded only to the first level in the Taylor series, leading to increased errors. However, by expanding the Taylor series to the second level in the second-order upwind approach, a reduction in truncation error is achieved.

The second-order upwind scheme uses the information of the center cell and the neighboring cells [26]. Therefore, the terms $U_{i-1,j}$ and $U_{i,j}$ are expanded to Taylor series as given by,

$$U_{i-1,j} = U_e - \frac{3\Delta x}{2} \frac{dU}{dx} \Big|_e + \frac{1}{2} \left(\frac{3\Delta x}{2} \right)^2 \frac{d^2U}{dx^2} \Big|_e \quad (3.1)$$

where the U is any property of the flow, sub index i represents the x -directional coordinate and j represents the y -directional coordinate of the cell, Δx is the distance between the cell centers in x -direction. These expressions are as shown in Figure 3.2. Also, Δy is the distance between the cell centers in y -direction, e represents the east face of the cell, w represents the west face of the cell, n represents the north face of the

cell, s represents the south face of the cell, N is cell number in x -direction, M is cell number in y -direction.

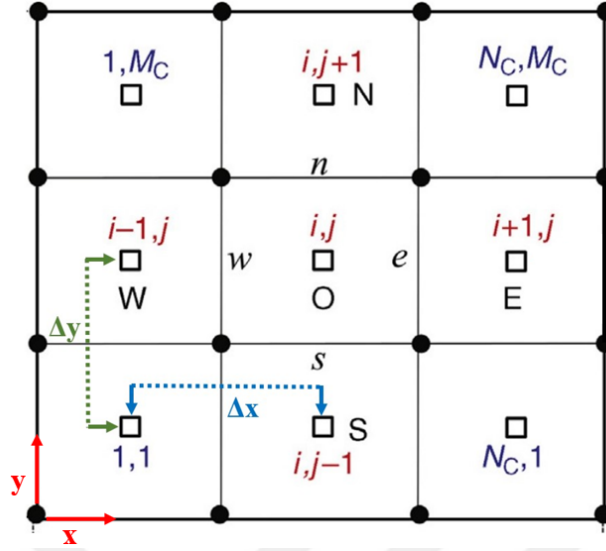


Figure 3.2 : Schematic representation of second-order upwind scheme.

So $U_{i-1,j}$ is a property of the western cell and U_e is the east face of the cell as seen in Figure 3.2. $i-1,j$ is to left of east face by $\frac{3}{2}\Delta x$.

$$U_{i,j} = U_e - \frac{\Delta x}{2} \frac{du}{dx} \Big|_e + \frac{1}{2} \left(\frac{\Delta x}{2} \right)^2 \frac{d^2u}{dx^2} \Big|_e \quad (3.2)$$

The following equation is obtained when equation 3.2 is multiplied by 3 and subtracted from equation 3.1,

$$U_e = \frac{3U_{i,j} - U_{i-1,j}}{2} + \frac{3}{8}(\Delta x)^2 \frac{d^2U}{dx^2} \Big|_e + \dots \quad (3.3)$$

A comparable set of formulas can also be developed for the velocity at the cell's west, north, and south faces. By performing integration of these formulas for the east, west, north, and south faces, and considering both the forward and backward directions of flow, the second-order upwind scheme can be articulated as equality sequence in 3.4. These equations have been derived only for the velocity u which is the velocity in the x -direction, and similar equations also apply to the velocity v that the velocity in the y -direction.

$$\begin{aligned}
U_{e,i,j} &= \begin{cases} \frac{3U_{i,j}-U_{i-1,j}}{2} & \text{if } \rho u|_{e,i,j} > 0 \\ \frac{3U_{i+1,j}-U_{i+2,j}}{2} & \text{if } \rho u|_{e,i,j} < 0 \end{cases}, \\
U_{n,i,j} &= \begin{cases} \frac{3U_{i,j}-U_{i,j-1}}{2} & \text{if } \rho u|_{n,i,j} > 0 \\ \frac{3U_{i,j+1}-U_{i,j+2}}{2} & \text{if } \rho u|_{n,i,j} < 0 \end{cases}, \\
U_{w,i,j} &= \begin{cases} \frac{3U_{i-1,j}-U_{i-2,j}}{2} & \text{if } \rho u|_{w,i,j} > 0 \\ \frac{3U_{i,j}-U_{i+1,j}}{2} & \text{if } \rho u|_{w,i,j} < 0 \end{cases}, \\
U_{s,i,j} &= \begin{cases} \frac{3U_{i,j-1}-U_{i,j-2}}{2} & \text{if } \rho u|_{s,i,j} > 0 \\ \frac{3U_{i,j}-U_{i,j+1}}{2} & \text{if } \rho u|_{s,i,j} < 0 \end{cases},
\end{aligned} \tag{3.4}$$

Here " ρu " indicates whether the air is in forward and backward directions. If this value is greater than zero, it is considered to be in the forward (positive) direction and if it is less than zero, it is considered to be in the backward (negative) direction. As seen in equation sets in 3.4 second-order upwind scheme should change by direction of the flow. In this thesis, the equations are discretized with the assumption that the flow directions u and v are positive.

3.3 Mathematical Expressions

The Euler Equations are the fundamental equations in fluid dynamics that describe the motion of ideal (frictionless and heat-exchange-free) fluids. These equations do not take into account the effects of internal friction and viscosity in the fluid and describe how the fundamental quantities of a fluid, such as velocity, pressure and density, vary in time and space. Euler Equations express how the acceleration of a fluid at any point changes under the influence of internal and external forces. In fields such as hydrodynamics and aerodynamics, Euler Equations are used to solve a wide range of problems, from simple fluid motions to complex flow regimes. Through these equations, velocity fields, pressure distributions and other important physical properties can be calculated efficiently for perfect gases and compressible fluids.

In this thesis, the terms 'inviscid', 'steady' and 'compressible' describe the properties of fluid flow. For a better understanding of the subject, these terms can be explained respectively. Viscosity is a property that expresses the internal frictional resistance of a fluid. This property measures the fluid's resistance to the rate of strain. In other words,

viscosity is the fluid's internal friction against flow. Inviscid is a term that refers to the state in which the viscosity (internal frictional resistance) of a fluid is zero. Steady means that a flow has a property that does not change with time. This means that the velocity, pressure, temperature and other physical properties of the flow at each point remain constant over time. That is, these values measured at any point do not change over time. Finally, the term compressible refers to the state in which the density of a fluid can change due to pressure changes. If the Mach number is greater than 0.3, the flow is considered compressible.

Since the Euler Equations do not include viscosity terms, they are particularly applicable in scenarios where viscosity effects are minimal, such as low air density and high velocity. This thesis is intended to focus on objects moving at high Mach numbers. Accordingly, the solution of the 2D Euler Equations for compressible flow (Mach number exceeding 0.3) will be considered and the flow will be assumed to be inviscid and steady. Essentially, the Euler Equations consist of three different sets of equations: the continuity equation, the momentum equations and the energy equations.

3.3.1 Continuity equation

The Continuity Equation expresses the conservation of mass of a fluid; that is, it states that there must be a balance between the mass of fluid entering a control volume and the mass of fluid leaving it. Mathematically, this is expressed for a 2D flow as in equation 3.5.

$$\frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} = 0 \quad (3.5)$$

Here, ρ represents the density of the fluid, while u and v are the velocity components of the fluid along the x - and y - axes of a 2D control volume, respectively. u , v and ρ are variables since the air is compressible (In compressible flows, density cannot be taken constant because it changes, velocities are already variable in incompressible flows). Therefore, Leibniz's rule is used to find the derivative given as equation 3.6.

$$\frac{\partial \rho}{\partial x} u + \frac{\partial u}{\partial x} \rho + \frac{\partial \rho}{\partial y} v + \frac{\partial v}{\partial y} \rho = 0 \quad (3.6)$$

Given Equation 3.6 is in a conservative form. Continuing the solution of the Continuity Equation, this expression is integrated as equation 3.7.

$$\int_s^n \int_w^e \frac{\partial \rho}{\partial x} u \, dx \, dy + \int_s^n \int_w^e \frac{\partial u}{\partial x} \rho \, dx \, dy + \int_w^e \int_s^n \frac{\partial \rho}{\partial y} v \, dy \, dx + \int_w^e \int_s^n \frac{\partial v}{\partial y} \rho \, dy \, dx = 0 \quad (3.7)$$

The relationship between the Cardinal directions and the x -, y - axes is illustrated in Figure axes relation can be shown in Figure 3.2. From West to East, increases. From South to North, increases.

If the integral boundaries are written in place, the equations in 3.8 and 3.9 are obtained respectively.

$$\int_s^n u(\rho_e - \rho_w) \, dy + \int_s^n \rho(u_e - u_w) \, dy + \int_w^e v(\rho_n - \rho_s) \, dx + \int_w^e \rho(v_n - v_s) \, dx = 0 \quad (3.8)$$

$$u(\rho_e - \rho_w) \Delta y + \rho(u_e - u_w) \Delta y + v(\rho_n - \rho_s) \Delta x + \rho(v_n - v_s) \Delta x = 0 \quad (3.9)$$

If equation 3.9 is written in the second order upwind approximation using equation (3.4) for the case where " ρu " is greater than zero (because in this thesis the flow direction is assumed to be positive), becomes the form in equation 3.10.

$$u_{i,j} \left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y + \rho_{i,j} \left(\frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \Delta y + v_{i,j} \left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x + \rho_{i,j} \left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \Delta x = 0 \quad (3.10)$$

If Equation 3.10 is rearranged slightly and the left-hand side is parenthesized $\rho_{i,j}$, equation 3.11 is obtained.

$$\begin{aligned} \rho_{i,j} \left(\left(\frac{6u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \Delta y + \left(\frac{6v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \Delta x \right) \\ = \left(\frac{4\rho_{i-1,j} - \rho_{i-2,j}}{2} \right) u_{i,j} \Delta y + \left(\frac{4\rho_{i,j-1} - \rho_{i,j-2}}{2} \right) v_{i,j} \Delta x \end{aligned} \quad (3.11)$$

Finally, when $\rho_{i,j}$ is left alone on the left-hand side of the equation, the spatially discretized Continuity Equation takes the form of equation 3.12.

$$\rho_{i,j} = \frac{\left(\frac{4\rho_{i-1,j}-\rho_{i-2,j}}{2}\right)u_{i,j}\Delta y + \left(\frac{4\rho_{i,j-1}-\rho_{i,j-2}}{2}\right)v_{i,j}\Delta x}{\left(\left(\frac{6u_{i,j}-4u_{i-1,j}+u_{i-2,j}}{2}\right)\Delta y + \left(\frac{6v_{i,j}-4v_{i,j-1}+v_{i,j-2}}{2}\right)\Delta x\right)} \quad (3.12)$$

