Florijan Reichmann, BSc

Embedded System Design for a Time-of-Flight Ultrasonic Flowmeter

MASTER'S THESIS

to achieve the university degree of

Master of Science

Master's degree programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Mag.rer.nat. Dr.rer.nat. Alexander Bergmann

Institute of Electronic Sensor Systems

Co-Supervisor

Dipl.-Ing. Reinhard Klambauer, BSc

Graz, November 2019

EIDESSTATTLICHE ERKLÄRUNG
AFFIDAVIT

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.
The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Datum / Date	Unterschrift / Signature

Abstract

Ultrasonic technology has been increasingly used for flow measurements in industrial applications over the past decades. Today, a wide range of ultrasonic-based methods are available to obtain the mass flow rate of gases. The majority of those techniques uses the approach of time-of-flight (TOF) measurement - the TOF measurement is used to calculate the flow velocity of a medium, from which the (mass) flow rate can be deduced. The present master thesis examines details on the signal processing for TOF flow rate measurement and the practical implementation of a system capable of performing such measurements. The work consists of the following two parts.

The first part presents the evaluation of a novel signal processing method that is used to determine the TOF of a medium based on its echo energy signal. In order to benchmark this novel approach against established classical methods, a simulation framework for ultrasonic wave propagation was implemented from scratch using open source software. Outputs of the simulation were then analyzed in order to determine the accuracy of the novel approach compared to classical cross-correlation methods. Based on the results of this evaluation, it is concluded that the novel echo energy method has the potential to become a considerable alternative to established classical approaches.

The second part of the thesis presents the development and design of an embedded system for an ultrasonic flowmeter. The system is based on a Zynq-board running a Linux operating system, and builds on ARM cores combined with a field programmable gate array. The main focus was the development of a custom module for high speed sampling of ultrasonic waves along with receiving transducers. This module was developed with VHDL and extends available software to enable the transfer of sampled data over an Ethernet connection. The outcome of the design is an embedded system with high-speed and real-time sampling capabilities and high bandwidth data transfer.

Zusammenfassung

Ultraschalltechnologie wird in den letzten Jahrzenten verstärkt für Durchflussmessungen im industriellen Bereich eingesetzt. Heute existiert eine Vielzahl an ultraschallbasierten Methoden, mit welchen die Massenflussraten von Gasen ermittelt werden können. Die Mehrheit dieser Methoden beruht auf Laufzeitmessung (time-of-flight), aus welcher zunächst die Strömungsgeschwindigkeit eines Mediums berechnet und daraus wiederum die Massenflussrate abgeleitet werden kann.

Diese Masterarbeit befasst sich mit der Signalverarbeitung von auf Laufzeit basierten Durchflussmessungen sowie der praktischen Implementierung eines Durchflussmesssystems und gliedert sich in folgende zwei Teile.

Im ersten Teil wird eine innovative Signalverarbeitungsmethode evaluiert, die auf der Laufzeitmessung des Echo-Energie Signals beruht. Für die Bewertung und Gegenüberstellung zu gängigen Methoden wurde ein Open-Source-Software basiertes Simulationsmodell für Ultraschallwellenausbreitung entwickelt. Die Simulationsergebnisse wurden herangezogen, um diese neuartige Echo-Energie Methode mit der klassischen Kreuzkorrelationsmethode zu vergleichen. Aus den Resultaten geht hervor, dass diese innovative Methode durchaus Potenzial hat, eine Alternative darzustellen.

Im zweiten Teil werden Entwurf und Implementierung eines eingebetteten Systems für einen Ultraschall-Durchflussmesser präsentiert. Das System besteht aus einem Zynq-Board mit Linux-Betriebssystem. Der Fokus des praktischen Teils lag auf der Entwicklung eines individuell entworfenen Digitalmoduls, welches zur Hochgeschwindigkeits-Abtastung des Ultraschallsignals mit empfangenden Transducern eingesetzt wird. Das Modul wurde in VHDL entwickelt und erweitert bestehende Software um die Möglichkeit der Übertragung abgetasteter Daten via Ethernet-Schnittstelle. Das Resultat der Implementierung ist ein System, welches hohe Abtastraten sowie Datenübertragung mit hoher Bandbreite unterstützt und in Echtzeitsystemen eingesetzt werden kann.

Acknowledgement

I express my deepest gratitude to Prof. Alexander Bergmann, the head of the Institute of Electronic Sensor Systems, for the opportunity to carry out this thesis under his valuable guidance and providing me access to all facilities at his institute.
I express my sincere and whole hearted thanks, to my mentor Reinhard Klambauer, who supported me throughout the entire period of the thesis. Thanks for the highly qualified technical advices, the moral support and the encouragement to make this thesis happen.
I would like to express my sincere thanks to all institute members for the technical support and their inspiration for scientific thinking.
Finally, I would like to express my heart full thanks to my parents, all relatives and friends for their endless support, their sacrifices and continuous encouragement throughout my years of study.

This thesis is dedicated to my beloved parents, Irena and Gregor Reichmann.

Diplomsko nalogo posvečam svojim dragim staršem,
Ireni in Gregorju Reichmann.

Contents

CHAPTER 1

Introduction

1.1 Problem statement

At the Institute of Electronic Sensor Systems (IES) an alternative flow measurement principal based on ultrasonic beamforming is being developed. Therefore a laboratory setup was already present. The setup consists of a metal pipe with embedded ultrasonic transducers which are used to send and receive sounds in the ultrasonic region. The sending unit includes multiple transducers forming an array - this is a special designed 3D-printed structure to perform beamforming. The metal pipe has an intended cutout to mount this array. [1]
The basic idea behind the flow measurement is the following: the sending unit generates an ultrasonic beam; this beam is deflected by the flow-field in the pipe; the receiving transducers sense the beam intensity continuously; the properties of this sensed signal are used to deduce the flow velocity.
The heart of the system is a Xilinx Zynq 7020 System-on-Chip (SoC) board. The Zynq family is a hybrid architecture with two ARM Cortex-A9 CPUs and a field-programmable gate array (FPGA) integrated on a single chip. This Zynq board is connected to a custom developed carrier board with additional electronics extending the analog and digital capabilities. The carrier board is mainly responsible for the power supply, the signal generation for the transducers and the sampling of the beam with the receiving transducers in combination with analog-to-digital converters.
The Zynq board is running an embedded-Linux operating system (OS). The OS runs an

user space application communicating with the logic of the FPGA. The application acts as a server and is remotely controlled by a host application over Ethernet.

At that time, only the signal generation for the sending unit was implemented. The logic for the sensing with the receiving transducers had to be implemented. The goal of the master thesis is to design and implement an embedded system for the receiving path. This includes digital design for an FPGA-module in VHDL to control the sampling with the analog-to-digital converters which are sampling the signals of the receiving transducers. Additionally, kernel space drivers and the extension of the user and host application for the transfer of the sensed data is required.
The expected data rate on the receiving path is greater than 30 MBps. This requires a design for speed and low latency of the digital part as well as efficient transfers on the Ethernet path.

To get familiar with the mathematics and physics of a flow measurement system, a simulation for wave propagation is implemented in the first place. With the data obtained by the simulation, two signal processing techniques for flow velocity calculation are evaluated. The classical cross-correlation signal processing technique is compared to a novel approach based on the echo energy signal.

CHAPTER 2

Theoretical Foundations

In order to lay the foundations for this work, this chapter provides the theoretical basics for the practical part. It covers the topics: ultrasonics, flowmeters, hardware architectures, and numerical methods for wave propagation simulation.

2.1 Ultrasonics

In physics, sound is a pressure disturbance propagating through a medium like solids, liquids or gases as longitudinal and transverse waves. Ultrasonics or ultrasounds are part of acoustics with frequencies higher than the range of human perception. The American National Standards Institute (ANSI) defines ultrasound as "sound at frequencies greater than 20 kHz" [2].
Ultrasonic ranges from the human inaudible frequency starting at 20 kHz to 1 THz where it goes over into the hypersonic range as illustrated in figure 2.1.

2.1.1 Introduction

Ultrasonic is present in the animal world since primitive times. Marine mammals like dolphins, whales and seals developed highly sophisticated organs to transmit and receive ultrasonics for navigation, hunting and communication. The creatures best-known for ultrasonic ashore are bats, being able to fly in the dark using ultrasonic echolocation to avoid obstacles.
From a technology point of view, research on ultrasonics started during World War I.

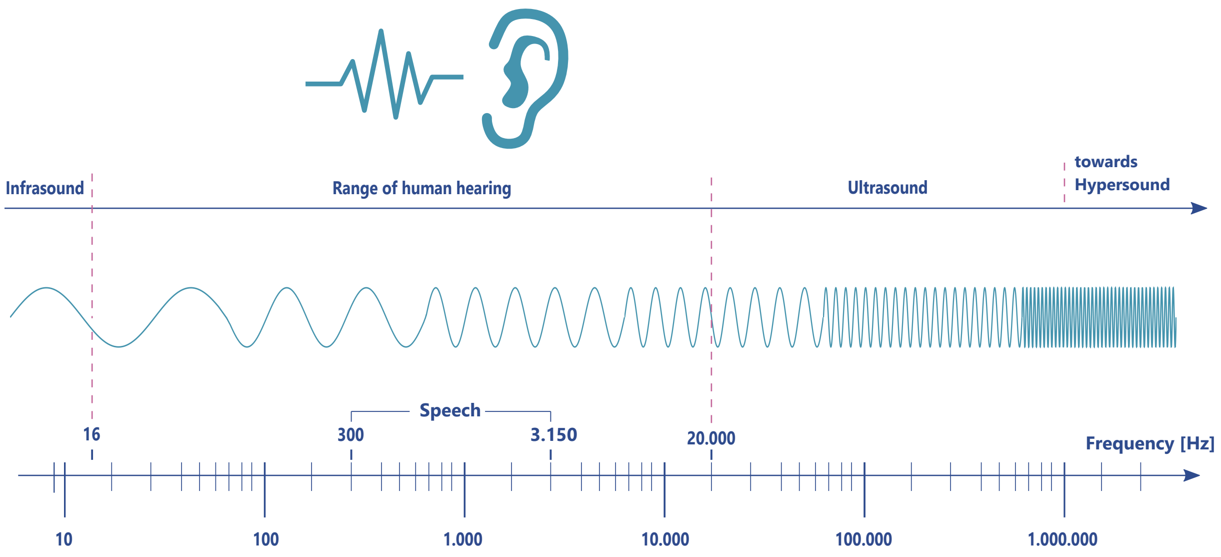


Figure 2.1: Classification of acoustic waves - starting at low frequencies, called infrasound, up to the hypersound region with frequencies above 1 THz.

The first echo-ranging application using ultrasonics was present by 1914 [3]. The major break-through in ultrasonics was the invention of a quartz based ultrasonic transducer by the French physicist Langevin in 1917 [4]. The main purpose of his work was the detection of enemy submarines. This was the first application in the scope of high frequency acoustics. The usage of ultrasonics under water to locate objects is often referred to as sound navigation ranging (SONAR).
Nowadays ultrasonic has a broad field of application in all areas, for example, industrial cleaning, welding, park distance control, navigation of ships and medical treatment - just to list a few.

2.1.2 Applications

Subsequently, some well known applications from different areas are introduced. Starting with underwater applications, the section will cover utilisations in the medical area as well as the industrial usage of ultrasonics.

2.1.2.1 SONAR - Sound Navigation Ranging

A historical moment leading to the first patents for obstacle avoidance using echo-ranging was the Titanic's collision with the iceberg in 1912. The first functional echo-ranging finder is dated back to the year 1914, patented by the Canadian R. Fessenden. Fessenden was using an acoustic transmitter-receiver with frequencies below the ultrasonic range

to detect icebergs several miles away. Fessenden was hired for the anti-submarine team of the Royal Navy. His device was used for submarine detection in first place, but was lacking accuracy. After consulting Langevin, the team came to the conclusion that his quartz transducer in the ultrasonic range is providing the solution. [3]
Beside the atrocities in the World War II, however, it brought the next remarkable advance using sound underwater. New sensors for torpedo and mine detection and the science of new stealth-mode capable sound materials avoiding reflections of SONAR as good as possible emerged.
Nowadays many civil applications exist. Examples for the most common underwater applications are navigation of ships in unknown sea-bed, wreck detection with 3D reconstruction, side-scan sonar for permanent inspection of the seafloor, mapping of sunken cities, laying of undersea cables, and locating large shoal of fish [5].

2.1.2.2 Medical and Biological

In present times, ultrasonic has become indispensable in the healthcare sector. In medicine, the usage of ultrasound is divided into diagnosis and treatment. Diagnosis refers mostly to medical imaging, which started in 1940s. For diagnosis lower power levels are in use to keep the structure of the tissue unchanged. The most famous and essential routine in diagnostics is the fetal imaging during gravidity.
The destructive ability of high power ultrasound was already revealed by experiments of Langevin by unintentionally killing smaller fish that were entering the high-intense ultrasonic beam which was used during experiments for the submarine detection. Such high intensity ultrasonics have an intense interaction with human tissue. The different interaction on healthy and diseased tissue is used for treatment. For therapy, intermediate intensities about 2-10 $\mathrm{W\,cm^{-2}}$ and for surgery intensities as high as 10^4 $\mathrm{W\,cm^{-2}}$ are used to cause a temporary or permanent change on the biological tissue [6].