3.3.2 Momentum equations

Momentum Equations are derived from Newton's second law of motion and express the variation of the momentum (amount of motion) of a fluid. These equations model how momentum, which is the product of the mass and velocity of each part of the fluid, changes under the influence of internal and external forces. Basically, they show how the velocity and pressure of the fluid change as a function of internal-external forces, viscosity and other influencing factors. Since viscous effects are not taken into account in this thesis, the 2D Momentum Equations for steady and compressible flow can be expressed as in equation 3.13 and 3.14 where p is pressure.

$$\frac{\partial \rho uu}{\partial x} + \frac{\partial \rho uv}{\partial y} = -\frac{\partial p}{\partial x} \quad (3.13)$$

$$\frac{\partial \rho uv}{\partial x} + \frac{\partial \rho vv}{\partial y} = -\frac{\partial p}{\partial y} \quad (3.14)$$

u , v and ρ are variables in the Momentum Equations like in the Continuity Equation. When the Leibniz rule is applied to equation 3.13, the resulting equations are as in 3.15-3.18.

$$\frac{\partial(\rho uu)}{\partial x} = u \left(u \frac{\partial \rho}{\partial x} + \rho \frac{\partial u}{\partial x} \right) + \rho u \frac{\partial u}{\partial x} \quad (3.15)$$

$$u \left(u \frac{\partial \rho}{\partial x} + \rho \frac{\partial u}{\partial x} \right) + \rho u \frac{\partial u}{\partial x} + u \left(v \frac{\partial \rho}{\partial y} + \rho \frac{\partial v}{\partial y} \right) + \rho v \frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} \quad (3.16)$$

$$\frac{\partial(\rho uv)}{\partial y} = u \left(v \frac{\partial \rho}{\partial y} + \rho \frac{\partial v}{\partial y} \right) + \rho v \frac{\partial u}{\partial y} \quad (3.17)$$

$$u^2 \frac{\partial \rho}{\partial x} + 2u\rho \frac{\partial u}{\partial x} + uv \frac{\partial \rho}{\partial y} + u\rho \frac{\partial v}{\partial y} + \rho v \frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} \quad (3.18)$$

Repeating the same process in equation 3.14 yields the result as in equations 3.19-3.22.

$$\frac{\partial(\rho uv)}{\partial x} = u \left(v \frac{\partial \rho}{\partial x} + \rho \frac{\partial v}{\partial x} \right) + \rho v \frac{\partial u}{\partial x} \quad (3.19)$$

$$\frac{\partial(\rho vv)}{\partial y} = v \left(v \frac{\partial \rho}{\partial y} + \rho \frac{\partial v}{\partial y} \right) + \rho v \frac{\partial v}{\partial y} \quad (3.20)$$

$$u \left(v \frac{\partial \rho}{\partial x} + \rho \frac{\partial v}{\partial x} \right) + \rho v \frac{\partial u}{\partial x} + v \left(v \frac{\partial \rho}{\partial y} + \rho \frac{\partial v}{\partial y} \right) + \rho v \frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} \quad (3.21)$$

$$uv \frac{\partial \rho}{\partial x} + u\rho \frac{\partial v}{\partial x} + \rho v \frac{\partial u}{\partial x} + v^2 \frac{\partial \rho}{\partial y} + 2v\rho \frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} \quad (3.22)$$

Equations 3.18 and 3.22 are in conservative form as with the continuity equation. These equations are integrated to discretization. If equation 3.18 are written in integral form, the equation becomes as in (3.23)

$$\begin{aligned} \int_s^n \int_w^e u^2 \frac{\partial \rho}{\partial x} dx dy + \int_s^n \int_w^e 2u\rho \frac{\partial u}{\partial x} dx dy + \int_w^e \int_s^n uv \frac{\partial \rho}{\partial y} dy dx + \\ \int_w^e \int_s^n u\rho \frac{\partial v}{\partial y} dy dx + \int_w^e \int_s^n \rho v \frac{\partial u}{\partial y} dy dx = - \int_s^n \int_w^e \frac{\partial p}{\partial x} dx dy \end{aligned} \quad (3.23)$$

Then, if the integral boundaries are set in place, equations 3.24 and 3.25 are obtained respectively.

$$\begin{aligned} \int_s^n u^2 (\rho_e - \rho_w) dy + \int_s^n 2u\rho (u_e - u_w) dy + \int_w^e uv (\rho_n - \rho_s) dx \\ + \int_w^e u\rho (v_n - v_s) dx + \int_w^e \rho v (u_n - u_s) dx = - \int_s^n (p_e - p_w) dy \end{aligned} \quad (3.24)$$

$$\begin{aligned} u^2 (\rho_e - \rho_w) \Delta y + 2u\rho (u_e - u_w) \Delta y + uv (\rho_n - \rho_s) \Delta x \\ + u\rho (v_n - v_s) \Delta x + \rho v (u_n - u_s) \Delta x = - (p_e - p_w) \Delta y \end{aligned} \quad (3.25)$$

When equation 3.22 is written in integral form, it becomes as in 3.26.

$$\begin{aligned} & \int_s^n \int_w^e uv \frac{\partial \rho}{\partial x} dx dy + \int_s^n \int_w^e u\rho \frac{\partial v}{\partial x} dx dy + \int_s^n \int_w^e \rho v \frac{\partial u}{\partial x} dx dy \\ & + \int_w^e \int_s^n v^2 \frac{\partial \rho}{\partial y} dy dx + \int_w^e \int_s^n 2v\rho \frac{\partial v}{\partial y} dy dx = - \int_w^e \int_s^n \frac{\partial p}{\partial y} dy dx \end{aligned} \quad (3.26)$$

When the integral boundaries are substituted, equations 3.27 and 3.28 are obtained, respectively.

$$\begin{aligned} & \int_s^n uv (\rho_e - \rho_w) dy + \int_s^n \rho u (v_e - v_w) dy + \int_s^n \rho v (u_e - u_w) dy \\ & + \int_w^e v^2 (\rho_n - \rho_s) dx + \int_w^e 2v\rho (v_n - v_s) dx = - \int_w^e (p_n - p_s) dx \end{aligned} \quad (3.27)$$

$$\begin{aligned} & uv (\rho_e - \rho_w) \Delta y + \rho u (v_e - v_w) \Delta y + \rho v (u_e - u_w) \Delta y \\ & + v^2 (\rho_n - \rho_s) \Delta x + 2v\rho (v_n - v_s) \Delta x = -(p_n - p_s) \Delta x \end{aligned} \quad (3.28)$$

Using the second-order upwind approximation as in the expression in 3.4, and reformulating equations 3.25 and 3.28, the spatially discretized forms of the Momentum Equations are as given in equations 3.29 and 3.30, respectively.

$$\begin{aligned} & u_{i,j}^2 \left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y + 2u_{i,j}\rho_{i,j} \left(\frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \Delta y + \\ & u_{i,j}v_{i,j} \left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x + u_{i,j}\rho_{i,j} \left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \Delta x \\ & + \rho_{i,j}v_{i,j} \left(\frac{3u_{i,j} - 4u_{i,j-1} + u_{i,j-2}}{2} \right) \Delta x = \frac{-3p_{i,j} + 4p_{i-1,j} - p_{i-2,j}}{2} \Delta y \end{aligned} \quad (3.29)$$

$$\begin{aligned} & u_{i,j}v_{i,j} \left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y + \rho_{i,j}u_{i,j} \left(\frac{3v_{i,j} - 4v_{i-1,j} + v_{i-2,j}}{2} \right) \Delta y \\ & + \rho_{i,j}v_{i,j} \left(\frac{3u_{i,j} - 4u_{i,j-1} + u_{i,j-2}}{2} \right) \Delta y + v_{i,j}^2 \left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x \\ & + 2v_{i,j}\rho_{i,j} \left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \Delta x = \frac{-3\rho_{i,j} + 4\rho_{i,j-1} - \rho_{i,j-2}}{2} \Delta x \end{aligned} \quad (3.30)$$

In order to find the velocity variable from the obtained discretized equations, some modifications need to be made in the equations. Using equation 3.29, the velocity in x -direction $u_{i,j}$ is found. For this the equation 3.29 is written as a quadratic equation which depends on $u_{i,j}$. First the equation is separated into $u_{i,j}^2$, $u_{i,j}$ and constants as seen in equation 3.31.

$$\begin{aligned}
& u_{i,j}^2 \left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y + 3u_{i,j}^2 \rho_{i,j} \Delta y + 2u_{i,j} \left(\left(\frac{-4u_{i-1,j} + u_{i-2,j}}{2} \right) \rho_{i,j} \Delta y \right) \\
& + u_{i,j} \left(\left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) v_{i,j} \Delta x \right) + u_{i,j} \left(\left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \rho_{i,j} \Delta x \right) \\
& + u_{i,j} \left(\frac{3\rho_{i,j} v_{i,j} \Delta x}{2} \right) + \left(\left(\frac{-4u_{i,j-1} + u_{i,j-2}}{2} \right) \rho_{i,j} v_{i,j} \Delta x \right) = \frac{-3p_{i,j} + 4p_{i-1,j} - p_{i-2,j}}{2} \Delta y
\end{aligned} \tag{3.31}$$

Then equation 3.31 is set equal to zero as in equation 3.32.

$$\begin{aligned}
& u_{i,j}^2 \left(\left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y + (3\rho_{i,j}) \Delta y \right) + u_{i,j} \left(\left((-4u_{i-1,j} + u_{i-2,j}) \rho_{i,j} \Delta y \right) \right. \\
& + \left(\left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) v_{i,j} \Delta x \right) + \left(\left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \rho_{i,j} \Delta x \right) \\
& \left. + \left(\frac{3\rho_{i,j} v_{i,j} \Delta x}{2} \right) - \left(\frac{-3p_{i,j} + 4p_{i-1,j} - p_{i-2,j}}{2} \Delta y \right) + \left(\left(\frac{-4u_{i,j-1} + u_{i,j-2}}{2} \right) \rho_{i,j} v_{i,j} \Delta x \right) = 0
\end{aligned} \tag{3.32}$$

Since Equation 3.32 is written as in 3.33, the coefficients A, B and C are as in equation 3.34.

$$Au_{i,j}^2 + Bu_{i,j} + C \tag{3.33}$$

$$\begin{aligned}
A &= \left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y + (3\rho_{i,j}) \Delta y \\
B &= \left((-4u_{i-1,j} + u_{i-2,j}) \rho_{i,j} \Delta y \right) + \left(\left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) v_{i,j} \Delta x \right) \\
&+ \left(\left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \rho_{i,j} \Delta x \right) + \left(\frac{3\rho_{i,j} v_{i,j} \Delta x}{2} \right) \\
C &= - \left(\frac{-3p_{i,j} + 4p_{i-1,j} - p_{i-2,j}}{2} \Delta y \right) + \left(\left(\frac{-4u_{i,j-1} + u_{i,j-2}}{2} \right) \rho_{i,j} v_{i,j} \Delta x \right)
\end{aligned} \tag{3.34}$$

Apply the same procedure to equation 3.30 to find velocity $v_{i,j}$ in y-direction. Equation 3.30 is written as a quadratic equation and separated into $v_{i,j}^2$, $v_{i,j}$ and constants as seen in equation 3.35.