2.1.2.3 Industrial Usage

Industrial applications of ultrasonics offer almost unlimited possibilities. The industrial sector is divided into two main parts - processing, and measurement and control. Typical processing applications are cleaning, metal forming, sonochemistry, soldering, and welding. Representatives on the measurement and control side are temperature measurement, flow measurement, echolocation of defects, material characterization and mi-

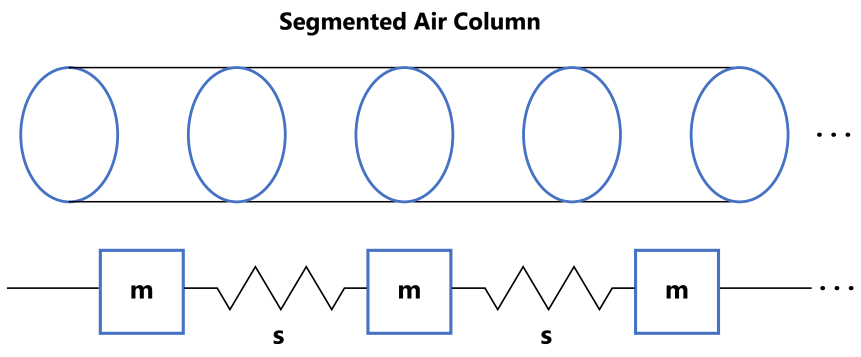


Figure 2.2: Mechanical mass (m) spring (s) analogy for the propagation of sound in air.

croscopy, whereas the last three are also known as nondestructive testing methods.

In the previous sections the history of ultrasonics and its current applications are briefly presented. The focus of the next sections will be on the physical properties of sound in general and on the technical possibilities to generate and detect ultrasonics.

2.1.3 Propagation in Gases

Ultrasonic is propagating like any other sound wave in a media and share many similarities with the propagation of electromagnetic radiation. An essential difference lies in the fact that ultrasonic waves need a media for propagation and therefore cannot travel through vacuum.

As a starting point for the propagation explanation, we consider an analogy with a mass hooked to a spring, forming a simple oscillator. By applying a force on the mass, either pulling or compressing, a harmonic oscillation is observable. The restoring force is given by Hooke's law with the opposite direction of the original displacement. An acoustic wave can be considered as a connection of infinite number of mass-spring elements in series shown in figure 2.2. As soon a disturbance provokes one of the mass elements, the energy is transfered to the next one, and so on propagating over the complete chain. There is no unison movement since mass has inertia and as such follows Newton's laws of motion. Generally, the oscillation will be damped due to friction between molecules and resistive influence.

The force on a mass is the analogy to a disturbance in a pressure field of a gas. The disturbance will propagate until it is fully damped. The speed of the disturbance moving through the medium is known as the velocity of sound and depends predominantly on the temperature - the exact relation is given in section 2.1.3.1.

2.1.3.1 Thermodynamics of Sound in Gases

Starting from a given gas-mass M, its physical properties can be described by its volume V_G, its density ρ_G, its pressure p_G and its temperature T_G. An increase in temperature causes an increase in pressure $p_G \approx T_G$ at a constant volume. The pressure in the gas is inverse proportional to its volume $p_G \approx 1/V_G$.

The state of matter under certain physical conditions is described by the equation of state of the general form

$$f(p, V, T) = 0 \tag{2.1}$$

where p is the absolute pressure, V its volume and T its absolute temperature. Its derived form is building the base for the determination of the velocity of sound in gases. The ideal gas law (equation of state of an ideal gas), is expressed by

$$pV_G = \frac{N}{N_{mol}} RT_G \tag{2.2}$$

where N is the number of molecules in V_G, N_{mol} is the number of molecules in a gram-molecular weight and R is the gas constant.

Combining equation 2.2 with the velocity of sound c in a gas, given by

$$c^2 = \gamma \frac{\partial p}{\partial \rho_G} \tag{2.3}$$

results in

$$c^2 = \gamma \frac{RT_G}{M} \tag{2.4}$$

with

$$\gamma = \frac{c_p}{c_v} \tag{2.5}$$

specifying the ratio between the specific heat c_p under constant pressure (isobaric) and the specific heat c_v at constant volume (isochoric). Obviously, the velocity of sound of a hypothetical ideal gas only depends on its physical composition and its temperature. At room temperature (20 °C), c is approximately 343 m/s in dry air.

Generally, the energy transmitted by sound or ultrasonic in form of stress waves has adiabatic nature, since the variation in pressure and resulting volume change is happening so quickly, that the heat flow cannot follow the dynamic process so rapidly.

Like any other wave, an ultrasonic wave undergoes reflection, refraction, scattering, diffraction, interference and absorption when it reaches another media or hits an object.

Unlike solids, liquids and gases are capable of transmitting longitudinal waves only. For longitudinal waves, the particle movement and propagation is in the same direction.
In principle, sound waves do not change their signal-shape when propagating in gases. Such a propagation is called non-dispersive. Voice communication would be unfeasible if this would not be the case - imagine changing frequencies and signal shapes during a conversation. For solids, like rods or plates, bending waves are the decisive components for the wave propagation. The wave length of the bending waves is frequency dependent. As such, single components propagate with different propagation velocities konwn as dispersive behaviour. [7]

2.1.4 Generation and Detection

Ultrasonic transducers are devices able to generate and detect ultrasonic energy. Speaking in general terms, a transducer converts one type of energy to another.
The first condenser based transducer was driven by a Poulsen arc at about 100 kHz back in 1915. In the following years various technical enhancements arose to manufacture ultrasonic transducers using all kind of suitable materials. The most commonly used devices are based on the piezoelectric effect. Piezoelectric transducers are used over the whole ultrasonic frequency range at various intensity levels. The selection criteria for piezoelectric materials are chosen on a number of different characteristics such as

- Piezoelectric performance
- Dielectric properties
- Elastic properties
- Stability

A list of properties on various materials can be found in [8].
As soon a piezoelectric element is stimulated by an electric field it results in a mechanical deformation of the crystal, called the inverse piezoelectric effect. An applied external alternating electric source causes a vibration with a frequency proportional to the source. This compresses the gas in front of the element which results in a longitudinal (ultrasonic) sound wave. Beside the role as an actuator, the piezoelectric element can act as a sensor

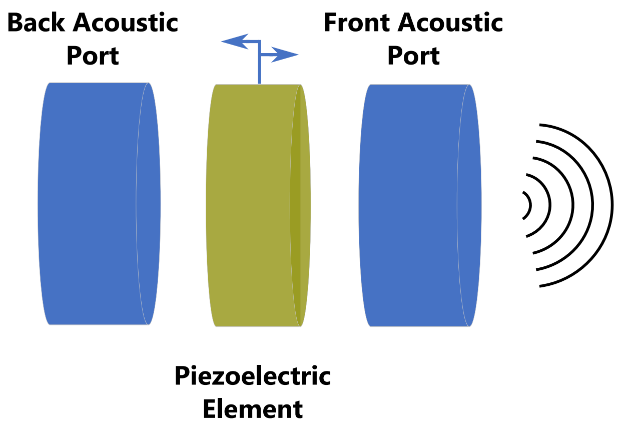


Figure 2.3: Three-port model of a transducer.

as well - a physical force causes a directional deformation of the element. This forms microscopic dipoles inside the element which leads to a measurable electric field.

The basic structure of a transducer can be thought as a three-port model shown in figure 2.3, a simplification of the more complex KLM model, proposed by Krimholtz [9]. The three-port model consists of two mechanical ports, the back- and the front-acoustic port, and an electrical port, the piezoelectric element. The front-acoustic port is known as acoustic matching layer, mainly responsible for matching acoustic impedances between the transducer and the propagating medium, and minimizing scattering effects by using special materials and proper sizing, both being essential aspects for the performance of the transducer. Inside the transducer an acoustic impedance mismatch between the piezoelectric element and the gas (air) is present. Due to the mismatch, the reflected wave may reverberate for a long period, known as ringing effect, badly influencing imaging and other ultrasonic applications. To avoid a ringing, the back-acoustic layer absorbs or at least highly damps the reflected wave. The ring time depends on the design of the transducer and the type of electrical pulse driving the piezoelectric element.
The electrical port matches the electrical characteristics of the transducer to the external circuits. Generally, the transducer input impedance should be real-valued. An imaginary part would cause an unwanted phase shift. To achieve maximum power transmission, the input impedance should match the source (usually $50\,\Omega$) [8].
Typical applications based on piezoelectric transducers are flow sensors and meters, medical diagnostics, fish detection, cleaning, park distance control, and none destructive testing.
Beside the classical piezoelectric transducers, there are a couple of different types listed

below

- Magnetostrictive transducers
- Electromagnetic acoustic transducers
- Capacitive transducers
- Pneumatic transducers
- Hydraulic transducers
- Mechanical transducers
- Thermal transducers
- Capacitive micromachined ultrasonic transducers (CMUT)

Their working principle can be found in [10], [11] and [12].

Another interesting transducer type is the laser ultrasound transducer, which is using the photoacoustic effect to produce high frequency, broadband, and high intensity ultrasonic waves. A material absorbs pulsed laser energy causing local thermal expansion. The expansion of the absorber generates an ultrasonic wave. The structure of the absorber determines the shape and beam of the generated ultrasonic wave. Figure 2.4 shows the resulting beam for a flat and concaved shaped absorber.
The classic piezoelectric high intensity focused ultrasound has a large aperture, large focusing volume and low frequency that hinder the use in precision therapies. In contrast, the laser based transducers can operate at high frequencies and single handheld probes can be miniaturized to several hundred microns by using optical fibers, being perfectly suited for biomedical applications. [13]

2.1.5 Arrays

Many conventional ultrasonic applications use a single transducer with a single beam propagating along one axis at a specific angle. The idea behind an array is the usage of many identical elements next to each other with synchronized phase. Multiple send out wave fronts will interfere, creating one superpositioned wavefront controlled by time-delayed synchronized control signals. This electrical steering capability can change the

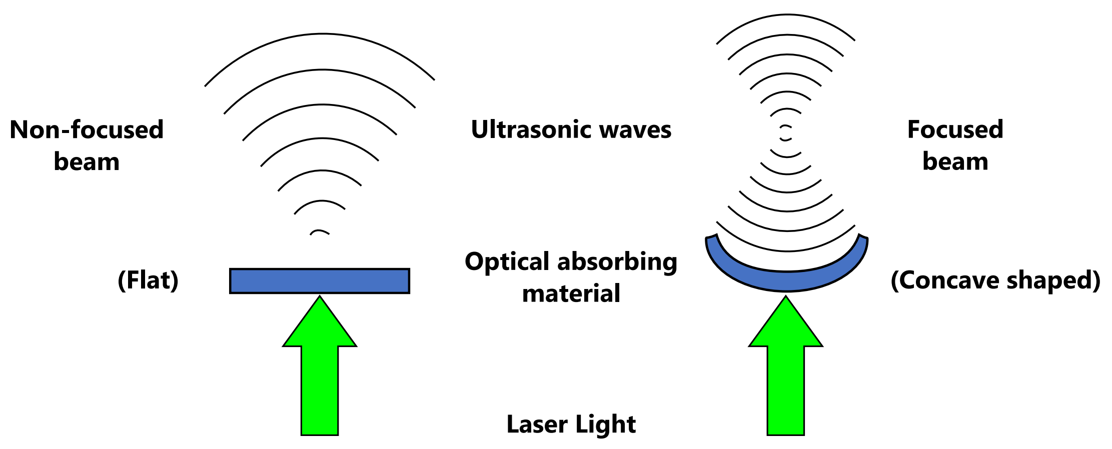


Figure 2.4: Concept of non-focused and focused laser transducer. The shape of the absorbing material determines the beam-form.

angle, the focal distance and the focal spot size of the ultrasonic beam. Figure 2.5 shows the capability of the array to change the focus by using control signals with parabolic-shaped time-delay. The steering capability is shown in figure 2.6, achieved by a linear time delay. [14]

The main advantage of ultrasonic arrays is the steering capability without the need of any mechanical components. As such, the maintenance costs and manufacturing complexity can be reduced.

In [1] such a phased array with up to sixteen ultrasonic transducers is in use. A special feature of this solution is the usage of custom designed acoustic wave guides. This enables the sensitive transducers to be mounted further away from the measured medium which is beneficial for the usage in harsh environments.

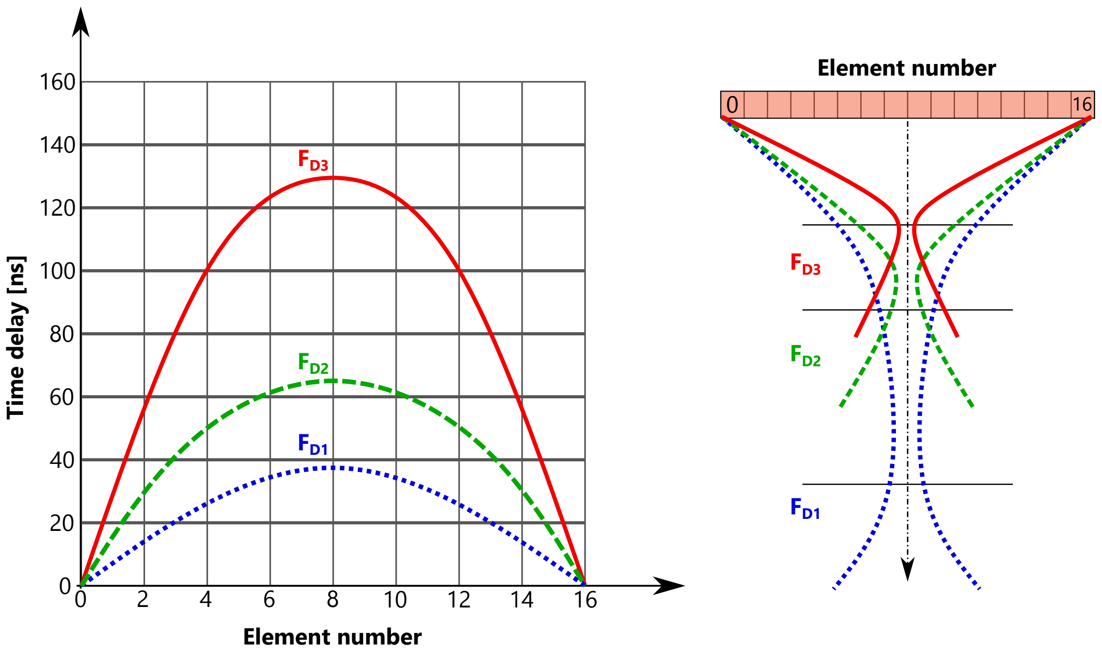


Figure 2.5: Beam focus capability of a transducer array. The time-delay of the excitation signal for the single transducers controls the resulting focus point. Adapted from [14].

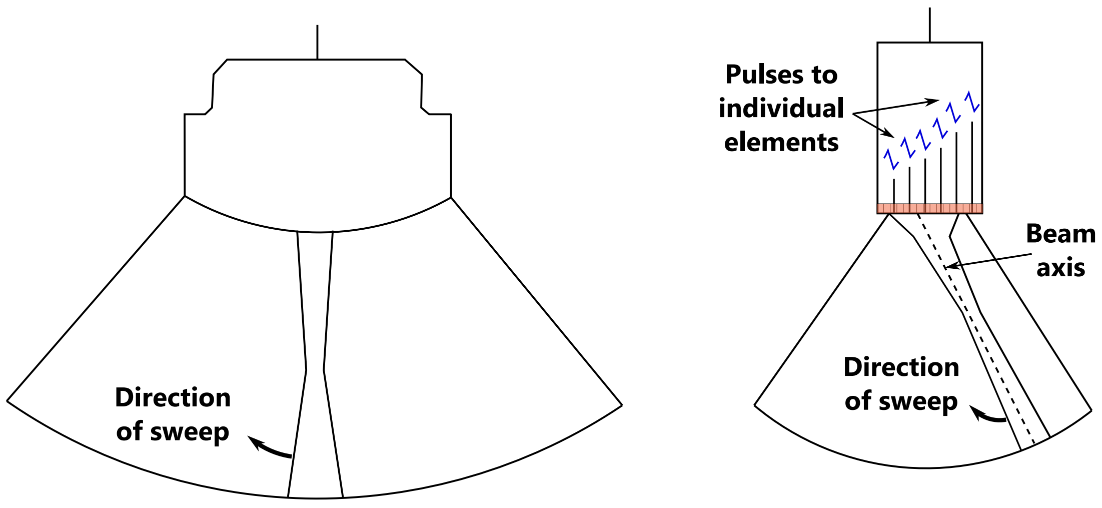


Figure 2.6: Beam steering capability of a transducer array. Left sketch shows a curved linear array with sector scan. Right sketch represents a phased array with beam steering.

2.2 Fluid Mechanics

2.2.1 Flow in a Pipe

The flow pattern in a pipe depends on various factors. An essential representative in fluid mechanics is the Reynolds number (Re), a dimensionless quantity predicting the flow pattern given by

$$Re = v_f \frac{\rho_G D_p}{\mu} \tag{2.6}$$

where μ is the dynamic viscosity, v_f the velocity of the medium in the pipe and D_p being the pipe diameter.

At low Re (< 2000), a laminar flow will settle along the pipe with a flow-motion parallel to the axis of the pipe and a no-slip condition at the pipe wall shown in figure 2.7 (a-b). The no-slip condition is responsible for the breakdown of the velocity to zero at the fluid-pipe boundaries, since the adhesive forces close to the pipe-wall are greater than the cohesive one. Characteristically for a laminar flow is a parabolic-shaped velocity-profile with maximum velocity in the centre, and zero flow at the walls. At higher Re, reverse current created by swirls is causing the fast velocity in the centre being combined with the slow velocity at the pipe walls, resulting in a flatter shape, known as turbulent profile. Figure 2.7c.) demonstrates profile types for different Re numbers. [15]

Obviously, v_f is expressing a mean velocity only, since the flow profile is non-uniform across D_p.

2.2.2 Flow Rate

The flow rate defines a time dependent expression. To allow a quantitative determination of the flow rate the density is introduced beforehand, being

$$\rho = \frac{m}{V} \tag{2.7}$$

defined as the ratio of mass m to volume V of matter [$\mathrm{kg\,m^{-3}}$]. With mass and volume, a separation into mass flow rate q_m and volume flow rate q_v is made

$$q_m = \frac{m}{t} \quad [\mathrm{kg\,s^{-1}}] \tag{2.8}$$

$$q_v = \frac{V}{t} \quad [\mathrm{m^3\,s^{-1}}] \tag{2.9}$$

Although measuring the mass flow rate is preferable, since its value is independent of temperature, density, conductivity, viscosity and pressure, it is technically harder to achieve

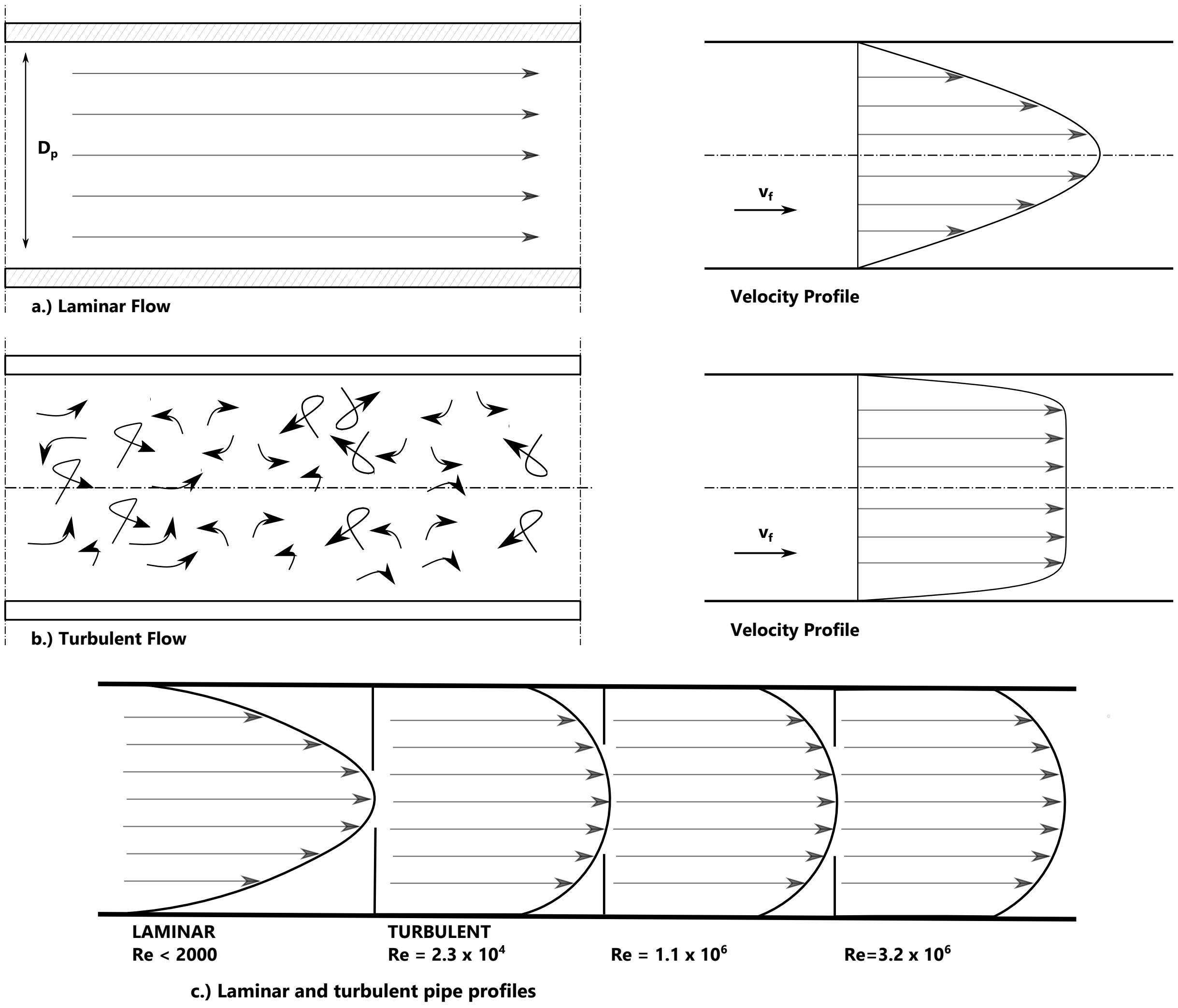


Figure 2.7: Figure (a-b) illustrates laminar and turbulent flow and velocity profile. Figure (c) shows pipe profiles for different Reynolds numbers.

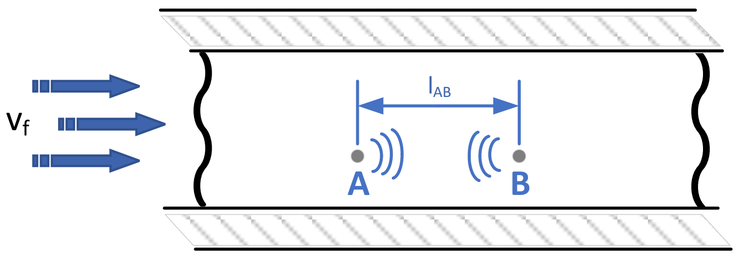


Figure 2.8: Two point sources A and B under the influence of a moving medium.

compared to the commonly used measurement of the volume flow. Generally, the tendency of the industry is towards direct mass measurement, since much more process parameters can be derived from the mass flow compared to the volume flow.

For a pipe with a diameter D_p, the volume flow rate is

$$q_v = v_f \frac{{D_p}^2 \pi}{4} \tag{2.10}$$

2.2.3 Sound Path in moving Media

Figure 2.8 illustrates two fixed point sources A and B separated by the distance l_{AB} under the influence of a moving gas with the mean velocity v_f. A sound impulse send from A travels with the total accumulated velocity of $c + v_f$ and takes time t_A to reach point B, given by

$$t_A = \frac{l_{AB}}{c + v_f} \tag{2.11}$$

An impulse from B needs to combat against the flowing medium and is slowed down, which results in a travel time t_B to reach A

$$t_B = \frac{l_{AB}}{c - v_f} \tag{2.12}$$

Assuming c is constant, t_A and t_B can be combined to extract v_f as

$$v_f = \frac{l_{AB}}{2} \left(\frac{1}{t_A} - \frac{1}{t_B} \right) \tag{2.13}$$

2.3 Flowmeters

A flowmeter is an instrument used to measure the volume or mass flow of a liquid or gas in a pipe. There is no such thing like an universal flowmeter being applicable for all situations. Each specific application requires the selection of a proper flowmeter type, whereat each technology has its own trade-offs.

2.3.1 Flow measurement Techniques

The process specifications will give the decision-making power to choose one specific, under the numerous available, flowmeter technologies. The expected flow rate, the fluid temperature, the flow mode, the desired accuracy, the installation requirements, and the expected pressure are only some of the criteria for the selection decision. A detailed list of selection criteria with suitable flowmeters can be found at [15].
Subsequently, some of the commonly used flowmeter technologies are presented.

2.3.1.1 Differential Pressure Flowmeters

The basis of a differential pressure flowmeter (DPF) is a pressure drop caused by artificial obstacles, representing an orifice that is placed in the cross section of the pipe shown in figure 2.9. A pressure tapping measures the inlet flow (usually turbulent) at a specific distance in front of the orifice. The measured pressure corresponds to the static pressure. At the orifice the flow converges and causes recirculation eddies on both sides of the restricted section. The vortex narrows down the flow to a size smaller than the hole, the so called vena contracta. Another pressure tapping measures the downstream pressure across the vena contracta. A Pressure transducer, a manometer or a Bourdon tube can be used to measure the differential pressure and deduce the flow.
The DPF is widely used for measuring liquids and gases even at high temperature and pressure. Detailed equations and further explanations can be found in [15] and [16].

2.3.1.2 Thermal Flowmeters

Predominantly there are two different types of thermal mass flowmeters (TMFMs). The capillary thermal mass flowmeter (CTMF) on the one hand; the in-line thermal mass flowmeter (ITMF) on the other hand. The former is used for flow measurements of gases and liquids. Although there are various configurations for the gas type of CTMFs, all of them follow the same principle. The gas flows through a pipe with a diameter-size which is sufficient enough to ensure laminar flow. Embedded into the pipe are temperature-sensors between a heating system. As the gas passes the heater, its temperature rises and

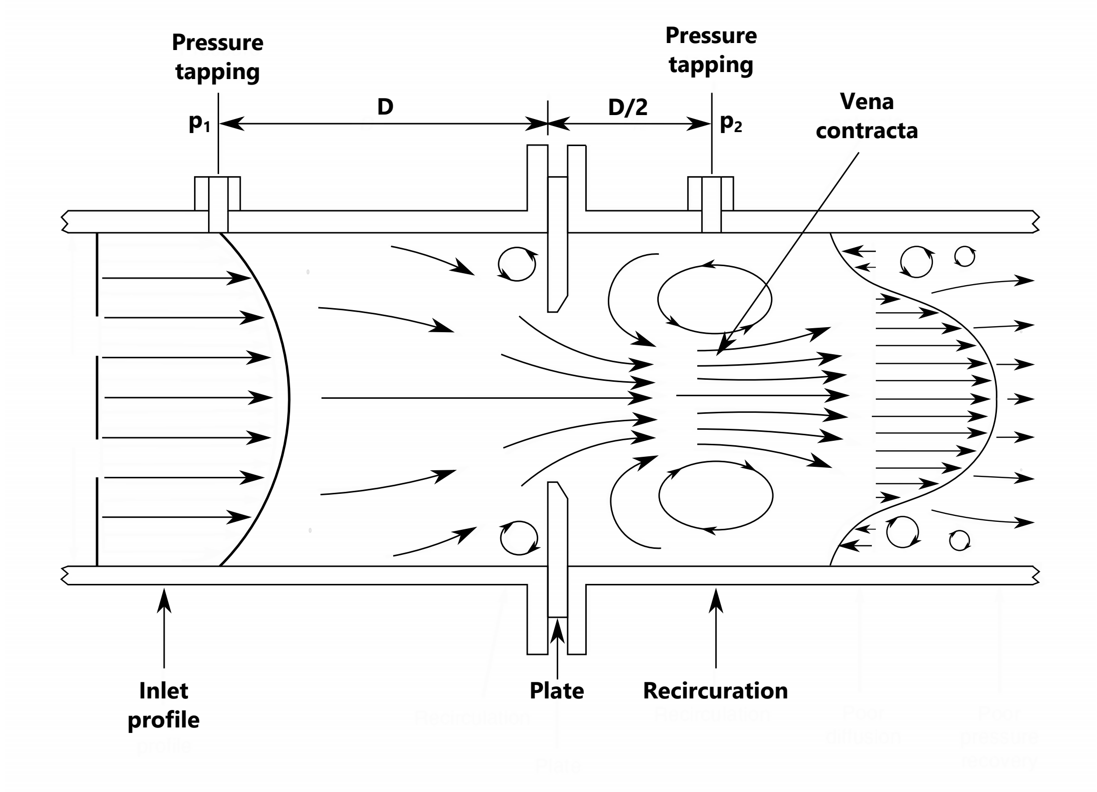


Figure 2.9: Sketch of a differential pressure flowmeter type. Plates narrow down the diameter resulting in a pressure drop. Adapted from [15].

brings the heat towards the the downstream temperature sensor. The difference between an additionally mounted upstream temperature sensor and the downstream sensor is used to calculate the mass flow. The temperature difference increments with increasing flow rate. [17]

Another approach is to heat the gas up to a constant temperature while measuring the current driving the heater necessary to keep the gas at a specific temperature. As the flow increases, the current needs adjustment to reach the same temperature. The current needed to keep a stable temperature is proportional to the mass flow rate.