$$\begin{aligned} v_{i,j}^2 \left(\left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x \right) + 3v_{i,j}^2 (\rho_{i,j} \Delta x) + 2v_{i,j} \left(\left(\frac{-4v_{i,j-1} + v_{i,j-2}}{2} \right) \rho_{i,j} \Delta x \right) \\ + v_{i,j} \left(\left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) u_{i,j} \Delta y \right) + v_{i,j} \left(\left(\frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \rho_{i,j} \Delta y \right) \\ + v_{i,j} \left(\frac{3\rho_{i,j} u_{i,j} \Delta y}{2} \right) - \frac{-3p_{i,j} + 4p_{i,j-1} - p_{i,j-2}}{2} \Delta x + \left(\left(\frac{-4v_{i-1,j} + v_{i-2,j}}{2} \right) \rho_{i,j} u_{i,j} \Delta y \right) = 0 \end{aligned} \quad (3.35)$$

Then equation 3.35 is set equal to zero as in equation 3.36.

$$\begin{aligned} v_{i,j}^2 \left(\left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x + 3(\rho_{i,j} \Delta x) \right) + v_{i,j} \left(\left(\left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) u_{i,j} \Delta y \right) \right. \\ \left. + \left(\frac{3\rho_{i,j} u_{i,j} \Delta y}{2} \right) + \left(\left(\frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \rho_{i,j} \Delta y \right) + ((-4v_{i,j-1} + v_{i,j-2}) \rho_{i,j} \Delta x) \right. \\ \left. - \left(\frac{-3p_{i,j} + 4p_{i,j-1} - p_{i,j-2}}{2} \Delta x \right) + \left(\left(\frac{-4v_{i-1,j} + v_{i-2,j}}{2} \right) \rho_{i,j} u_{i,j} \Delta y \right) \right) = 0 \end{aligned} \quad (3.36)$$

Since equation 3.36 is written as a quadratic equation as in 3.37, the coefficients D, E and F are as in equation 3.38.

$$Dv_{i,j}^2 + Ev_{i,j} + F \quad (3.37)$$

$$\begin{aligned} D &= \left(\left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x + 3(\rho_{i,j} \Delta x) \right) \\ E &= \left(\left(\left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) u_{i,j} \Delta y \right) + \left(\left(\frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \rho_{i,j} \Delta y \right) \right. \\ &\quad \left. + \left(\frac{3\rho_{i,j} u_{i,j} \Delta y}{2} \right) + ((-4v_{i,j-1} + v_{i,j-2}) \rho_{i,j} \Delta x) \right) \\ F &= - \left(\frac{-3p_{i,j} + 4p_{i,j-1} - p_{i,j-2}}{2} \Delta x \right) + \left(\left(\frac{-4v_{i-1,j} + v_{i-2,j}}{2} \right) \rho_{i,j} u_{i,j} \Delta y \right) \end{aligned} \quad (3.38)$$

After finding all these equations, the roots of the quadratic equations derived to obtain the velocity values from the Momentum Equations are found using the equations in 3.39 and 3.40.

$$u_{i,j} = -B \pm \frac{\sqrt{B^2 - 4AC}}{2A} \quad (3.39)$$

$$v_{i,j} = -E \pm \frac{\sqrt{E^2 - 4DF}}{2D} \quad (3.40)$$

3.3.3 Energy equations

Energy equations are used in CFD analysis to understand how energy is distributed and exchanged in a fluid. These equations include forms of energy exchange such as heat transfer, internal energy changes and work. The energy equation models the relationships between the thermal properties of the fluid and its velocity and pressure. In this way, CFD simulations can predict not only the mechanical motion of the fluid, but also its thermal behavior. Energy Equations for 2D inviscid, steady and compressible flow are as given in equations 3.41 and 3.42.

$$\frac{\partial(\rho u E)}{\partial x} + \frac{\partial(\rho u K)}{\partial x} = 0 \quad (3.41)$$

$$\frac{\partial(\rho v E)}{\partial y} + \frac{\partial(\rho v K)}{\partial y} = 0 \quad (3.42)$$

Here E is the internal energy and K is the kinetic energy of the fluid. It is usually expressed as in 3.43.

$$K = \frac{1}{2} m V^2 \quad (3.43)$$

m represents the mass and V the fluid velocity in the equation 3.43.

Since we are talking about the examination of very small particles in the flow, m in this equation becomes the unit mass. When the kinetic energy is differentiated with respect to the x -direction, it is specifically related to the velocity in the x -direction. In the same

way when differentiated with respect to the y-direction, it is specifically related to the velocity in the y-direction. Therefore the Energy Equations can be written as in 3.44 and 3.45.

$$\frac{\partial(\rho u E)}{\partial x} + \frac{1}{2} \frac{\partial(\rho u^3)}{\partial x} = 0 \quad (3.44)$$

$$\frac{\partial(\rho v E)}{\partial y} + \frac{1}{2} \frac{\partial(\rho v^3)}{\partial y} = 0 \quad (3.45)$$

Thus, the Leibniz rule can be applied to obtain the derivative for each variable.

$$\frac{\partial \rho}{\partial x} u E + \frac{\partial u}{\partial x} \rho E + \frac{\partial E}{\partial x} \rho u + \frac{\partial \rho}{\partial x} \frac{u^3}{2} + \rho \frac{3u^2}{2} \frac{\partial u}{\partial x} = 0 \quad (3.46)$$

$$\frac{\partial \rho}{\partial y} u E + \frac{\partial v}{\partial y} \rho E + \frac{\partial E}{\partial y} \rho v + \frac{\partial \rho}{\partial y} \frac{v^3}{2} + \rho \frac{3v^2}{2} \frac{\partial v}{\partial y} = 0 \quad (3.47)$$

Since equation 3.46 and 3.47 are in conservation form, they are integrated to carry on with the discretization of the Energy Equations. Equation 3.48 is the integrated form of equation 3.46.

$$\begin{aligned} \int_s^n \int_w^e \frac{\partial \rho}{\partial x} u E dx dy + \int_s^n \int_w^e \frac{\partial u}{\partial x} \rho E dx dy + \int_s^n \int_w^e \frac{\partial E}{\partial x} \rho u dx dy \\ + \int_s^n \int_w^e \frac{\partial \rho}{\partial x} \frac{u^3}{2} dx dy + \int_s^n \int_w^e \rho \frac{3u^2}{2} \frac{\partial u}{\partial x} dx dy = 0 \end{aligned} \quad (3.48)$$

Substituting the integral boundaries gives equations 3.49 and 3.50.

$$\begin{aligned} \int_s^n u E (\rho_e - \rho_w) dy + \int_s^n \rho E (u_e - u_w) dy + \int_s^n \rho u (E_e - E_w) dy \\ + \int_s^n \frac{u^3}{2} (\rho_e - \rho_w) dy + \int_s^n \frac{3u^2 \rho}{2} (u_e - u_w) dy = 0 \end{aligned} \quad (3.49)$$

$$\begin{aligned} u E (\rho_e - \rho_w) \Delta y + \rho E (u_e - u_w) \Delta y + \rho u (E_e - E_w) \Delta y \\ + \frac{u^3}{2} (\rho_e - \rho_w) \Delta y + \frac{3u^2 \rho}{2} (u_e - u_w) \Delta y = 0 \end{aligned} \quad (3.50)$$

Equation 3.51 shows the integrated form of equation 3.47.

$$\begin{aligned}
& \int_w^e \int_s^n \frac{\partial \rho}{\partial y} v E dy dx + \int_w^e \int_s^n \frac{\partial v}{\partial y} \rho E dy dx + \int_w^e \int_s^n \frac{\partial E}{\partial y} \rho v dy dx \\
& + \int_w^e \int_s^n \frac{\partial \rho}{\partial y} \frac{v^3}{2} dy dx + \int_w^e \int_s^n \rho \frac{3v^2}{2} \frac{\partial v}{\partial y} dy dx = 0
\end{aligned} \tag{3.51}$$

By substituting the integral boundaries the equations 3.52 and 3.53 are obtained.

$$\begin{aligned}
& \int_w^e v E (\rho_n - \rho_s) dx + \int_w^e \rho E (v_n - v_s) dx + \int_w^e \rho v (E_n - E_s) dx \\
& + \int_w^e \frac{v^3}{2} (\rho_n - \rho_s) dx + \int_w^e \frac{3v^2 \rho}{2} (v_n - v_s) dx = 0
\end{aligned} \tag{3.52}$$

$$\begin{aligned}
& v E (\rho_n - \rho_s) \Delta x + \rho E (v_n - v_s) \Delta x + \rho v (E_n - E_s) \Delta x \\
& + \frac{v^3}{2} (\rho_n - \rho_s) \Delta x + \frac{3v^2 \rho}{2} (v_n - v_s) \Delta x = 0
\end{aligned} \tag{3.53}$$

When equation 3.50 and 3.53 are reformulated using the second-order upwind approximation, as detailed in statement 3.4 for a positive direction, the discretized equations in 3.54 and 3.55 are obtained respectively.

$$\begin{aligned}
& u_{i,j} E_{i,j} \left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y + \rho_{i,j} E_{i,j} \left(\frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \Delta y \\
& + \rho_{i,j} u_{i,j} \left(\frac{3E_{i,j} - 4E_{i-1,j} + E_{i-2,j}}{2} \right) \Delta y + \frac{u_{i,j}^3}{2} \left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y \\
& + \frac{3u_{i,j}^2 \rho_{i,j}}{2} \left(\frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \Delta y = 0
\end{aligned} \tag{3.54}$$

$$\begin{aligned}
& v_{i,j} E_{i,j} \left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x + \rho_{i,j} E_{i,j} \left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \Delta x \\
& + \rho_{i,j} v_{i,j} \left(\frac{3E_{i,j} - 4E_{i,j-1} + E_{i,j-2}}{2} \right) \Delta x + \frac{v_{i,j}^3}{2} \left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x \\
& + \frac{3v_{i,j}^2 \rho_{i,j}}{2} \left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \Delta x = 0
\end{aligned} \tag{3.55}$$

If Equations 3.54 and 3.55 are bracketed with $E_{i,j}$, Equations 3.56 and 3.57 are obtained.

$$\begin{aligned}
E_{i,j} & \left(\left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y u_{i,j} + \left(\frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \Delta y \rho_{i,j} + \frac{3\rho_{i,j} u_{i,j} \Delta y}{2} \right) \\
& = - \left(\frac{-4E_{i-1,j} + E_{i-2,j}}{2} \right) \Delta y \rho_{i,j} u_{i,j} - \frac{u_{i,j}^3}{2} \left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y \\
& \quad - \frac{3u_{i,j}^2 \rho_{i,j}}{2} \left(\frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \Delta y
\end{aligned} \tag{3.56}$$

$$\begin{aligned}
E_{i,j} & \left(\left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x v_{i,j} + \left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \Delta x \rho_{i,j} + \frac{3\rho_{i,j} v_{i,j} \Delta x}{2} \right) \\
& = - \left(\frac{-4E_{i,j-1} + E_{i,j-2}}{2} \right) \Delta x \rho_{i,j} v_{i,j} - \frac{v_{i,j}^3}{2} \left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x \\
& \quad - \frac{3v_{i,j}^2 \rho_{i,j}}{2} \left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \Delta x
\end{aligned} \tag{3.57}$$