ITMFs have an alternative design concept. Two thermal sensors are placed into the flow: one measures the reference temperature of the gas; the other is kept at a specific temperature above the reference temperature. The amount of heat required to keep the second sensor above the first one is proportional to the mass flow rate.

A few silicon based single chip solutions TMFMs are available. Smaller footprint, reduced costs and low power consumption are the main benefits of the silicon based devices compared to the classical TMFMs. The packaging of the chip (bonding together with gas in- and outlets) and the installation in the flow system are technical challenging.

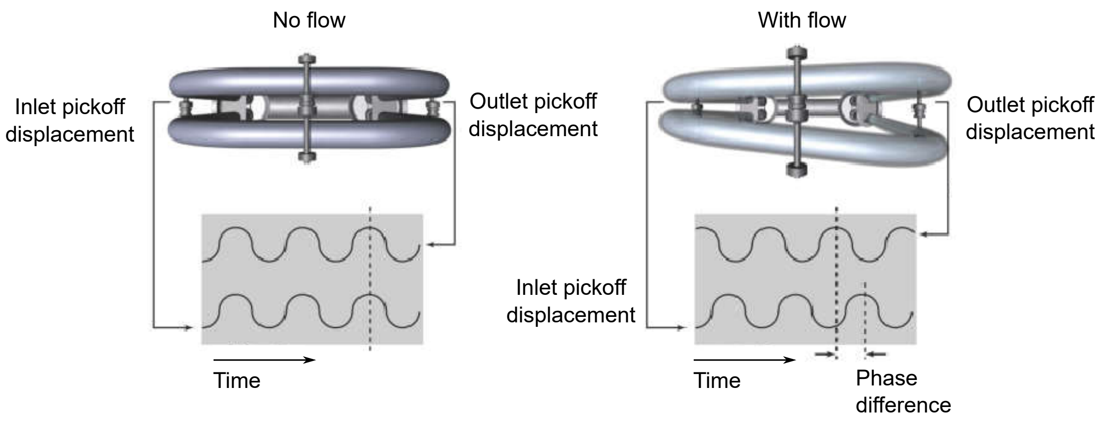

No flow
Inlet pickoff displacement
Outlet pickoff displacement
Time
With flow
Outlet pickoff displacement
Inlet pickoff displacement
Time
Phase difference

Figure 2.10: Phase shift measurement without flow (left figure) and with flow (right figure). Modified after [19].

Furthermore, the chips are not suited for liquid flows and aggressive gases. [18]

2.3.1.3 Coriolis Mass Flowmeters

The French scientist Gaspard Gustave de Coriolis set up the physical basis for this kind of flowmeter back in the nineteenth century. Li and Lee proposed the first flowmeter based on this physical property back in 1953.
When mass flows in a flexible tube, Coriolis forces are bending and twisting the tube. The small displacements are measured by two sensors at the inlet and outlet section. If there is no mass flow, the displacement measured by both sensors is equal and the measured phase shift is zero - show on the left-hand side in figure 2.10. In case of a mass flow, the phase shift in the measurement signals is a direct measure of how much gas or liquid is flowing through the pipe - show on the right-hand side in figure 2.10. Beside the mass flow, the sensors also measure the swing frequency of the tube. This oscillation, the so called resonance frequency depends on the tube geometry, the mass of the flowing media, and the material properties. It can be used to deduce the density of the fluid. The simultaneous measurement of flow and density gives several opportunities for quality control in industrial applications. The multi-variable measuring, very high accuracy ($\pm$ 0.15 % of rate) and wear free construction are making them a favourite choice for various flow measurement applications. High initial costs, installation limits and limited material selections for wetted parts can be listed as disadvantages. [15], [16]

2.3.2 Ultrasonic Flowmeters

According to [20], a German patent (1928) shows the first usage of an acoustic contrapropagating flow measuring device in a pipe. In the 1950s the the usage of transducers to measure flow velocity became popular. In 1955, Kritz [21] presented techniques to compensate different flow profiles and the mathematical background for the computation of the mass flow rate. Satomura showed the usage of ultrasonic Doppler method in 1956 and together with Kaneko build the first transcutaneous noninvasive Doppler flowmeter in 1958 for medical applications [22].

It lasted until the 1990s, till there was a broader acceptance of ultrasonic based flow meters for industry usage. The technology before was lacking performance and missing satisfying accuracy especially for natural gas transfer metering, where an accuracy of 0.5 % is targeted.

Versatility, improved accuracy, high precision transit time measuring and simple usability are some of the technical aspects making ultrasonic flowmeters (UFMs) a success story on the global flowmeter market. According to the IMARC group, the market for ultrasonic flowmeters breached the US$ one billion mark in 2018, with a future outlook to reach US$ 1.9 Billion by 2024. [23]

2.3.2.1 Measurement principles

Overall, there are two basic measurement principles distinguishable for UFMs:

1. Transit time method
2. Doppler method

Transit Time method The principle of the transit time method (TTM) is illustrated in figure 2.11. Two transducers, T_1 and T_2 are placed diagonal within an angle Φ to each other in a pipe. In the first step, T_1 acts as a sender of an ultrasonic impulse, while T_2 remains in a receiving state. The time needed by the impulse to reach T_2 is the downstream-time t_d. In the second step, the operation mode of the transducers switches and the time needed by the impulse to travel from T_2 to T_1 is measured, known as the upstream-time t_u.

Equation 2.13 acts as a basis for the calculation of the velocity of the medium, whereas

t_A equals to t_u and t_B equals to t_d, respectively. Under consideration of angle Φ, t_d and t_u becomes:

$$t_d = \frac{L}{c + v_f cos\left(\Phi\right)} \tag{2.14}$$

$$t_u = \frac{L}{c - v_f cos\left(\Phi\right)}. \tag{2.15}$$

The difference in the time-of-flight is $\Delta t = t_u - t_d$:

$$\Delta t = \frac{L}{c - v_f cos\left(\Phi\right)} - \frac{L}{c + v_f cos\left(\Phi\right)} \tag{2.16}$$

Solving equation 2.16 for v_f results in a quadratic equation:

$$\Delta t \left(\frac{v_f}{c}\right)^2 cos^2\left(\Phi\right) + \frac{2Lv_f}{c^2} cos\left(\Phi\right) - \Delta t = 0 \tag{2.17}$$

For small Mach numbers ($v_f \ll c$), the quadratic term can be neglected and v_f results in:

$$v_f = \frac{\Delta t c^2}{2L cos\left(\Phi\right)} \tag{2.18}$$

It is shown in [15] that the mean time t_m (mean of the current transit time in each direction) squared is $t_m^2 \approx t_u t_d$ and hence, c^2 can be replaced by $L^2/t_u t_d$. Finally, the equation to determine the mean flow velocity of a medium in a pipe expresses to:

$$v_f = \frac{L}{2cos\left(\Phi\right)} \left(\frac{1}{t_d} - \frac{1}{t_u}\right) \tag{2.19}$$

The distance L can be determined under measuring the time-of-flight (TOF) under zero flow and a constant c, which is known for a defined system state.

Taking into account the deviation caused by production and installation in the field, the volume flow from equation 2.10 can be corrected to:

$$q_v = K \cdot v_f \frac{{D_p}^2 \pi}{4} + dev \tag{2.20}$$

where K is the correction factor for imprecision at installation and dev is the system deviation. Both factors can be obtained by calibration under different flow rates [24].

For laminar flow, the average value calculated will be two-thirds of the maximum velocity which is in the centre of the pipe. For turbulent flow, the Reynolds number needs to be

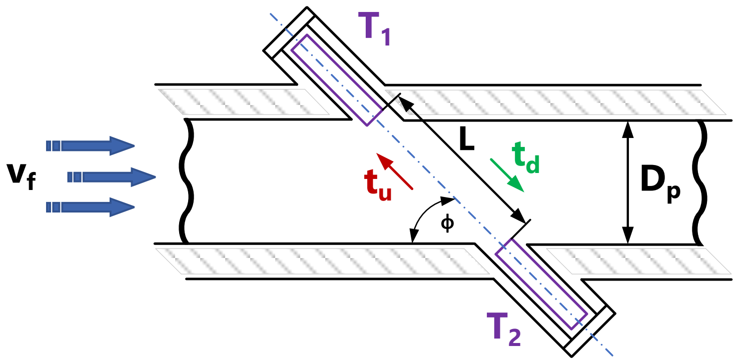


Figure 2.11: Arrangement of the transducers T_1 and T_2 for the transit time method.

taken into account. Thus, equation 2.19 needs an additional correction factor to consider the Reynolds number.

Doppler method The doppler method (DM) relies on the effect of the variation of the frequency when sound is reflected or emitted from a moving object relative to the observer, known as the the Doppler shift. In contrast to the TTM, the doppler method can operate with a single transducer block T_B (sending and receiving transducer in one housing). Alternatively, the transducer can also be placed in a separate block.
The measuring medium needs to have impurities enclosed (particles or bubbles) to make the doppler method working. Without these impurities the emitted wave is reflected from non-moving objects and does not undergo any change in frequency. If, however, the wave is reflected by one of the moving impurities under the angle α, a frequency shift Δf is observed:

$$\Delta f = 2 f_T cos\left(\alpha\right) \frac{v_f}{c} \tag{2.21}$$

where f_T is the transmitter frequency. To eliminate the dependency of c, which is heavily affecting the Doppler shift, a static inlet section for a sound path at the transmitter can be provided. In [16], the transmitter is casted in resin, which is forming a defined sound path. In doing so, Snell's law is applicable:

$$\frac{cos\,\alpha}{c} = \frac{cos\,\beta}{c_v}. \tag{2.22}$$

where c_v is the defined velocity in the resin.

Equation 2.21 can be rewritten to:

$$v_f = \frac{c_v}{2 f_T cos\left(\beta\right)} \Delta f = const \cdot \Delta f \tag{2.23}$$

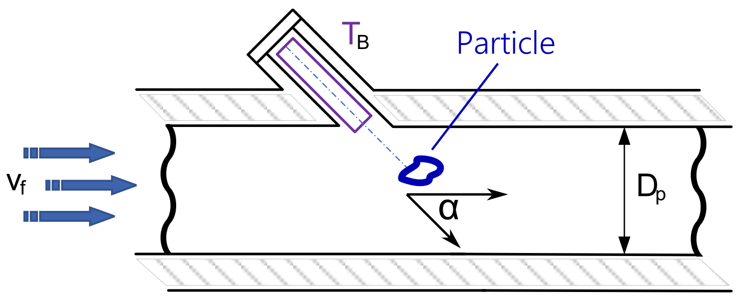


Figure 2.12: Doppler flowmeter with one possible arrangement of the transducers (single block T_B). The frequency of the reflected or emitted sound wave from the particle is changing - Doppler shift.

The term $c_v/cos(\beta)$ can be determined beforehand and as such expressed as const. The calculation of v_f becomes independent of the Doppler shift dependency to c.

Summary The TTM is the favoured type of the two available, because of high accuracy (in calibrated state $< 0.1\,\%$), fast measurement intervals and the lack of the need of moving particles in the media [25].
A TTM flowmeter requires the pipe to be fully filled to measure correct volumetric flow. This is not required for a DM flowmeter, which can still report the flow velocity, as long mounted below the liquid level, even if the pipe is not fully filled. For changing states of the media hybrid solutions exists which are combining the transit time and the doppler method.

2.3.2.2 Installation

There are three different installation types for ultrasonic flowmeters - namely clamp-on, inline and insertion, they are shown in figure 2.14. As the name implies, the clamp-on type uses a pair of transducers which are clamped directly on the outside of the pipe. The transducers are mounted at a specified distance. Figure 2.13 shows a possible arrangement of the transducers. Obviously, the ultrasonic wave is refracted several times - originating at the transducer, the wave is refracted as it passes the pipe-wall and also at the transition to the gas flowing inside the pipe. At the facing pipe-wall it is reflected and refracted back through the pipe-wall to the receiving transducer. Snell's law is applicable for the calculation of the refraction, implying that the material properties (fluid and pipe) need to be known. The flow velocity is proportional to the time-difference of the upstream and

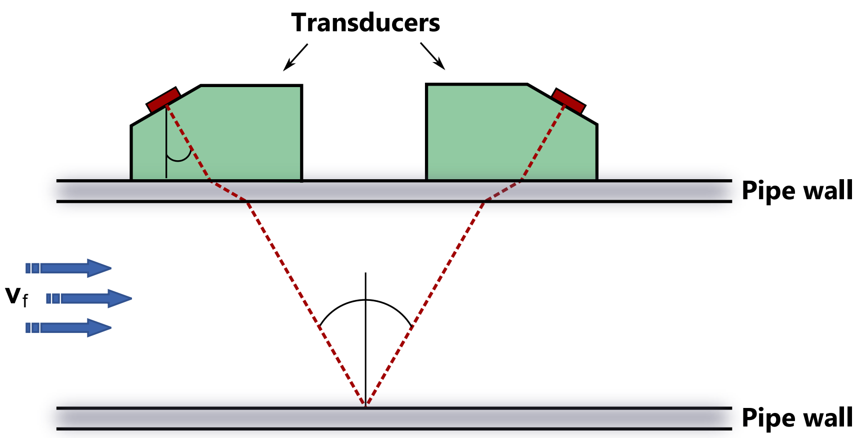


Figure 2.13: Clamp-on flowmeter in V-mode configuration. The dashed line is indicating the path of the ultrasonic wave.

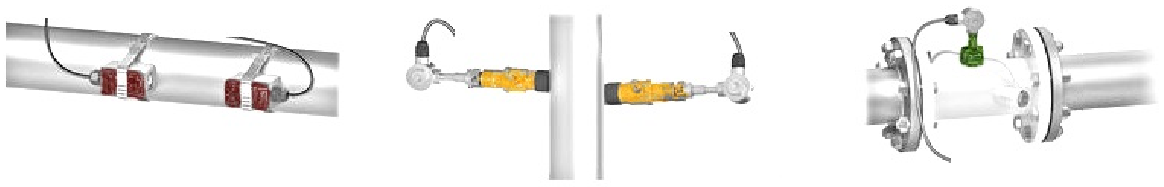


Figure 2.14: Installation types - from left to right: clamp-on, insertion, inline type. Figure adapted from product catalog after [28].

downstream measurement. Detailed mathematical expressions for the calculation of v_f are found in [26].
In the past, clamp-on flowmeters used the doppler method but have been replaced mainly by the usage of the transit time method nowadays. Although the accuracy of the clamp-on type is worser compared to the other types, their advantage lies on the non-invasive nature with low installation costs and easy replacement of defect units without the need to shutdown the production system. [27]

Inline and insertion types are two variants of the so called wetted flowmeters. These meters are exposed to the fluid directly or indirectly. Both types are mounted permanently to the pipe wall. The insertion type requires drilling of the pipes to mount the transducers. On the contrary, the inline type is a rigid piece, called spoolpiece, with pre-mounted transducers, thus, minimizing installation errors and installed in line with the pipe. This type has higher accuracy ($\pm 0.5\,\%$ and better) and long-term stability over the clamp-on type, but is usually more expensive and harder to maintain - a malfunction might require to shut down the flow.

2.3.2.3 Multi Paths and Flow Profile compensation

In section 2.3.2.1, the transit time method is described as a single path application with just a pair of transducers inclined at an angle mounted opposite to each other. The flow velocity is integrated along this single path and calculated by equation 2.19, giving the mean flow velocity. This classical method has two drawbacks, (1) the upstream and downstream measurements are performed at different timings in respect to the current flow rate and thus, only express a mean of the real flow rate and (2) the single path might contain local eddies, which could heavily influence the measurement. Depending on its orientation, the eddy can either slowdown or accelerate the passing wave front. As such, the velocity profile causes the most significant part of the measurement-error.
Kritz, 1955, showed that the mean flow velocity, obtained by integrating over a laminar profile across the diameter of the pipe, results in an one-third high excess-influence of the velocities near the axis. Kritz suggested to use off-diameter paths to have less influence to the flow profile. In 1964, Knapp suggested the first quadrature method, using multiple chords (two crossed paths over the cross section) to properly average over the flow. The chord idea compensates the influence of swirls shown in figure 2.15, since the wave front is passing a swirl in the negative direction on one path and in the positive direction on the crossed path and thus, eliminating the contribution of this disturbance.
Generally, the location of the transducers has a significant impact on the accuracy of the flowmeter, leading to many proposals for different positioning-schemes in the past years presented in figure 2.16. The centered diametric arrangement 2.16a.) has too much influence from the center of the pipe. Figure 2.16b.) shows the proposal of Baker and Thompson (1975) which has a low error over the whole Reynolds number range using two paths. Using multiple paths (Hastings, 1968) with weighting (Vaterlaus, 1995) is used in arrangement 2.16c.) - the integration is more complicated but achieves good quality and high accuracy. Figure 2.16d-e.) are examples of the so called reflex mode - this is a V or W shaped path where transducers can be placed on the same side of the pipe. A triangle path is shown in figure 2.16f.) summing over a large profile-area with benefits from off-axis integration. [15]
Another counter measure for disturbed profiles are flow straighteners. Flow straighteners have a special design-pattern to remove swirls from the flow. They are mounted before the flowmeters to minimize metering errors caused by swirls in the flow. [15]

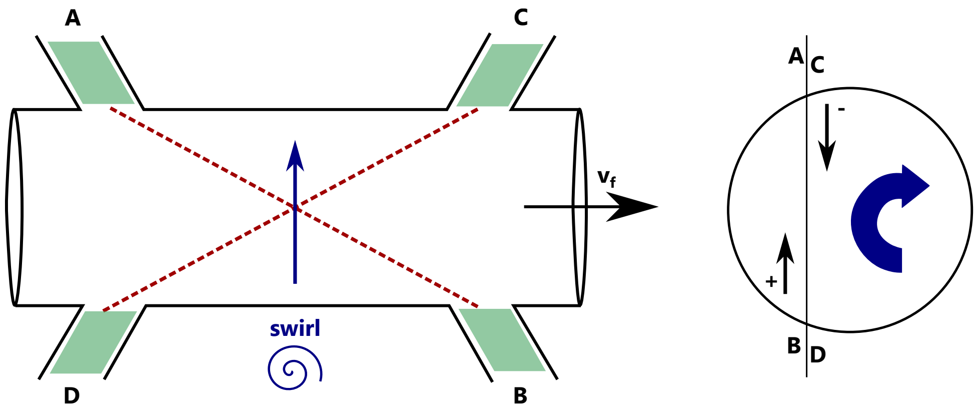


Figure 2.15: Chord installation - Two crossed beam paths (AB and CD) to minimize impact of swirls on the measurement.

In [29] a simulation of a high-precision UFM with 18 paths is presented using special direction transducers, which are able to generate three independent and directed ultrasound waves each. They show that there configuration easily compensates for swirls, is able to achieve an accuracy better than 0.15 %, and is suitable for a rough flow profile estimation.

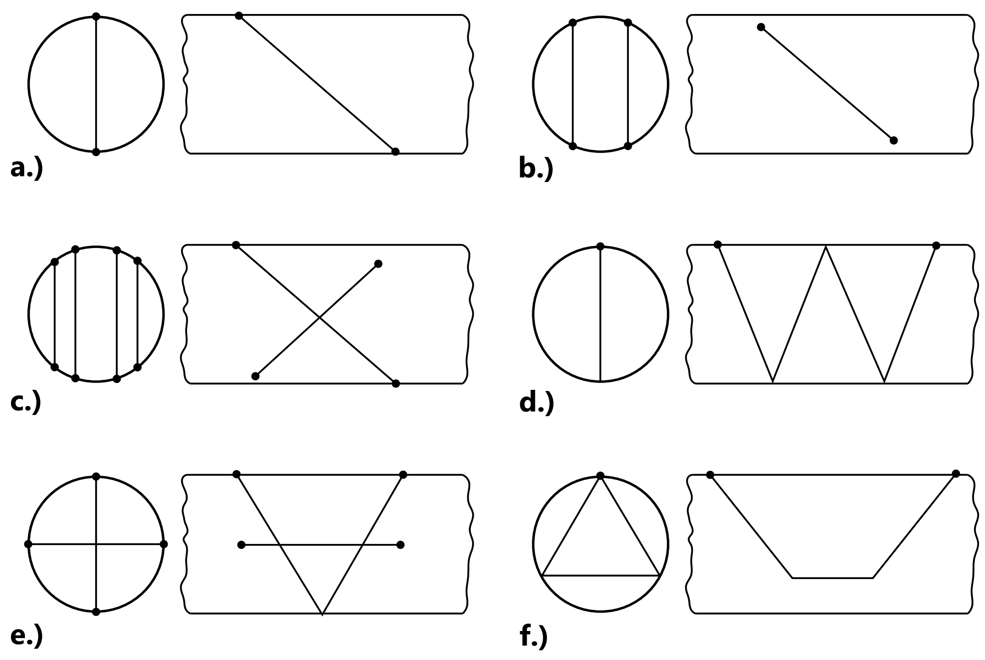


Figure 2.16: Sound paths for various transducer positions. a.) centered diametric arrangement b-c.) Multiple off-axis paths d-f.) Reflex mode: W, V and triangle paths (Adapted from [15], figure 13.5).

2.4 Signal Processing for Flow Determinations

There is a variety of signal processing methods available to determine the flow rate with data from ultrasonic transducers. The threshold method (THM), the cross correlation method (CCM), and the curve-fitting method (CFM) are the most commonly used.
The THM is the simplest one and allows a straightforward implementation on hardware with low-power consumption and limited processing power suited for battery powered applications. The CCM and the CFM are more difficult to implement and require a higher amount of processing energy.
Subsequently, all three methods are elaborated in details.

2.4.1 Threshold Method

At the THM, the echo amplitude is compared to a preset threshold. The intersection at which the signal exceeds the threshold is known as feature point (FP) and used to obtain the TOF. The choice of the threshold is crucial for this method. Typical threshold levels are between -20 to -35 dB below the maximum of the pulse [30], or three to five times higher than the background noise level [31].
THM is very susceptible to noise and unstable flow fields. To improve system robustness

and decrease noise influence, various adoptions of the THM have been presented like the double-THM [32], or the sliding window method with sub-threshold counting [33].

The excitation signals are often bursts of a single frequency with special filtering to optimize spectral efficiency. High oversampling is recommended on the receiving end, since low sampling frequencies decrease the resolution to accurately detect a threshold reach, and as such increase the measured delay. [31]

2.4.2 Curve-Fitting Method

At the CFM, a nonlinear function is used to fit a curve of an ultrasonic echo envelope. The vertex between the echo and the fitted curve provides an approximation of the TOF. The Gaussian model and the exponential model are most frequently used as fitting techniques. The goal is to create an unbiased TOF measurement. The CFM usually has a lower bias compared to the THM [33].

2.4.3 Cross Correlation Method

The cross correlation method uses the cross-correlation function to obtain the TOF. This function computes the similarities between two discrete signals (f, g) and gives a peak at the location where the signals match best. Mathematically this expresses as

$$(f * g)[n] \hat{=} \sum_{m=-\infty}^{\infty} \overline{f[m]} g[m+n] \tag{2.24}$$

where $\overline{f[m]}$ is the complex conjugate of $f[m]$. The maximum peak gives the time shift between the two signals, whereat the width of the peak is proportional to the bandwidth of the signal. [16]
To improve the noise level, signals that produces a narrow peak and low sidelobes are preferred. The Barker code, published in 1953, fulfills such properties. This coding scheme has ideal autocorrelation properties and is listed in table 2.1. [34]

CCM is working well, even when the signal is severely disturbed by noise. It is also unbiased and maximizes the signal-to-noise ratio since the cross-correlation of two ideal random noise signals is theoretically zero.
CCM has the drawback that it is computational more complex. [33]

Length	**Code Sequence**	
2	-1 -1	-1 -1
3	-1 -1 -1	
4	-1 -1 -1 -1	-1 -1 -1 -1
5	-1 -1 -1 -1 -1	
7	-1 -1 -1 -1 -1 -1 -1	
11	-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1	
13	-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1	

Table 2.1: Known Barker codes - perfect binary phase codes. Even lengths also have complementary sequences. After Wikipedia.

2.4.4 Echo Energy Method

The echo energy method (EEM) is a novel signal processing method published by mid of May 2018 [35]. This approach uses the area of the echo energy signal (EES) to find a stable FP to obtain the flow rate. The EES is the squared signal-response to an excitation signal.

2.4.4.1 Working Principle

Figure 2.17 shows a typical shape of an ultrasonic echo signal and their timing-relation to the excitation signal (ES). The ultrasonic transit-time t, is the starting point of the ES until the appearance of the echo signal. The start of the ES can be controlled and measured very accurately.
It is hard to obtain the end-point of t exactly, since the signal energy of the echo is very low and susceptible to noise. As a countermeasure, t will be indirectly calculated involving the position of the FP.
The time t_2, is the span between the start of the echo and the position of the FP. The time duration from the ES to the FP is t_1

$$t_1 = t + t_2 \tag{2.25}$$

t_2 can be determined under zero flow by taking the channel length and the velocity of sound into account. The only variable now is the location of the FP. The determination of the FP includes the EES.