If the term $E_{i,j}$ is left alone on the left-hand side of the equations, the equations 3.58 and 3.59 are obtained. The equations obtained here include the energy calculation, which differs with respect to the x - and y -axes.

$$\begin{aligned}
E_{i,j\mathbf{x}} & = - \left(\frac{-4E_{i-1,j} + E_{i-2,j}}{2} \right) \Delta y \rho_{i,j} u_{i,j} - \frac{u_{i,j}^3}{2} \left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y \\
& \quad - \frac{3u_{i,j}^2 \rho_{i,j}}{2} \left(\frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \Delta y \Bigg/ \left(\left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y u_{i,j} \right. \\
& \quad \left. + \left(\frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \Delta y \rho_{i,j} + \frac{3\rho_{i,j} u_{i,j} \Delta y}{2} \right)
\end{aligned} \tag{3.58}$$

$$\begin{aligned}
E_{i,j\mathbf{y}} & = - \left(\frac{-4E_{i,j-1} + E_{i,j-2}}{2} \right) \Delta x \rho_{i,j} v_{i,j} - \frac{v_{i,j}^3}{2} \left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x \\
& \quad - \frac{3v_{i,j}^2 \rho_{i,j}}{2} \left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \Delta x \Bigg/ \left(\left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x v_{i,j} \right. \\
& \quad \left. + \left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \Delta x \rho_{i,j} + \frac{3\rho_{i,j} v_{i,j} \Delta x}{2} \right)
\end{aligned} \tag{3.59}$$

Equations 3.58 and 3.59 are then averaged to find the total energy derived in equation 3.60.

$$\begin{aligned}
E_{i,j} = & 0.5 \left(- \left(\frac{-4E_{i-1,j} + E_{i-2,j}}{2} \right) \Delta y \rho_{i,j} u_{i,j} - \frac{u_{i,j}^3}{2} \left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y \right. \\
& - \frac{3u_{i,j}^2 \rho_{i,j}}{2} \left(\frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \Delta y \Bigg/ \left(\left(\frac{3\rho_{i,j} - 4\rho_{i-1,j} + \rho_{i-2,j}}{2} \right) \Delta y u_{i,j} \right. \\
& \left. + \left(\frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2} \right) \Delta y \rho_{i,j} + \frac{3\rho_{i,j} u_{i,j} \Delta y}{2} \right) \\
& + 0.5 \left(- \left(\frac{-4E_{i,j-1} + E_{i,j-2}}{2} \right) \Delta x \rho_{i,j} v_{i,j} - \frac{v_{i,j}^3}{2} \left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x \right. \\
& - \frac{3v_{i,j}^2 \rho_{i,j}}{2} \left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \Delta x \Bigg/ \left(\left(\frac{3\rho_{i,j} - 4\rho_{i,j-1} + \rho_{i,j-2}}{2} \right) \Delta x v_{i,j} \right. \\
& \left. + \left(\frac{3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}}{2} \right) \Delta x \rho_{i,j} + \frac{3\rho_{i,j} v_{i,j} \Delta x}{2} \right)
\end{aligned} \tag{3.60}$$

3.3.4 Temperature

Since inviscid effect will be investigated in this thesis, it was decided to model air as an ideal gas. Therefore, for temperature, an equation will be used that states that the total internal energy held or released by a unit mass of a substance under constant pressure, which is especially valid for ideal gases, is directly proportional to the temperature variation of that substance. The mathematical formula for this statement is written as given in equation 3.61

$$T = \frac{E}{Cp} \tag{3.61}$$

Here T represents temperature and Cp represents specific heat capacity. Cp is taken as $1006.43 \frac{J}{kg \cdot K}$ for this study.

3.3.5 Pressure

Usually in compressible flow problems and high velocity flows, especially when the Mach number is greater than 0.3, the pressure can be calculated using the ideal gas equation. Since this thesis focuses on high velocity compressible flows, the pressure is obtained using the ideal gas equation given in equation 3.62.

$$p = \rho RT \quad (3.62)$$

where R is gas constant taken as $287.1 \frac{J}{kg \cdot K}$.

3.4 Initial Model Algorithm

The equations derived from the initial modeling phase are scripted with a specific algorithm to be used in MATLAB simulation. The details are given in Algorithm 1. This algorithm carefully outlines the procedural steps to simulate fluid dynamics within a computational domain of 100 cells. Initially, the process involves establishing initial conditions for various flow properties, including velocity components (u and v), energy (E), temperature (T) and pressure (p), providing a basic foundation for the simulation. The density is initialized using the ideal gas equation. The algorithm then proceeds to solve the Euler Equations, including the Continuity, Momentum and Energy equations.

Algorithm 1: Initial Model

Result: ρ, u, v, E, T, p

initialization;

$u = u^0, v = v^0, E = E^0, T = T^0, p = p^0$

Calculate initial ρ from equation 3.62

$\rho^n = \rho^0;$

for iteration $n = 1, 2, \dots, 100$ **do**

for $i = 3, \dots, 10$ **do**

for $j = 3, \dots, 10$ **do**

do in parallel

 Solve equation 3.12 for Continuity and find $\rho_{i,j}^n$

 Solve equation 3.29 and 3.30 for Momentum and find $u_{i,j}^n, v_{i,j}^n$

 Solve equation 3.60 for Energy and find $E_{i,j}^n$

end

$\rho_{i,j} = \rho_{i,j}^n, u_{i,j} = u_{i,j}^n, v_{i,j} = v_{i,j}^n, E_{i,j} = E_{i,j}^n$

do in parallel

 Solve equation (3.61) for Temperature and find $T_{i,j}^n$

 Solve equation (3.62) for Pressure and find $p_{i,j}^n$

end

$T_{i,j} = T_{i,j}^n, p_{i,j} = p_{i,j}^n$

end

end

end

The algorithm solves the Continuity Equation 3.12 to update the density ($\rho_{i,j}^n$) at each node. Simultaneously, the Momentum Equations 3.29-3.30 are solved to determine the velocity components ($u_{i,j}^n$ and $v_{i,j}^n$). The Energy Equation 3.60 is solved to calculate the thermal energy dissipation in the flow and determine ($E_{i,j}^n$). Following the resolution of the supplementary equations for temperature 3.61 and pressure 3.62, the derived values are updated to reflect the new fluid properties, prompting the commencement of the subsequent iteration.

The algorithm calculates 100 cells in 100 iterations. The ultimate initial model is created by writing MATLAB scripts according to the algorithm given in this section. The design and accuracy of the first model is of great importance. According to the design methodology, model refining will be carried after the initial model is built. For this reason, the initial model forms the basis of the entire study.



4. ASSESSMENT OF INITIAL MODEL ACCURACY

In this chapter, a comparison and evaluation will be presented between the MATLAB simulation results of the initial model detailed in Chapter 3 and the results obtained under the same conditions using the commercial CFD software Ansys Fluent. Firstly, how modeling is done in Ansys Fluent is explained and simulation details are given. Then, the difference between MATLAB simulation and Ansys Fluent simulation is examined and MATLAB simulation is validated. This process ensures a comprehensive comparison between the theoretical model developed in MATLAB and its practical implementation in a commercial CFD environment, thus validating the initial model's effectiveness and accuracy in simulating real-world fluid dynamics.

4.1 Ansys Modelling

In this thesis, it was decided to reproduce the initial model simulated in MATLAB in Ansys Fluent for comparison purposes. This approach facilitates the verification of the designed initial model. In ANSYS Fluent commercial software, the flow field designed for analysis is defined using the Design Modeler option. Considering the purposes of this study, the flow field is characterized as a square area with dimensions of one meter by one meter. Then, the geometry is carried to the next step for mesh generation.

A meticulous mesh analysis is undertaken for the devised geometry, recognizing mesh quality as a crucial factor in computational analysis. Inadequate or sub-optimal meshing can significantly compromise the accuracy of the results. In this thesis, the grid across the flow domain's x - and y -axes is uniformly segmented into 10 parts each via edge sizing, leading to a configuration of 100 squares and 121 nodes. Given the investigation of an unobstructed flow field within this research, an extensive mesh refinement was deemed unnecessary due to the inherently high quality of the mesh. The evaluation of the mesh's integrity focused on orthogonal quality as the primary indicator. Orthogonal quality, ranging from 0 to 1, serves as a measure where values

closer to 1 signify superior mesh quality, with those above 0.7 typically indicating a high-quality mesh [28]. The lowest recorded orthogonal quality is exceptionally high at 0.999877 for the mesh analysis studied in this thesis. Illustrations of the constructed mesh alongside the designated velocity inlets and outlets are presented in Figure.4.1.

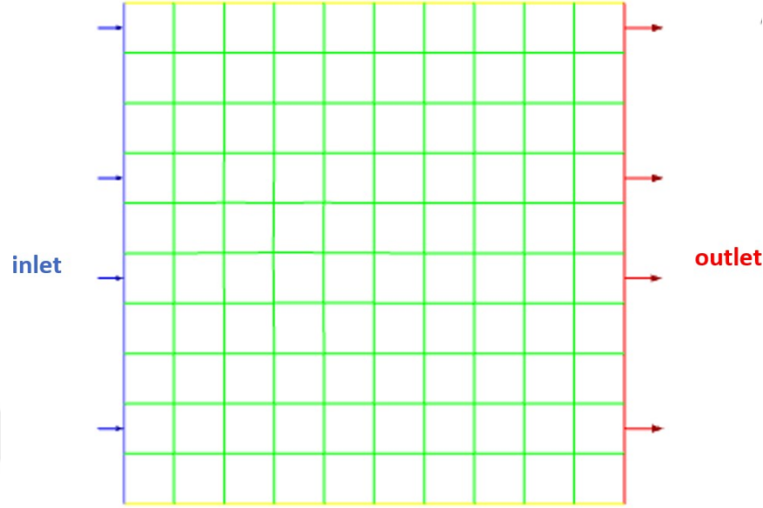


Figure 4.1 : Mesh structure and velocity inlet and outlet.

After meshing, the next step in Ansys Fluent involves the setup process. This step involves defining the flow characteristics using the selected parameters and then running the simulation according to these settings. Since the computer used for this analysis has 4 cores, 4 is entered in the process section. A parallel solver is used and the simulation is performed using a double precision setting.

The setup process is then started by entering all the parameters required for the analysis into the system. Initially, the analysis is set to use a Density-based solver. For the purposes of this study, the turbulence model chosen is inviscid to be compatible with the resolution of the Euler Equations. Activation of the energy equations is also performed. Since air is modeled as an ideal gas, it is chosen as the ideal gas in the software and the specific heat capacity, C_p , is assumed constant at $\frac{J}{kg \cdot K}$ 1006.43 $\frac{J}{kg \cdot K}$. The inlet velocity is defined as 1000 m/s in the x -direction (equivalent to Mach 2.8) and 0 m/s in the y -direction. The pressure at the inlet is set to 101325 Pa and the initial temperature to 300 K. A second order upwind scheme was chosen for discretization and with this, all set-up parameters and solution methods are set to match the MATLAB simulation exactly. All setup parameters are provided in Table 4.1.

Table 4.1 : Set-up Parameters

Initial values and constants	
Density	$1.176655 \frac{kg}{m^3}$
Pressure	$101325 Pa$
Velocity u	$1000 m/s$
Velocity v	$0 m/s$
Temperature	$300 K$
Specific heat constant	$1006.43 \frac{J}{kg \cdot K}$
Gas constant	$287.1 \frac{J}{kg \cdot K}$

Following these configurations, the simulation is run for 100 iterations. Ansys Fluent simulation results are saved for comparison with MATLAB simulation results.

4.2 Assessment

Employing the algorithm outlined in Chapter 3, under the "Initial Model Algorithm" section, the simulation incorporating the discretized equations was conducted in MATLAB, maintaining identical initial conditions and iteration count.

Figure 4.2 - 4.7 shows the difference between the calculations for each property of the flow in simulations using Ansys Fluent and MATLAB script. The differences in the calculation of density, pressure, velocity u , velocity v , energy and temperature are shown respectively. The difference between the results is determined by comparing the results obtained from both solutions at each node during the last iteration. The comparison plots reveal minimal differences between MATLAB simulations derived from mathematically discretized equations and the analysis carried out in ANSYS Fluent. Such an agreement is remarkable in the field of CFD simulations, where small changes in numerical methodologies, network configurations or solver setups can yield markedly different results. The fact that these results are almost identical underlines the accuracy of the MATLAB code used.

The results prove the accuracy of the initial model developed. According to the Flowchart in Figure 2.2, it means that the initial model has been validated and assessment has been done and it is possible to proceed to the next step: Elaboration.

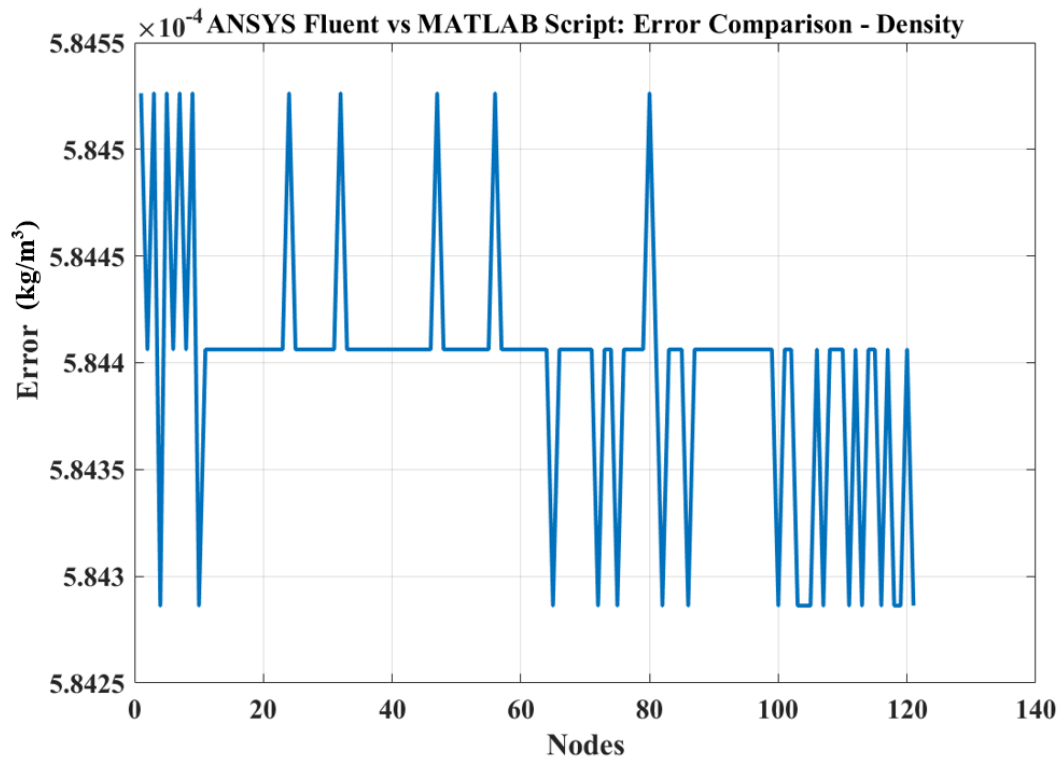


Figure 4.2 : Comparison of *density* results between ANSYS Fluent and MATLAB script.

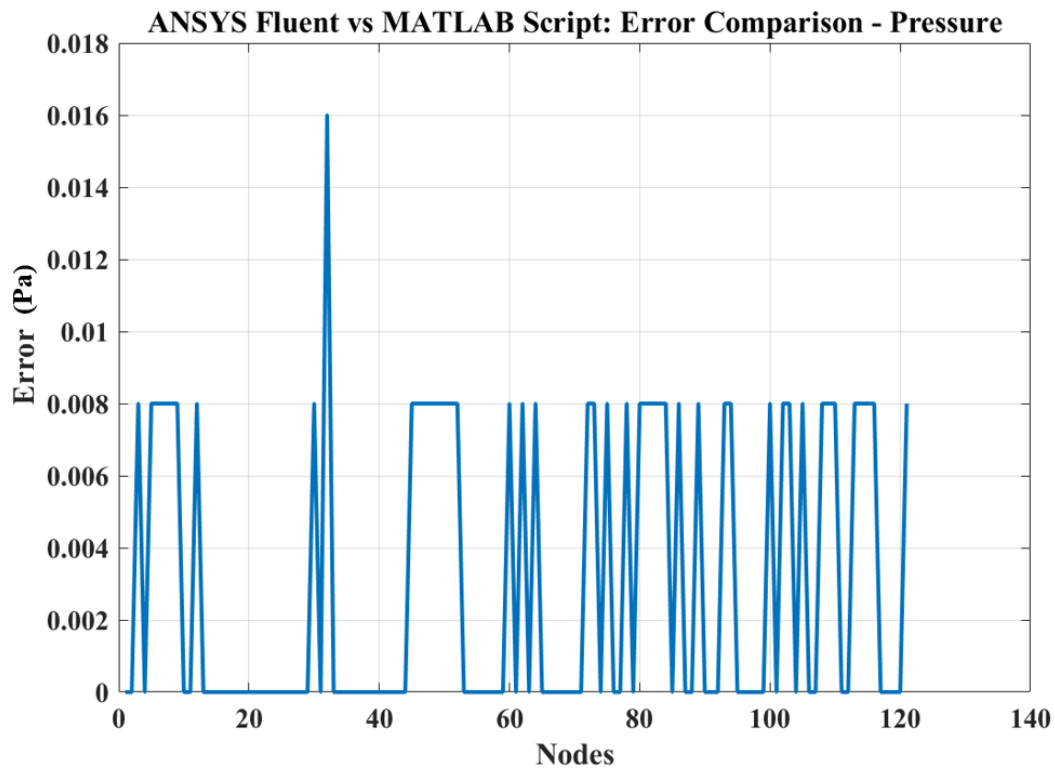


Figure 4.3 : Comparison of *pressure* results between ANSYS Fluent and MATLAB script.

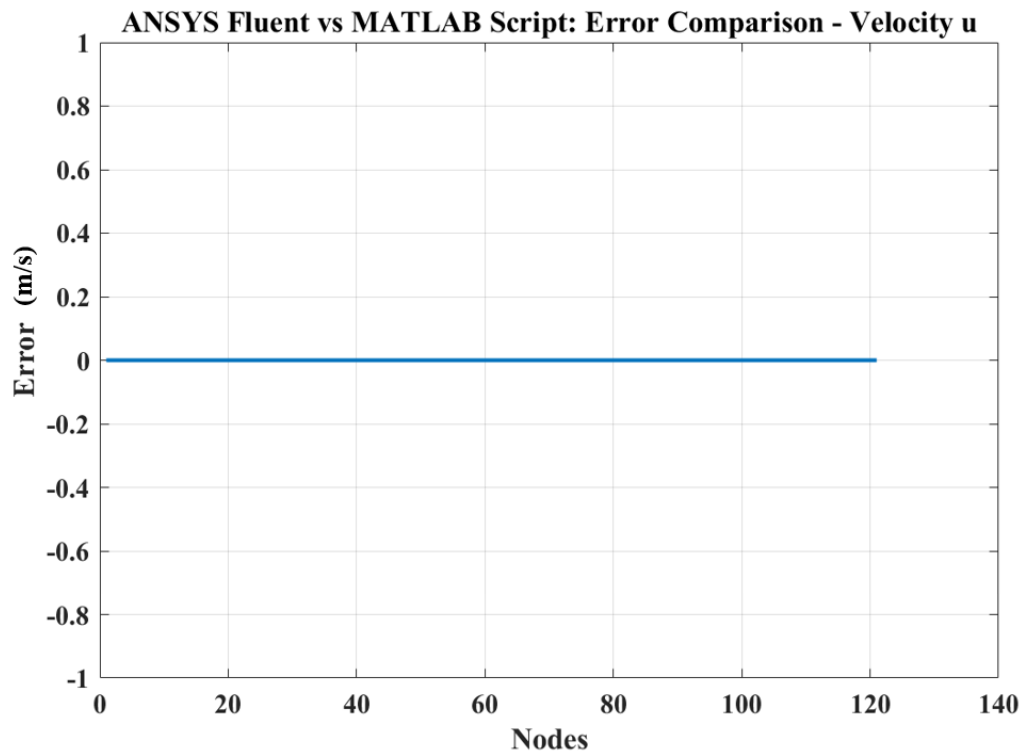


Figure 4.4 : Comparison of x -velocity (u) results between ANSYS Fluent and MATLAB script.