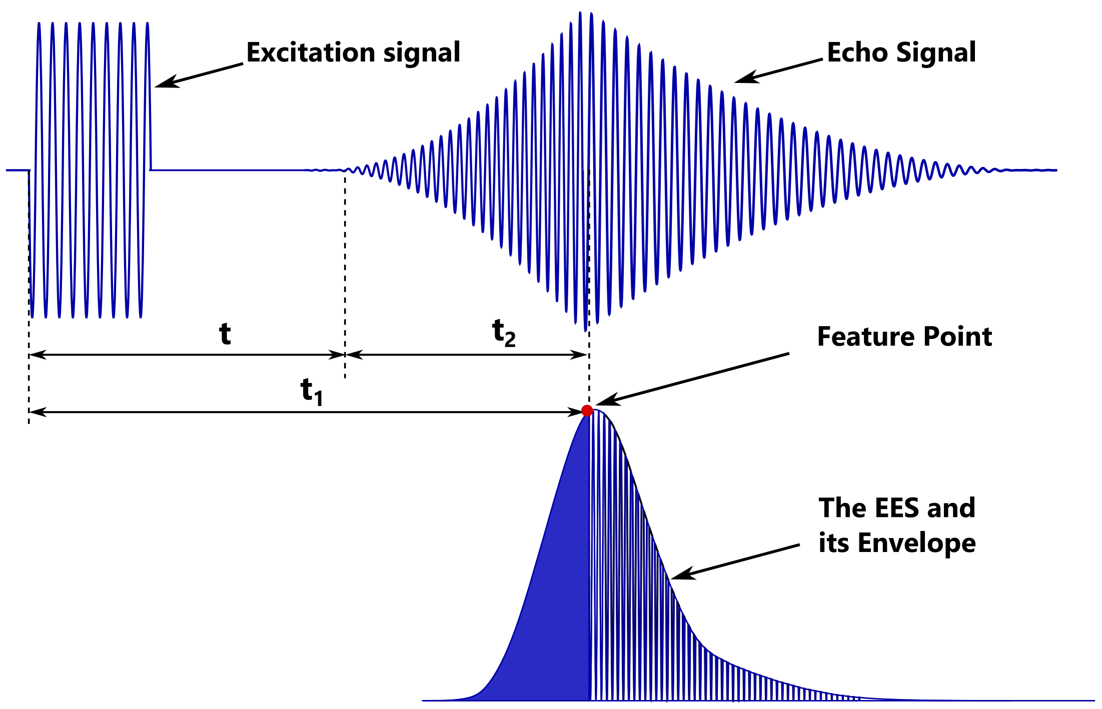


Figure 2.17: Excitation signal with a typical shape of the ultrasonic echo signal which is noticeable after the propagation time t. The feature point is determined on the echo energy signal. Adapted from [35].

From signal processing point of view, the energy E of a discrete signal $x[n]$ is

$$E = \langle x[n], x[n] \rangle = \sum_{n=-\infty}^{\infty} |x[n]|^2 \tag{2.26}$$

Equation 2.26 makes the relation between the echo and the EES. The normalized EESs and its envelopes under different flow rates are shown in figure 2.18. Looking at the envelopes, it is obvious that the rising slopes are unaffected by the flow rate. The descending slopes of the envelopes are slightly distorted, so the focus for the determination of the FP is on the starting section of the EES.

The area under the starting section is now integrated and accumulated until it reaches a preset fixed cumulative energy (FCE) value. This position is used to calculate the exact location of the FP.

As mentioned earlier, the end position of t, which is the starting position of the integration, can not be determined exactly (mainly due to noise). When noise is taken into the integration, the location of the FP might vary under the same flow rate, which is highly unwanted. To avoid such fluctuations, an integral threshold ϵ is defined. Thus, the integration will only start once the EES amplitude reaches the value of ϵ. Since noise is unpredictable and could exceed the value of ϵ way before the real start of the EES,

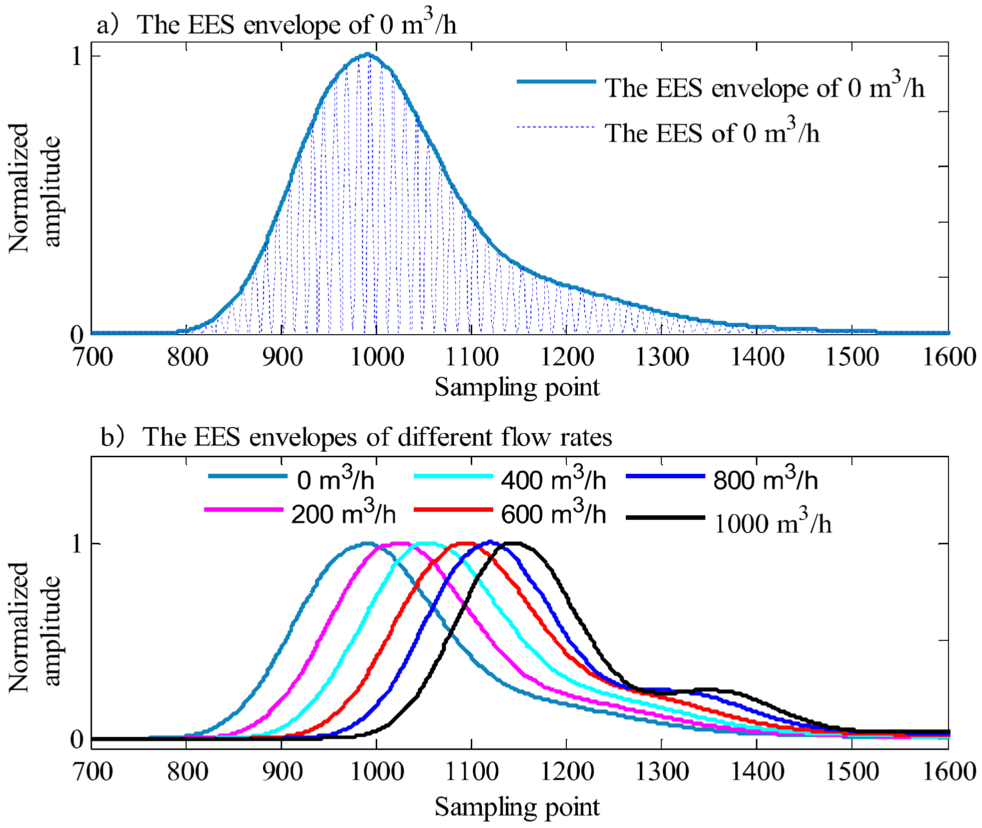


Figure 2.18: Echo energy signal envelopes for different flow rates. Reproduced with permission after [35].

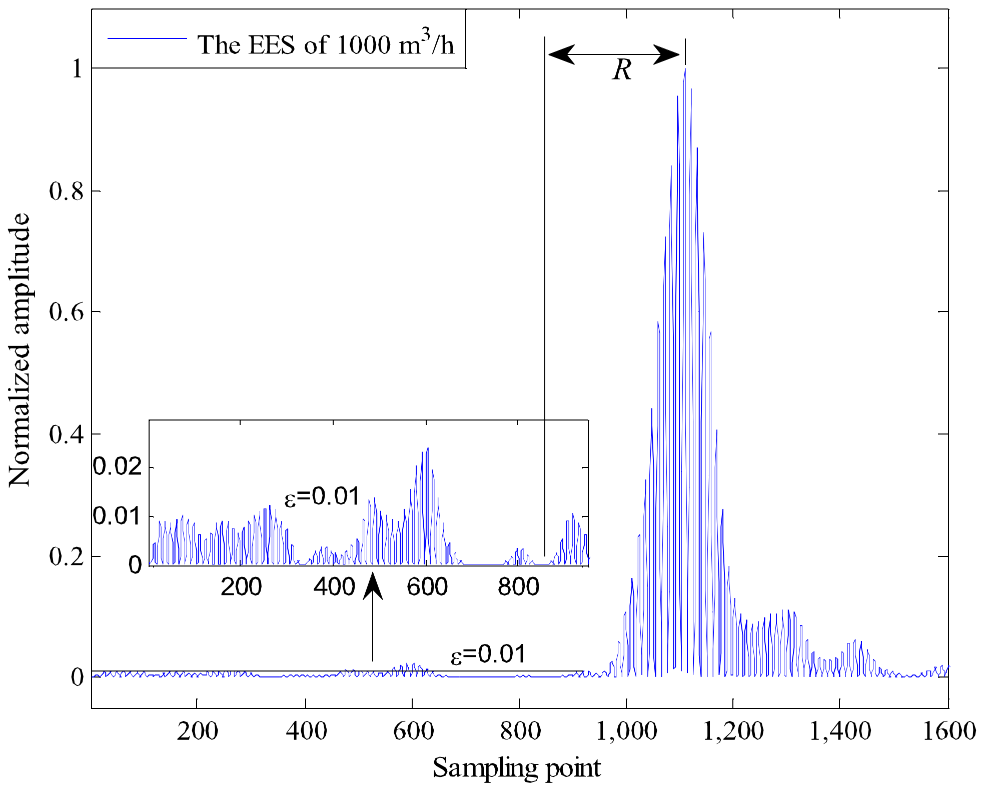


Figure 2.19: Integration range R with threshold ϵ at $1000\,\mathrm{m}^3/\mathrm{h}$. Reproduced with permission after [35].

the time frame for the start of the integration is limited by a range R as shown in figure 2.19. The range R is selected backward and spans 9 to 10 cycles in front of the maximum peak of the EES and defines the starting point for the threshold search. R and ϵ are two countermeasures to reduce the influence of noise to find a stable feature point.

For the surface integration, the EES needs to be subdivided into small areas in the first place. For this purpose, local peak points are identified and connected with a linear fitting method. Two adjacent peak points are now forming a trapezoid. The surface of the rising envelope is now divided into trapezoids $S_1, ..., S_n$, whose areas can be calculated by

$$S_n = \frac{1}{2}(x_{n+1} - x_n)(\overline{y_n} + \overline{y_{n+1}}) \tag{2.27}$$

where $(x_n, \overline{y_n})$ and $(x_{n+1}, \overline{y_{n+1}})$ are the xy-locations of two adjacent peak points. The horizontal bar on the y-locations denotes the normalized amplitude. The area of the single trapezoids is summed up until its value is greater than or equal to the FCE value. The sum corresponds to the cumulative energy S, which is

$$S = \sum_{i=1}^{n} S_i \leq FCE \tag{2.28}$$

where n is the number of trapezoids until the FCE value is reached. The FCE value S_A is subtracted from S to obtain the energy difference, shown in figure 2.20. The area difference shown in figure 2.21 is highlighted as yellow trapezoid between the adjacent peak points b and a. The area difference value S' is again a trapezoid, obtained by

$$S' = S - S_A = \sum_{i=1}^{n} S_i - S_A \tag{2.29}$$

The corner points b and a, at locations (x_1, y_1) and (x_2, y_2) are known from the finding of the local peak points. As such, the slope k is calculated by

$$k = \frac{y_2 - y_1}{x_2 - x_1} \tag{2.30}$$

Generally, the area of a trapezoid S' is

$$S' = \frac{1}{2}\left[(y_2 - kx) + y_2\right] x \tag{2.31}$$

where x is the width of S', corresponding to the horizontal offset of FP1 from point a. The unknown value x can be obtained by reforming equation 2.31 to

$$x = \frac{2y_2 - \sqrt{4y_2^2 - 8kS'}}{2k} = \frac{y_2 - \sqrt{y_2^2 - 2kS'}}{k} \tag{2.32}$$

There is only one applicable solution for equation 2.32, and thus, the horizontal offset x and vertical offset $k \cdot x$ for FP1 are definite.

The selection of the FCE value is crucial for a stable FP. In figure 2.22 two FPs for different FCE values are shown. Is the FCE value chosen low, less trapezoids are taken into account for the summation, resulting in FP2 which is in the lower half of the ascending section of the envelope, labeled as trapezoid S_c. Is the FCE value chosen higher, more trapezoids are summed up resulting in FP1 which is in the upper half of the ascending section of the envelope, labeled as trapezoid S_a. Since the send out energy, as well as the FCE value remain constant, the area of S_a and S_c is equal. For small fluctuations, the vertical offset of FP1 will vary much less than the one of FP2. For stability reasons, it is therefore beneficial to choose a FCE value that is between the maximum peak point a and the preceding point b of the EES.

Once a FP is found, the transit-time of the upstream t_u and the downstream t_d are calculated by

$$t_u = T_1 + \Delta t_u - t_2 = T_1 + x_u T_S - t_2 \tag{2.33}$$

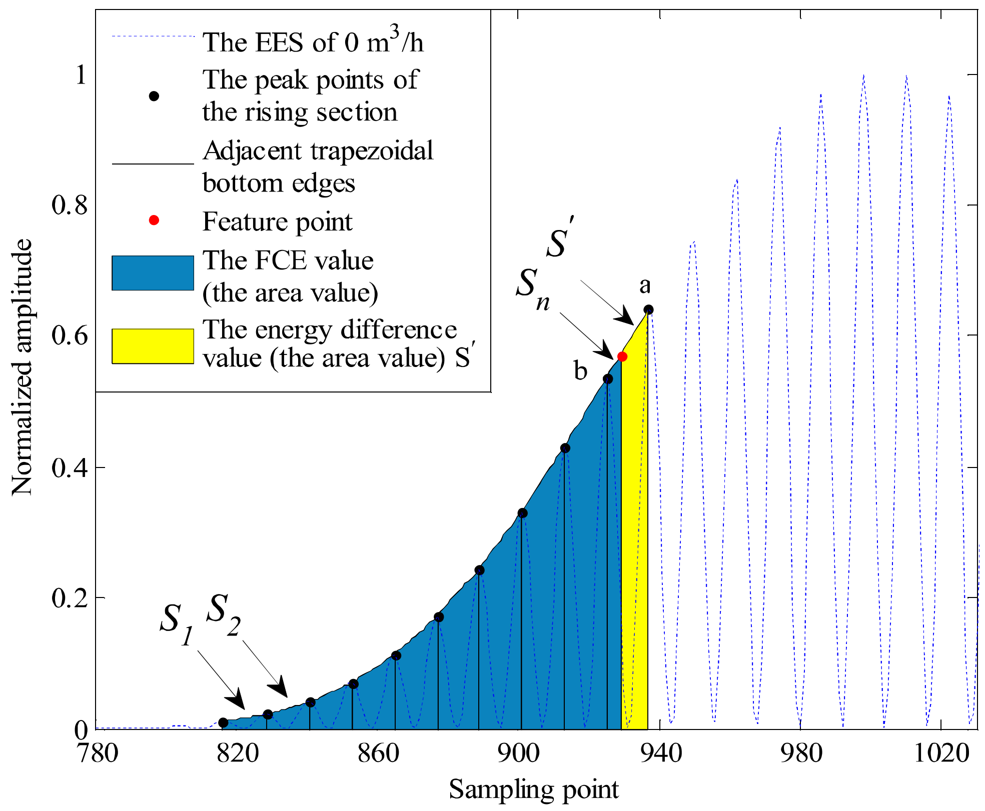


Figure 2.20: Cumulative energy of the rising section. Reproduced from [35] with permission.