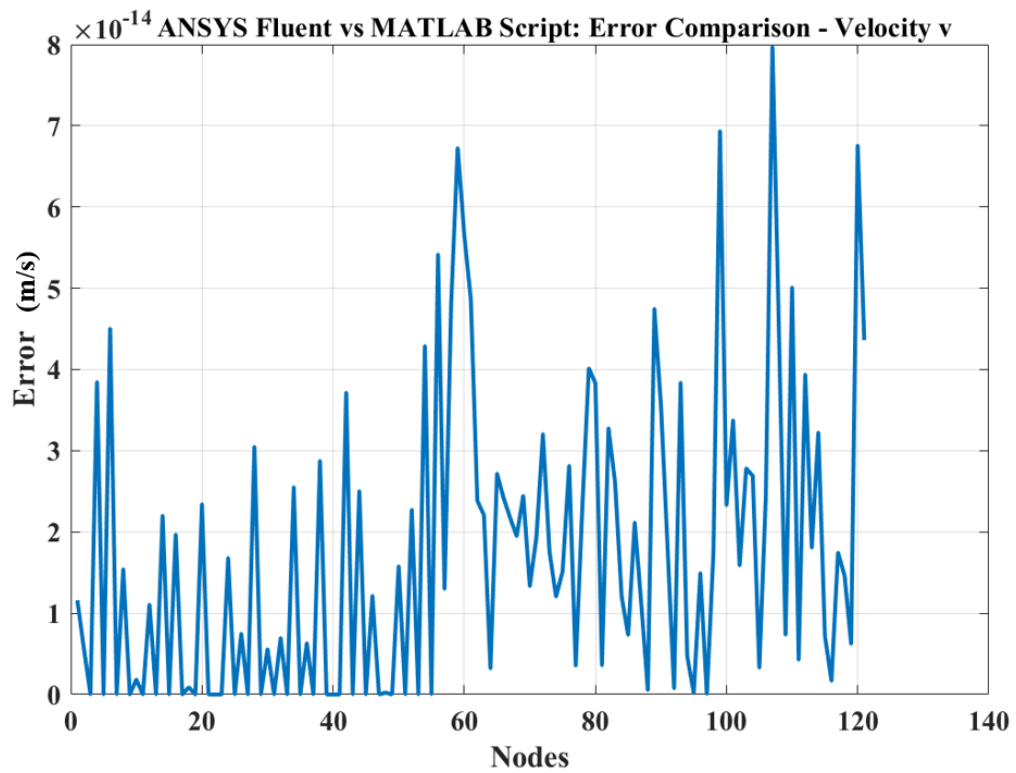


Figure 4.5 : Comparison of y -velocity (v) results between ANSYS Fluent and MATLAB script.

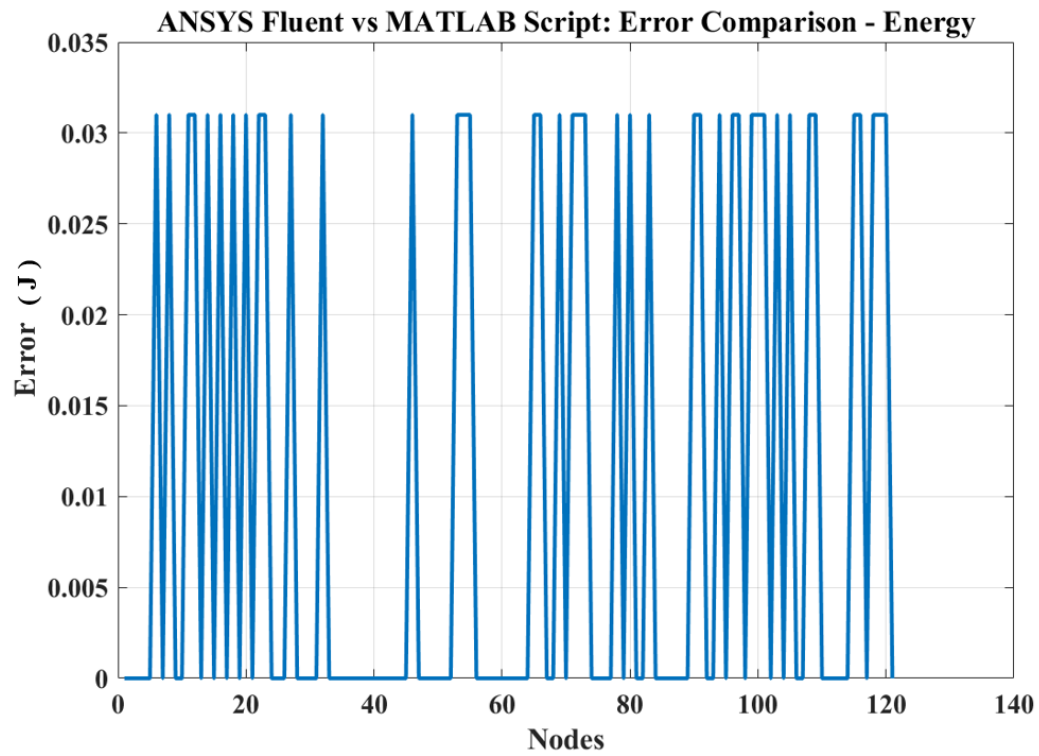


Figure 4.6 : Comparison of *energy* results between ANSYS Fluent and MATLAB script.

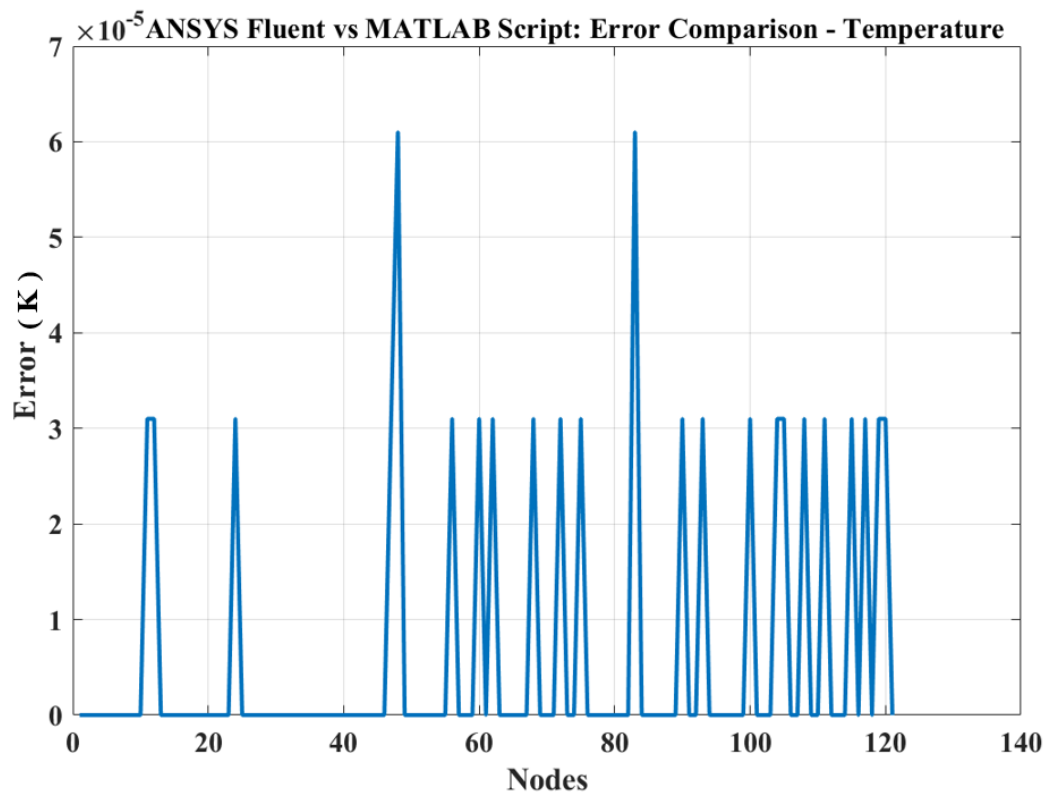


Figure 4.7 : Comparison of *temperataure* results between ANSYS Fluent and MATLAB script.

5. MODEL ELABORATION AND ASSESSMENT

In this chapter, the next step in the design methodology is taken. In this step, elaboration will be carried out for the designed model. How this step is carried out is explained, and then the time and resource performance results of the FPGA are evaluated.

Following the comparison and validation of the outcomes derived from the MATLAB script, which employs discretized equations, against those obtained from Ansys Fluent, the HDL codes are subsequently produced. Distinct HDL codes are meticulously crafted for each equation involved. The necessary equations for the elaboration process are systematically arranged as detailed in Appendix-A/D, exclusively focusing on computations pertaining to a single finite volume. These equations are meticulously segmented into atomic rows, ensuring that each row encapsulates only a singular mathematical operation.

Following the validation and alignment of the results from the MATLAB scripts, the corresponding HDL codes are meticulously generated. This generation process is facilitated by the MATLAB HDL Coder, which adeptly translates the detailed script codes into HDL format, specifically tailored for solving the Euler Equations within a singular cell. The newly crafted HDL codes are then subjected to a secondary synthesis process, once again employing the MATLAB HDL Coder tool for this purpose. For this synthesis, the 'xczu7cg-ffvf1517-2L-e' device, belonging to the esteemed Zynq UltraScale+ MPSoCs family of chips, is judiciously selected. The outcomes of the synthesis are thoroughly evaluated by initiating the project, created within MATLAB HDLcoder, in the Vivado environment. This evaluation meticulously examines FPGA resource usage, focusing on the critical resources available, such as Look-Up Tables (LUTs), LUT Random Access Memory (LUTRAM), Flip-Flops (FFs), Digital Signal Processors (DSPs), and Buffers (BUFGs), cataloging their allocation across various computations like continuity, momentum, energy, temperature, and pressure. The

summarized results of this comprehensive assessment are prominently displayed in Table 5.1.

Table 5.1 : Resource Utilization for Equations

Resource	Available	Cont. Est.	Mom. Est.	En. Est.	Temp. Est.	Pres. Est.
LUT	230400	21384	172633	183818	5155	1759
LUTRAM	101760	1785	12561	10844	232	174
FF	460800	27259	184975	195533	6903	2600
DSP	1728	72	324	333	0	18
BUFG	544	1	1	1	1	1

The data presented in Table 5.1 offers a granular view of the resource utilization for each category of equation processed. For instance, it highlights the extensive use of LUTs for momentum and energy equations, indicative of the complex logical operations these particular computations necessitate. Similarly, the allocation of DSPs, predominantly utilized in momentum and energy calculations, underscores the intensive arithmetic processing required. The minimal use of BUFPGs across all equations underscores their specific application for synchronization purposes within the FPGA architecture.

This detailed breakdown not only demonstrates the efficient use of FPGA resources in processing the discrete equations critical to solving CFD problems but also showcases the potential for optimization in future iterations of the design. It underscores the importance of carefully selecting and managing FPGA resources to achieve an optimal balance between computational efficiency and resource utilization, paving the way for more advanced simulations that could encompass larger computational domains or more complex physical phenomena.

Considering that the total number of required resources exceeds the capacity available in a single device, it becomes necessary to include two FPGA devices to accommodate the entire design.

The total time of all equations solved by the design for a cell is calculated via MATLAB HDL Coder. In this thesis, one iteration time is accepted as the performance criterion. The time required to solve all equations in 1 iteration for a finite volume is 606.816

ns. This outcome achieves a speed 1.17 times greater than the findings reported in [23], where Euler equations were solved in one dimension on a CPU. Furthermore, it demonstrates a 3.18 times increase in speed relative to the work of [21], which addressed the solution of 2D Euler equations on a CPU.





6. CONCLUSION and FUTURE WORK

6.1 Conclusion

In this thesis, an FPGA based hardware accelerator was designed using model based design to accelerate CFD analysis. This designed hardware accelerator was first simulated in MATLAB by creating an initial model. The accuracy of the initial model was tested with Ansys Fluent results. Model Elaboration was then performed on the initial model. After model elaboration, the performance of the final model created using Matlab HDL Coder was examined and compared with previous studies.