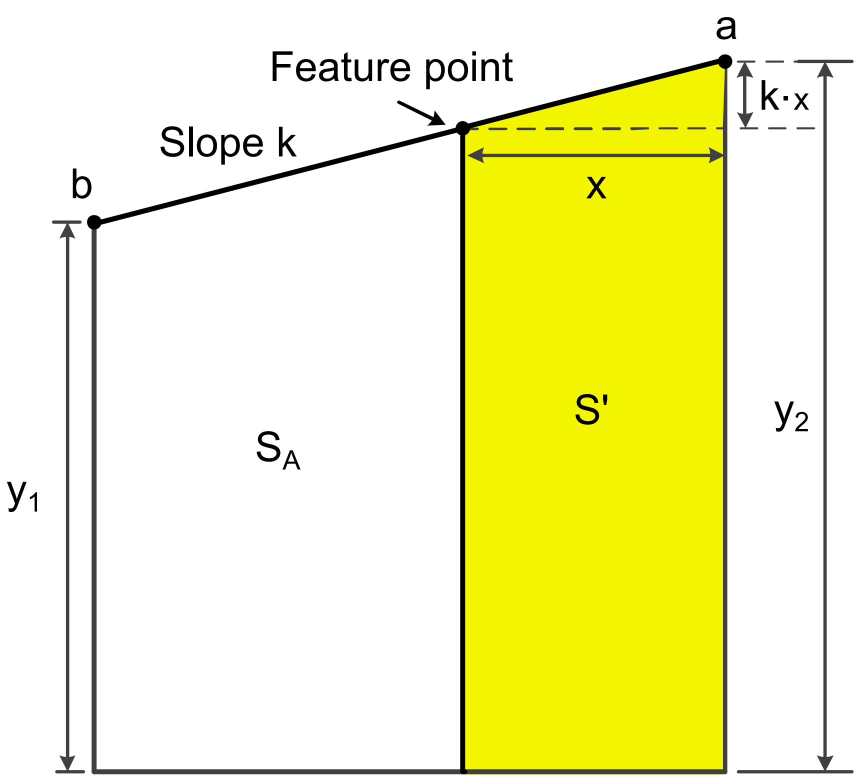


Figure 2.21: Feature point calculation with the area difference of the FCE value. Adapted from [35].

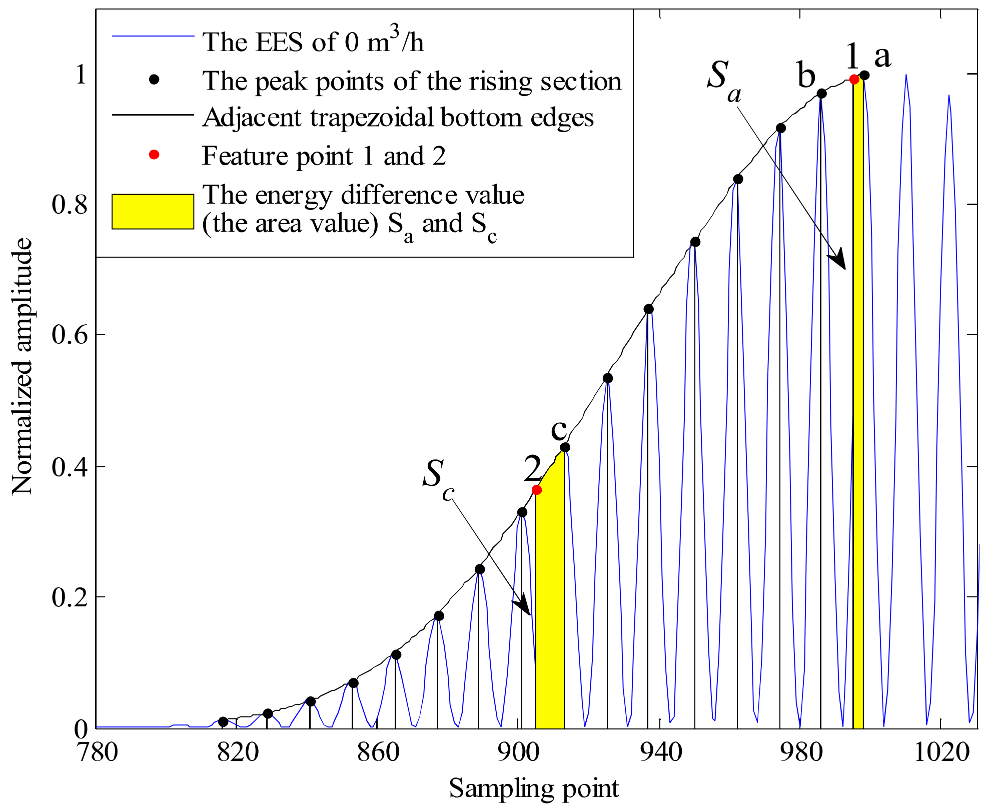


Figure 2.22: Stability analysis on feature points 1 and 2. Reproduced from [35] with permission.

$$t_d = T_1 + \Delta t_d - t_2 = T_1 + x_d T_S - t_2 \tag{2.34}$$

where T_1 is the time difference between the ES and the start of sampling. Δt_u and Δt_d are the time frame from the start of sampling to the position of the FP of the respective transit-times. x_u and x_d are the horizontal offsets of the FPs of t_u and t_d. T_S is the sampling frequency of the echo signal.

Finally, the flow rate q_v is

$$q_v = v_f D_p \alpha = \frac{L}{2cos\Phi}\left(\frac{1}{t_d} - \frac{1}{t_u}\right) D_p \alpha \tag{2.35}$$

where α is a correction factor for the number and distribution of the channels combined with the Reynolds number and D_p is the pipe diameter.

2.5 Basics on Simulation of Wave Propagation

This section introduces the basic theory for a simulation of a wave propagation. The simulation is a starting point to get familiar with the principles of ultrasonic measurements.

With the simulation data two different signal processing techniques for flow velocity calculations are evaluated.

2.5.1 Introduction

The speed of sound, an essential quantity in the field of acoustic wave propagation, was already introduced in section 2.1.3.1. This paves the way to introduce the wave equation. Before digging into the deep math of wave equations, we start with the definition of a wave. A wave is a disturbance that carries energy through a medium or vacuum. There are many kinds of waves, like electromagnetic waves, sound waves or gravitational waves, all sharing similar properties when propagating. Since we are dealing with ultrasound, all of the following explanations and equations will be based on the acoustic scope.

The wave equation in its general form in one dimension x, is written as

$$\frac{\partial^2 p}{\partial t^2} = c^2 \frac{\partial^2 p}{\partial x^2} \tag{2.36}$$

and with a mathematical different notation as

$$p_{tt} = c^2 p_{xx} \tag{2.37}$$

Equation 2.36 is classified as second-order hyperbolic partial differential equation (PDE) and connects the time derivative of the pressure p, and the space derivative of the pressure with the speed of sound c. The d'Alembert principle gives a general analytical solution for this equation. In this work we do not focus on the analytical solution, but rather on finding solutions to perform algebraic computation in software. For an algebraic computation all derivatives in a PDE are replaced by numerical methods.

2.5.2 Numerical methods

Typically, numerical methods discretize a continuous problem into smaller elements. In the discretion step, the simulation space, might be a landscape, a pipe or any other structure, is divided into grid lines. The grid intersections, also known as grid points or nodes, represent dependent variables of the PDE. A grid is known as collocated, if each node stores all dependent variables.

For a two-dimensional grid, the spacing in horizontal direction is Δx, and Δy in the vertical direction. The grid spans in x-direction with $i = 0, ..., n_x$ nodes, and in y-direction

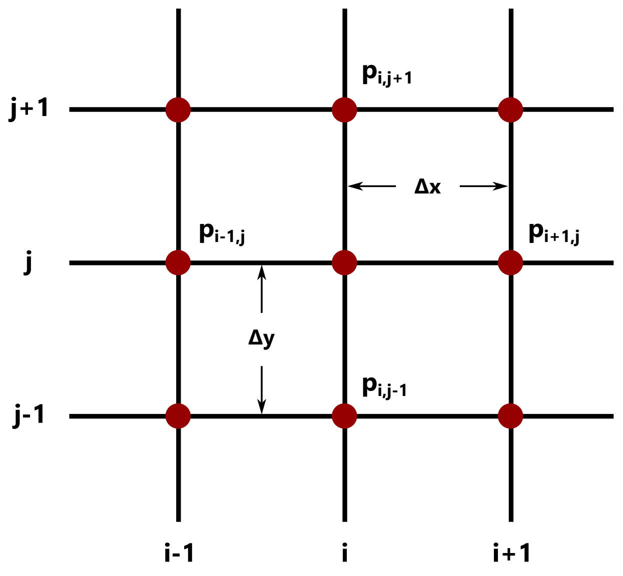


Figure 2.23: Uniformly spaced two-dimensional grid, also known as square.

with $j = 0, ..., n_y$ nodes. A grid is said to be a uniform, if the spacing in one of the directions is constant. A grid is called square, if the spacing is equal in all directions ($\Delta x = \Delta y$ see figure 2.23).
A node, at a certain discrete location (i, j), is represented by $p_{i,j} = p(x_i, y_j) = p((i - 1)\Delta x, (j - 1)\Delta y$. Depending on the problem domain, the grid, also known as mesh, might be non-uniform or irregular, to represent complex-shaped structures.
Certain regions can be subdivided into finer structures to have a higher resolution for rapid changing dependent variables. There are even algorithms to adapt the mesh dynamically to mimic anisotropic features presented in [36].

2.5.3 Finite Difference Method

The finite difference method (FDM) attempts to approximate all derivative terms in a PDE with difference equations. Taylor's theorem in combination with the finite differences, introduced by him back in 1715, are used for this approximation.

Generally, a first-order derivative of a function p at point x is defined as

$$p'(x) = \lim_{\Delta x \to 0} \frac{p(x + \Delta x) - p(x)}{\Delta x} \tag{2.38}$$

and its second-order derivative written as

$$p''(x) = \lim_{\Delta x \to 0} \frac{p'(x + \Delta x) - p'(x)}{\Delta x} \tag{2.39}$$

FDM type	Expression	Truncation error
forward difference ($T1$)	$p_i^{'} \approx \frac{p_{i+1}-p_i}{\Delta x}$	$O(\Delta x)$
backward difference ($T2$)	$p_i^{'} \approx \frac{p_i-p_{i-1}}{\Delta x}$	$O(\Delta x)$
central difference ($T1-T2$)	$p_i^{'} \approx \frac{p_{i+1}-p_{i-1}}{2\Delta x}$	$O(\Delta x)^2$

Table 2.2: Most commonly used FDM types with their truncation error.

For a discrete point p_i, and its subsequent neighbour p_{i+1}, a Taylor series expansion expresses as

$$p_{i+1} = p(x_i + \Delta x) = p_i + \Delta x p_i^{'} + \frac{\Delta x^2}{2!} p_i^{''} + \frac{\Delta x^3}{3!} p_i^{'''} + ... + O(\Delta x) \tag{2.40}$$

where $O(\Delta x)$ is the truncation error term.

By reordering equation 2.40 to express $p_i^{'}$ we get

$$T1: \quad p_i^{'} = \frac{p_{i+1} - p_i}{\Delta x} - \frac{\Delta x}{2!} p_i^{''} - \frac{\Delta x^2}{3!} p_i^{'''} + ... + O(\Delta x) \tag{2.41}$$

Similarly it can be done for the prior point, p_{i-1} of p_i

$$p_{i-1} = p(x_i - \Delta x) = p_i - \Delta x p_i^{'} + \frac{\Delta x^2}{2!} p_i^{''} - \frac{\Delta x^3}{3!} p_i^{'''} + ... + O(\Delta x) \tag{2.42}$$

$$T2: \quad p_i^{'} = \frac{p_i - p_{i-1}}{\Delta x} + \frac{\Delta x}{2!} p_i^{''} - \frac{\Delta x^2}{3!} p_i^{'''} + ... + O(\Delta x) \tag{2.43}$$

The three most common considered FDMs can be expressed from equations 2.41 and 2.43, by neglecting the second derivative and $O(\Delta x)$, listed in table 2.2. The order of the approximation is a crucial criteria for the accuracy of a FDM. By evaluating $O(\Delta x)$, a quantitative comparison can be made.

Obviously, the wave equation depends on time and space. In the previous steps, the approximation of the derivative in space is shown. The same procedure applies for the derivative in time. We introduce a grid in space-time with $t_{n+1} = (n+1)\Delta t$, noted as p^{n+1}, where Δt is the time step. Consequently, the derivative in time approximates to

$$p_t \approx \frac{p_i^{n+1} - p_i^n}{\Delta t^2} \tag{2.44}$$

The FDMs are applicable for higher-order derivatives as well. The central difference approximation for a second-order derivative in time is

$$p_{tt}(x_i) \approx \frac{p_i^{n+1} - 2p_i^n + p_i^{n-1}}{\Delta t^2} \tag{2.45}$$

and in space

$$p_{xx}(x_i) \approx \frac{p_{i+1}^n - 2p_i^n + p_{i-1}^n}{\Delta x^2} \tag{2.46}$$

The one-dimensional wave equation 2.37 can now be approximated by equation 2.45 and 2.46, resulting in:

$$\begin{gathered} p_{tt} = c^2 p_{xx} \\ \frac{p_{i+1}^n - 2p_i^n + p_{i-1}^n}{\Delta x^2} = c^2 \frac{p_i^{n+1} - 2p_i^n + p_i^{n-1}}{\Delta t^2} \end{gathered} \tag{2.47}$$

Rearranging equation 2.47 to calculate a future time-step gives:

$$p_i^{n+1} = 2p_i^n - p_i^{n-1} + \underbrace{\left[c \frac{\Delta t}{\Delta x}\right]^2}_{CFL^2} \left(p_{i+1}^n - 2p_i^n + p_{i-1}^n\right) \tag{2.48}$$

where CFL is the so called Courant-Friedrichs-Lewy condition (see section 2.5.5).