For this purpose, the Hardware design methodology to be followed in this thesis is first explained. According to this methodology, the Euler equations are first discretized using FVM and second-order upwind method. Then the discretized equations are scripted in MATLAB according to a certain algorithm and MATLAB simulation is completed. This MATLAB simulation is called the initial model.

Since the accuracy of the initial model is important for the realism of the study, the results were compared by analyzing the same conditions in Ansys Fluent commercial software. The difference between the MATLAB simulation and Ansys Fluent results was found to be acceptably small. Thus, the assessment and validation of the initial model was performed.

Once the accuracy of the initial model was proven, the next stage of the design phase was started. In the Model Elaboration section, the first matlab codes written in the initial model were restructured as dynamic blocks. After this configuration, the codes written using MATLAB HDL Coder were translated into HDL. The codes translated into HDL language were again transferred to Xilinx Vivado via MATLAB. Here resource utilization was examined. Then, after the automatic models of the generated codes were created in MATLAB simulink, it was calculated how many seconds it takes the system to perform one iteration for a mesh.

According to the results obtained, solving the Euler equations in a single cell requires an iteration time of 606.816 nanoseconds. This achievement represents a speedup of 1.17 times compared to traditional one-dimensional CPU methodologies and an impressive 3.18 times improvement compared to previous two-dimensional CPU-based approaches. These developments underline the potential of FPGA-based hardware accelerators to significantly reduce the time-cost of CFD analysis.

6.2 Future Work

Future investigations should focus on broadening the scope of the existing hardware architecture to incorporate an expanded number of cells within a single FPGA unit. This advancement aims to amplify the system's computational prowess for tackling increasingly intricate simulations, all the while adeptly handling the surge in processing requirements. Moreover, there is a pressing need to refine the discretization techniques employed, particularly within the realm of the second-order upwind method. Such enhancements are vital for accurately modeling scenarios that include obstacles within the flow field, thereby necessitating the formulation of advanced mathematical models that encapsulate flow dynamics from every conceivable direction.

Transitioning the solver from two dimensions to three dimensions stands out as a critical stride towards encapsulating the full spectrum of physical phenomena in simulations, thereby paving the way for applications that demand a higher degree of realism. The collective pursuit of these research avenues promises not only to push the boundaries of design and simulation capabilities but also to expand the horizon of possible applications, marking substantial progress in the field.

REFERENCES

- [1] **Anderson** (2009). *Computational Fluid Dynamics*, McGraw-Hill.
- [2] **Carlson, J., Jaffe, J. and Wiles, A.** (2006). *The Millennium Prize Problems*, Clay Mathematics Institute and American Mathematical Society.
- [3] **Khalil, E.E.** (2012). CFD History and Applications, *CFD Letters*, 4(2), 43–46.
- [4] **DeLorimier, M. and DeHon, A.** (2005). Floating-point sparse matrix-vector multiply for FPGAs, *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*, pp.75–85.
- [5] **Yeniceri, R. and Yalcin, M.E.** (2009). An emulated digital wave computer core implementation, *2009 European Conference on Circuit Theory and Design*, IEEE, pp.831–834.
- [6] **Memis, S. and Yeniceri, R.** (2023). Towards fpga based digital twin of uav swarms: An area efficient hardware accelerator of transformation matrix of 6-dof block, *AIAA SCITECH 2023 Forum*, p.2130.
- [7] **Farhan, I.H.** (1991). *Theoretical and experimental aerodynamic analysis for high-speed ground vehicles*, Loughborough University.
- [8] **Hosseini, E.** (2019). CFD analysis of the aerodynamic characteristics of biconvex airfoil at compressible and high Mach numbers flow, *SN Applied Sciences*, 1(10), 1283.
- [9] **Yigit, E.E. and Yeniceri, R.** (2024). FPGA Based Hardware Accelerator for Euler Equations with Finite Volume Method, *AIAA SCITECH 2024 Forum*, p.0044.
- [10] **Altayan, W. and Alonso, J.J.** (2023). Accelerating the Lattice Boltzmann Method, *2023 IEEE Aerospace Conference*, IEEE, pp.1–20.
- [11] **Rojek, K., Rojek, M. and Paropkari, V.R.** (2019). CFD Acceleration with FPGA, *2019 Great Lakes Symposium on VLSI*, International Workshop on Heterogeneous High-performance Reconfigurable Computing, pp.87–92.
- [12] **Rojek, K., Halbiniak, K. and Kuczynski, L.** (2021). CFD code adaptation to the FPGA architecture, *The International Journal of High Performance Computing Applications*, 35(1), 33–46.

- [13] **Nagy, Z., Nemes, C., Hiba, A., Kiss, A., Csík, Á. and Szolgay, P.** (2012). FPGA based acceleration of computational fluid flow simulation on unstructured mesh geometry, *22nd International Conference on Field Programmable Logic and Applications (FPL)*, IEEE, pp.128–135.
- [14] **Du, C., Firmansyah, I. and Yamaguchi, Y.** (2020). High-Performance Computation of LGCA Fluid Dynamics on an FPGA-Based Platform, *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, IEEE, pp.520–525.
- [15] **Andres, E., Widhalm, M. and Caloto, A.** (2009). Achieving high speed CFD simulations: Optimization, parallelization, and FPGA acceleration for the unstructured DLR TAU code, *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, p.759.
- [16] **Xiang, Y., Yu, B., Yuan, Q. and Sun, D.** (2017). GPU acceleration of CFD algorithm: HSMAC and SIMPLE, *Procedia Computer Science*, 108, 1982–1989.
- [17] **Hüner, Y., Gayretli, M.G. and Yeniçeri, R.** (2021). Hw/sw design space exploration of a complementary filter on zynq soc, *2021 8th International Conference on Electrical and Electronics Engineering (ICEEE)*, IEEE, pp.1–5.
- [18] **Senel, S. and Yeniceri, R.** (2023). A Hardware Accelerator Design for Quaternion to Euler Angles Conversion, *2023 30th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, IEEE, pp.1–4.
- [19] **Memis, S.** (2023). *A Refined Methodology for Model-Based FPGA Hardware Design: An Example of Quadrotor Dynamical Model Implementation*, Istanbul Technical University, Istanbul, Turkey.
- [20] **Talip, M.S.A. and Amano, H.** (2011). A Design of One-Dimensional Euler Equations for Fluid Dynamics on FPGA, *1st International Symposium on Access Spaces (ISAS) IEEE-ISAS 2011*, IEEE, pp.170–173.
- [21] **Sanchez-Roman, D., Sutter, G., Lopez-Buedo, S., Gonzalez, I., Gomez-Arribas, F.J. and Aracil, J.** (2011). An euler solver accelerator in FPGA for computational fluid dynamics applications, *2011 VII Southern Conference on Programmable Logic (SPL)*, IEEE, pp.149–154.
- [22] **Gan, L., Fu, H., Yang, C., Luk, W., Xue, W., Mencer, O., Huang, X. and Yang, G.** (2014). A highly-efficient and green data flow engine for solving euler atmospheric equations, *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, IEEE, pp.1–6.
- [23] **Ebrahimi, A. and Zandsalimy, M.** (2017). Evaluation of FPGA hardware as a new approach for accelerating the numerical solution of CFD problems, *IEEE Access*, 5, 9717–9727.

- [24] **Kamalakkannan, K., Mudalige, G.R., Reguly, I.Z. and Fahmy, S.A.** (2021). High-level FPGA accelerator design for structured-mesh-based explicit numerical solvers, *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, IEEE, pp.1087–1096.
- [25] **Tomczak, T., Księżyk, M., Hanke, J., Kostur, M. and Marek, M.** (2023). Double-precision hardware accelerator for incompressible Navier–Stokes equations solver based on discontinuous Galerkin method, *Advances in Engineering Software*, 176, 103393.
- [26] **Mazumder, S.** (2015). *Numerical methods for partial differential equations: finite difference and finite volume methods*, Academic Press.
- [27] **Moukalled, F., Mangani, L., Darwish, M., Moukalled, F., Mangani, L. and Darwish, M.** (2016). *The finite volume method*, Springer.
- [28] **Adam, N.M., Attia, O.H., Al-Sulttani, A.O., Mahmood, H.A., As’arry, A. and Rezali, K.A.M.** (2020). Numerical analysis for solar panel subjected with an external force to overcome adhesive force in desert areas, *CFD Letters*, 12(9), 60–75.



APPENDICES

APPENDIX A : Elaboration of Continuity Equation

APPENDIX B : Elaboration of Momentum Equations

APPENDIX C : Elaboration of Energy Equations

APPENDIX D : Elaboration of Temperature and Pressure Equation





APPENDIX A : Elaboration of Continuity Equation

```
1 % Continuity Equation Solution
2 % Calculates new density (ro)
3 function ro = continuity_solution(u_i_j , u_i_jminus1 , ...
4     u_i_j_minus2 , v_i_j , v_iminus1_j , ...
5     v_iminus2_j , ro_ijminus1 , ro_i_jminus2 , ...
6     roiminus1_j , ro_iminus2_j , dx , dy)
7
8 r1 = u_i_j * dy;
9 r2 = 0.5 * r1;
10 r3 = 4 * ro_ijminus1;
11 r4 = r3 - ro_i_jminus2;
12 r5 = r2 * r4;
13 r6 = v_i_j * dx;
14 r7 = 0.5 * r6;
15 r8 = 4 * roiminus1_j;
16 r9 = r8 - ro_iminus2_j;
17 r10 = r7 * r9;
18 r11 = r5 + r10;
19 r12 = 6 * u_i_j;
20 r13 = 4 * u_i_jminus1;
21 r14 = r12 - r13;
22 r15 = r14 + u_i_j_minus2;
23 r16 = r15 * dy;
24 r17 = 6 * v_i_j;
25 r18 = 4 * v_iminus1_j;
26 r19 = r17 - r18;
27 r20 = r19 + v_iminus2_j;
28 r21 = r20 * dx;
29 r22 = r16 + r21;
30 r23 = 0.5 * r22;
31 ro = r11 / r23;
32
33 end % of Continuity Equation solution
```



APPENDIX B : Elaboration of Momentum Equation

```
1 % Momentum Equation Solution
2 % Calculates new x-velocity (u) and y-velocity(v)
3 function [u,v]=momentum_solution(u_i_j , u_i_jminus1 ,...
4     u_i_jminus2 , u_iminus2_j , u_iminus1_j ,...
5     v_i_j , v_iminus1_j ,v_iminus2_j ,...
6     v_i_jminus1 , v_i_jminus2 , ro_i_j ,...
7     ro_i_jminus1 , ro_i_jminus2 ,...
8     ro_iminus1_j , ro_iminus2_j , p_i_j ,...
9     p_i_jminus1 ,p_i_jminus2 , p_iminus1_j ,...
10    p_iminus2_j ,dx ,dy)
11
12 a1 = 3 * ro_i_j;
13 a2 = -4 * ro_i_jminus1;
14 a3 = a1 + a2;
15 a4 = a3+ ro_i_jminus2;
16 a5 = a4 / 2;
17 a6 = a5 * dy;
18 a7 = a1 * dy;
19 A = a6 + a7;
20
21 b1 = -4 * u_i_jminus1;
22 b2 = b1 + u_i_jminus2;
23 b3 = b2 * ro_i_j;
24 b4 = b3 * dy;
25 b5 = 3 * ro_i_j;
26 b6 = -4 * ro_iminus1_j;
27 b7 = b5 + b6;
28 b8 = b7 + ro_iminus2_j;
29 b9 = b8 / 2;
30 b10 = b9 * v_i_j;
31 b11 = b10 * dx;
32 b12 = 3 * v_i_j;
33 b13 = -4 * v_iminus1_j;
34 b14 = b12 + b13;
35 b15 = b14 + v_iminus2_j;
36 b16 = b15 / 2;
37 b17 = b16 * ro_i_j;
38 b18 = b17 * dx;
39 b19 = 3 * ro_i_j;
40 b20 = b19 * v_i_j;
41 b21 = b20 * dx;
42 b22 = b21 / 2;
43 B1 = b4 + b11;
```

```

44 B2 = B1 + b18;
45 B = B2 + b22;
46
47 c1 = -3 * p_i_j;
48 c2 = 4 * p_i_jminus1;
49 c3 = c1 + c2;
50 c4 = c3 - p_i_jminus2;
51 c5 = c4 / 2;
52 c6 = -c5 * dy;
53 c7 = -4 * u_iminus1_j;
54 c8 = c7 + u_iminus2_j;
55 c9 = c8 / 2;
56 c10 = c9 * ro_i_j;
57 c11 = c10 * v_i_j;
58 c12 = c11 * dx;
59 C = c6 + c12;
60
61 u1 = B * B;
62 u2 = -4 * A;
63 u3 = u2 * C;
64 u4 = u1 + u3;
65 u5 = sqrt(u4);
66 u6 = 2 * A;
67 u7 = -B + u5;
68 u = u7 / u6;
69
70 d1 = 3 * ro_i_j;
71 d2 = -4 * ro_iminus1_j;
72 d3 = d1 + d2;
73 d4 = d3 + ro_iminus2_j;
74 d5 = d4 / 2;
75 d6 = d5 * dx;
76 d7 = 3 * ro_i_j * dx;
77 D = d6 + d7;
78
79 e1 = -4 * v_iminus1_j;
80 e2 = e1 + v_iminus2_j;
81 e3 = e2 * ro_i_j;
82 e4 = e3 * dx;
83 e5 = 3 * ro_i_j;
84 e6 = -4 * ro_i_jminus1;
85 e7 = e5 + e6;
86 e8 = e7 + ro_i_jminus2;
87 e9 = e8 / 2;
88 e10 = e9 * u_i_j;
89 e11 = e10 * dy;
90 e12 = 3 * u_i_j;
91 e13 = -4 * u_i_jminus1;

```

```

92 e14 = e12 + e13;
93 e15 = e14 + u_i_jminus2;
94 e16 = e15 / 2;
95 e17 = e16 * ro_i_j;
96 e18 = e17 * dy;
97 e19 = 3 * ro_i_j;
98 e20 = e19 * u_i_j;
99 e21 = e20 * dy;
100 e22 = e21 / 2;
101 E1 = e4 + e11;
102 E2 = e18 + e22;
103 E = E1 + E2;
104
105
106 f1 = -3 * p_i_j;
107 f2 = 4 * p_i_jminus1_j;
108 f3 = f1 + f2;
109 f4 = f3 - p_i_jminus2_j;
110 f5 = f4 / 2;
111 f6 = -f5 * dx;
112 f7 = -4 * v_i_jminus1_j;
113 f8 = f7 + v_i_jminus2_j;
114 f9 = f8 / 2;
115 f10 = f9 * ro_i_j;
116 f11 = f10 * u_i_j;
117 f12 = f11 * dy;
118 F = f6 + f12;
119
120
121 v1 = E * E;
122 v2 = -4 * D;
123 v3 = v2 * F;
124 v4 = v1 + v3;
125 v5 = sqrt(v4);
126 v6 = 2 * D;
127 v7 = -E + v5;
128 v = v7 / v6;
129
130
131 end % of Momentum equation solution

```



APPENDIX C : Elaboration of Energy Equation

```
1 % Energy Equation Solution
2 % Calculates new energy (E)
3 function e = energy_solution(u_i_j , u_iminus2_j ,...
4     u_iminus1_j , v_i_j , v_i_jminus1 ,...
5     v_i_jminus2 , ro_i_j , ro_i_jminus1 ,...
6     ro_i_jminus2 , ro_iminus1_j ,...
7     ro_iminus2_j , E_iminus1_j , E_iminus2_j ,...
8     E_i_jminus1 , E_i_jminus2 , dx , dy)
9
10 n1_1 = -4 * E_iminus1_j;
11 n1_2 = n1_1 + E_iminus2_j;
12 n1_3 = n1_2 / 2;
13 n1_4 = n1_3 * dy;
14 n1_5 = n1_4 * ro_i_j;
15 n1_6 = n1_5 * u_i_j;
16 n1_7 = u_i_j * u_i_j * u_i_j;
17 n1_8 = n1_7 / 2;
18 n1_9 = 3 * ro_i_j;
19 n1_10 = -4 * ro_iminus1_j;
20 n1_11 = n1_9 + n1_10;
21 n1_12 = n1_11 + ro_iminus2_j;
22 n1_13 = n1_12 / 2;
23 n1_14 = n1_13 * dy;
24 n1_15 = n1_8 * n1_14;
25 n1_16 = u_i_j * u_i_j;
26 n1_17 = n1_16 * 3;
27 n1_18 = n1_17 * ro_i_j;
28 n1_19 = n1_18 / 2;
29 n1_20 = 3 * u_i_j;
30 n1_21 = -4 * u_iminus1_j;
31 n1_22 = n1_20 + n1_21;
32 n1_23 = n1_22 + u_iminus2_j;
33 n1_24 = n1_23 / 2;
34 n1_25 = n1_24 * dy;
35 n1_26 = n1_19 * n1_25;
36
37 numerator1_1 = -n1_6 - n1_15;
38 numarator1 = numerator1_1 - n1_26;
39
40 d1_1 = 3 * ro_i_j;
41 d1_2 = -4 * ro_iminus1_j;
42 d1_3 = d1_1 + d1_2;
43 d1_4 = d1_3 + ro_iminus2_j;
```

```

44 d1_5 = d1_4 / 2;
45 d1_6 = d1_5 * dy;
46 d1_7 = d1_6 * u_i_j;
47 d1_8 = 3 * u_i_j;
48 d1_9 = -4 * u_iminus1_j;
49 d1_10 = d1_8 + d1_9;
50 d1_11 = d1_10 + u_iminus2_j;
51 d1_12 = d1_11 / 2;
52 d1_13 = d1_12 * dy;
53 d1_14 = d1_13 * ro_i_j;
54 d1_15 = 3 * ro_i_j;
55 d1_16 = u_i_j * dy;
56 d1_17 = d1_15*d1_16;
57 d1_18 = d1_17 / 2;
58
59 denominator1_1 = d1_7 + d1_14;
60 denominator1 = denominator1_1 + d1_18;
61
62 term1 = numerator1 / denominator1;
63
64 n2_1 = -4 * E_i_jminus1;
65 n2_2 = n2_1 + E_i_jminus2;
66 n2_3 = n2_2 / 2;
67 n2_4 = n2_3 * dx;
68 n2_5 = n2_4 * ro_i_j;
69 n2_6 = n2_5 * v_i_j;
70 n2_7 = v_i_j * v_i_j * v_i_j;
71 n2_8 = n2_7 / 2;
72 n2_9 = 3 * ro_i_j;
73 n2_10 = -4 * ro_i_jminus1;
74 n2_11 = n2_9 + n2_10;
75 n2_12 = n2_11 + ro_i_jminus2;
76 n2_13 = n2_12 / 2;
77 n2_14 = n2_13 * dx;
78 n2_15 = n2_8 * n2_14;
79 n2_16 = v_i_j * v_i_j;
80 n2_17 = n2_16 * 3;
81 n2_18 = n2_17 * ro_i_j;
82 n2_19 = n2_18 / 2;
83 n2_20 = 3 * v_i_j;
84 n2_21 = -4 * v_i_jminus1;
85 n2_22 = n2_20 + n2_21;
86 n2_23 = n2_22 + v_i_jminus2;
87 n2_24 = n2_23 / 2;
88 n2_25 = n2_24 * dx;
89 n2_26 = n2_19 * n2_25;
90
91 numerator2_1 = -n2_6 - n2_15;

```

```

92 numerator2 = numerator2_1 - n2_26;
93
94 d2_1 = 3 * ro_i_j;
95 d2_2 = -4 * ro_i_jminus1;
96 d2_3 = d2_1 + d2_2;
97 d2_4 = d2_3 + ro_i_j;
98 d2_5 = d2_4 / 2;
99 d2_6 = d2_5 * dx;
100 d2_7 = d2_6 * v_i_j;
101 d2_8 = 3 * v_i_j;
102 d2_9 = -4 * v_i_jminus1;
103 d2_10 = d2_8 + d2_9;
104 d2_11 = d2_10 + v_i_jminus2;
105 d2_12 = d2_11 / 2;
106 d2_13 = d2_12 * dx;
107 d2_14 = d2_13 * ro_i_j;
108 d2_15 = 3 * ro_i_j * v_i_j * dx;
109 d2_16 = d2_15 / 2;
110
111 denominator2_1 = d2_7 + d2_14 ;
112 denominator2 = denominator2_1 + d2_16;
113
114 term2 = numerator2 / denominator2;
115
116 E=0.5*(term1+term2);
117
118 end % of Energy equation solution

```



APPENDIX D : Elaboration of Temperature and Pressure

```
1 % Temperature Solution
2 % Calculates new Temperature respect to energy
3 function T = temperature_solution(e_i_j ,Cp )
4 T = e_i_j /Cp ;
5 end % of Temperature solution

1 % Pressure Solution
2 % Calculates new pressure respect to density
3 % and temperature
4 function p= pressure_solution( ro_i_j ,R, T_i_j) %
5 p1 = ro_i_j *R;
6 p = p1*T_i_j;
7 end % of the solution of Pressure
```



CURRICULUM VITAE

Name SURNAME: Emine Elif Yiğit

EDUCATION:

- **B.Sc.:** 2021, Samsun University, Aeronautics and Astronautics Faculty, Aeronautics and Astronautics
- **M.Sc.:** 2024, Istanbul Technical University, Aviation Institute, Department of Defence Technologies

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2023-Present Research Assistant, Istanbul Technical University, Aviation Institute

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Yigit E.E.** and Yeniceri, R. (2024). FPGA Based Hardware Accelerator for Euler Equations with Finite Volume Method, *AIAA SCITECH 2024 Forum*, p.0044