The two-dimensional form of equation 2.48 is:

$$p_{i,j}^{n+1} = 2p_{i,j}^n - p_{i,j}^{n-1} + CFL^2 \left(p_{i+1,j}^n + p_{i-1,j}^n - 4p_{i,j}^n + p_{i,j+1}^n + p_{i,j-1}^n\right) \tag{2.49}$$

Looking at equation 2.49, a certain access pattern to neighbouring points is visible in order to update a future value for the current node $p_{i,j}^{n+1}$. This influence of nearby points is called stencil or mask and is specific to the FDM in use.
By looking at the grid in figure 2.24, the stencil can not be applied to boundary points directly, since neighbouring nodes for an update would be missing (dotted surrounding). Thus, only the inner nodes can be updated by the FDM. All points that can not be reached by the stencil are called ghost points and are part of the boundary layer. These points dependent on the given boundary conditions (see section 2.5.4) and are calculated separately.

2.5.4 Initial and Boundary Conditions

To fully solve a PDE, well defined initial and boundary conditions are needed. It needs a right amount of specified conditions to achieve a unique solution. Conditions with a small error, these are so called ill-posed problems, might have a tremendous impact on

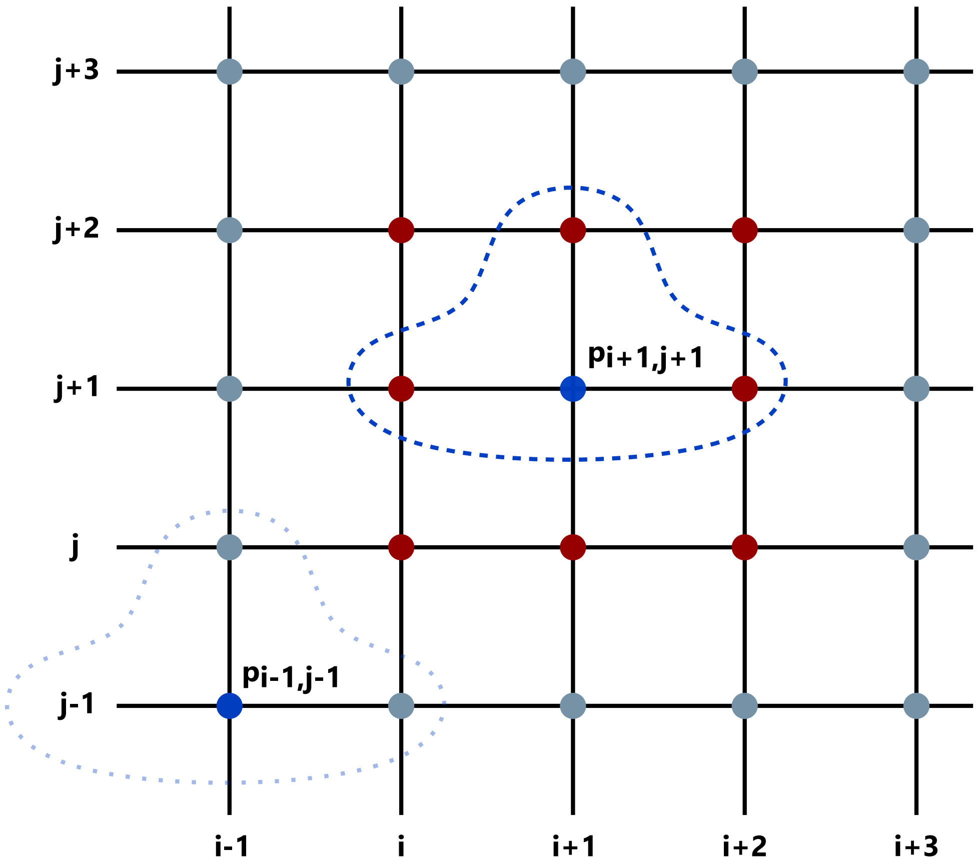


Figure 2.24: Stencil for central FDM on a two-dimensional grid (dashed lines). Boundary grid points, also known as ghost points, are coloured gray. Special handling on ghost points needed - dotted stencil is missing neighbouring points for an update.

the solution [37]. Initial conditions give the system a defined starting point and need to be well chosen.
Boundary conditions are needed to update the values of the ghost points. Those ghost points are representing the physical boundaries of the simulation space, and as such need to mimic the physical properties of those borders. With regards to a wave, this might be attenuation, reflection, absorption, slip conditions or a similar effect. Usually, ghost points are the points at extrema locations (one of the coordinate equals to $i = 0$, $i = n_x$, $j = 0$ or $j = n_y$). The stencil decides how many nodes are part of the ghost points.

For open boundary simulations (infinite stretched out space), reflections or wraparounds at borders are highly undesired. The most straightforward solution is, to simply extend the numerical mesh to delay reflective events. This solution might have a dramatic impact on the computational time and therefore the implementation of artificial absorbing conditions (ABCs) [38], [39] or nonreflecting conditions [40] are preferred, or sometimes even indispensable.
The goal to absorb the incident wave by hundred percent at the border, using absorbing or nonreflecting boundary conditions (NBCs) seems to be impossible - edge artifacts and the difficulty of handling complex-structured meshes with many neighbouring points on the borders will always lead to low level reflections. Even in a combined approach of those two methods [41], weak reflections are still a present.
An alternative approach is given by a perfectly matched layer (PML), introduced by Berenger in 1993 [42]. The PML uses an absorbing layer instead of an absorbing condition and is designed in a way to avoid reflections totally (theoretical) by involving a change to a complex coordinate system in which propagative modes are highly damped and evanescent modes remain unaffected.
In practice, ABCs are more general compared to PMLs, but are limited to absorb incident waves only at certain angles. PMLs usually have a much higher degree of damping on the reflections, but are not suited for all kind of numerical schemes, since the method is not always feasible. Nevertheless, they are the technique of choice to model absorbing conditions, if applicable. [43], [44]

2.5.5 Finite Difference Schemes

Finite difference schemes (FDSs) are divided into explicit and implicit schemes. At explicit schemes, like Forward Euler, Upwind, Lax-Friedrichs or Leapfrog [45], [46], [37], the calculation for the next time step involves the values from the previous time step. This dependency is limiting the advance of the time to a maximum allowable step-size Δt. Violating this restriction might lead to divergence of the algorithm.
The time step is governed by the so called Courant-Friedrichs-Lewy (CFL) condition, which gives a necessary, but not sufficient condition for convergence while solving special kind of FDM problems. The CFL is defined as

$$CFL = c\frac{\Delta t}{\Delta x} \leq C_{max} \tag{2.50}$$

where Δx is the grid spacing and C_{max} a constant depending on the FDM method in use - typically $C_{max} = 1$ for central FDM schemes. Looking at equation 2.50 and assuming c as constant, there are two degrees of freedom to influence the convergence as well as the accuracy of the simulation. There is no rule for an exact value of the time step or the grid spacing. A too large value of Δt or Δx might give numerical errors or unstable results, whereas too small values lead to a significant increase of the computational burden.
The advantages of the explicit scheme are the possibility to solve for all unknowns at a point with a single step and usually a straightforward implementation of the algorithm.

For implicit schemes, there is no explicit formula, which means that the terms of a future time step appear on both sides of the FDS. This requires solvers for linear equations to get a solution over the whole grid, which generally means more effort on the implementation. In contrast to the explicit schemes, there is no restriction on the time step - they are always stable. The chosen time step is only a factor for the aspired accuracy. Well known explicit schemes are Backward Euler, Crank-Nicolson and Runge-Kutta [45], [46], [37].

2.6 Hardware

2.6.1 Embedded Systems

The term embedded system (ES) has many definitions.

"... is a microprocessor-based system that is built to control a function or range of functions and is not designed to be programmed by the end user", after Heath (2002) [47].

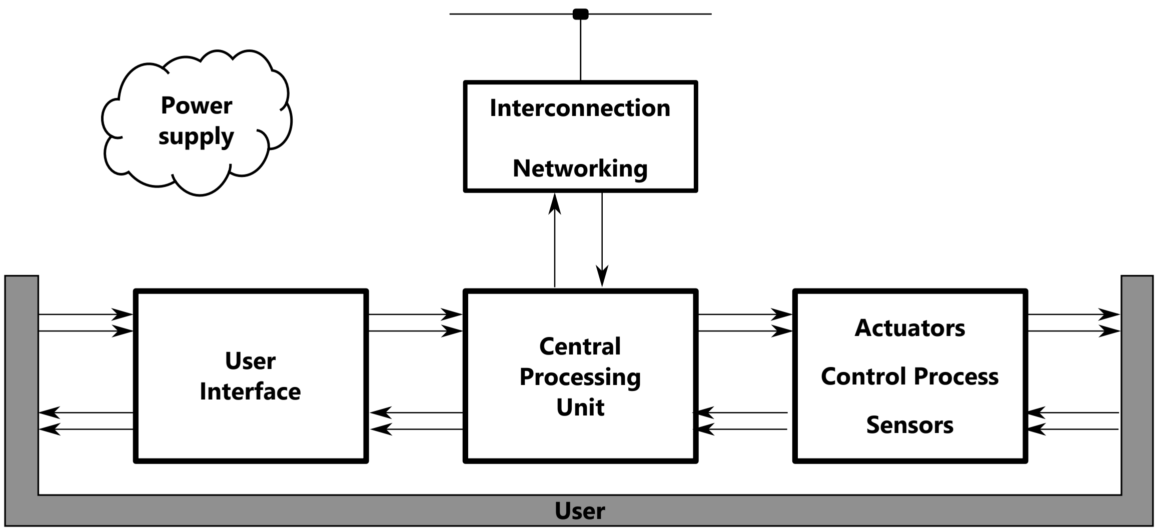


Figure 2.25: Three layer model of an embedded system.

”A combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a dedicated function”, after the BARR group [48].

An ES is made to fulfill a specific task. The processing unit usually does not emerge from an ES, it is rather the functionality of the system that is in the foreground. An ES interacts with its environment with sensors and actuators and can be abstracted by a three layer model shown in figure 2.25. The system usually consists of a layer with an interface to the outer world, a processing or action layer and a user layer. In the processing layer different kind of architectures are in use. The CPU could be a simple microcontroller, a FPGA, a DSP, an ASIC or a hybrid system.

2.6.2 ARM®Architecture

ARM®, former known as Acorn RISC Machine (Reduced Instruction Set Computer Machine), was introduced in 1985 by the British company Acorn. The architecture is RISC based and used in microcontrollers and -processors by a variety of manufacturers. ARM®is selling IP cores via a licensing scheme rather than building hardware themselves. [49]
Low energy consumption paired with high processing power makes them a preferred choice for embedded systems especially in the mobile sector. The ARM®processor uses a Harvard architecture with separate paths for instructions and data, and a bus-width of either 32-bit or 64-bit.

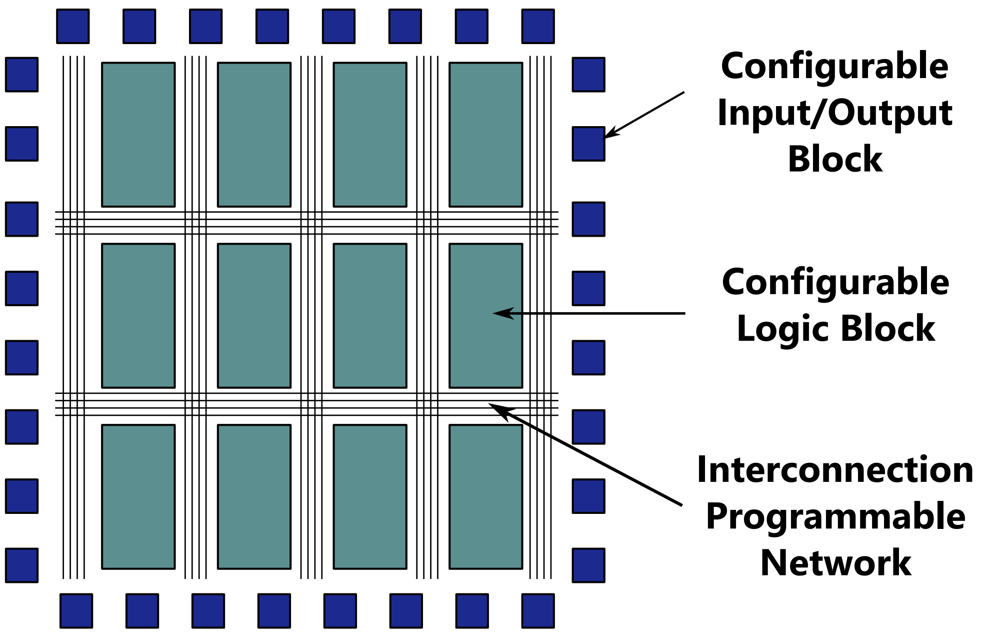

Configurable Input/Output Block
Configurable Logic Block
Interconnection Programmable Network

Figure 2.26: Generic structure of an FPGA.

2.6.3 Field Programmable Gate Array

An Field Programmable Gate Array (FPGA) is a generic programmable component in terms of hardware and belongs to the family of programmable logic components. The hardware is not pre-configured and contains logical blocks which can be freely connected together.

Figure 2.26 shows the generic structure of an FPGA. It consists of configurable logic blocks (CLBs) shown in figure 2.27. The CLBs can be linked by an interconnection network. A ring of I/O blocks surrounds the structure for external interfacing. The CLBs are configured to express arbitrary digital functions in combinational or sequential way. For the combinatorial part, lookup tables (LUTs) are employed, which can also be configured to be used as ROM or RAM. The sequential part is driven by D-type Flip Flops with all its control lines. [50]

FPGAs are available in several programming technologies. They differ between the re-programmable types (Flash/EEPROM, SRAM) and the one-time programmable type (Anti-fuse). Among them, the SRAM type is the most widely used one, since it scales well with the CMOS technology, benefiting from the dense integration level and high speed capabilities nowadays. The SRAM type lacks the capability of storing the FPGA configuration when supply is turned off and therefore needs to be programmed each time it is turned on.

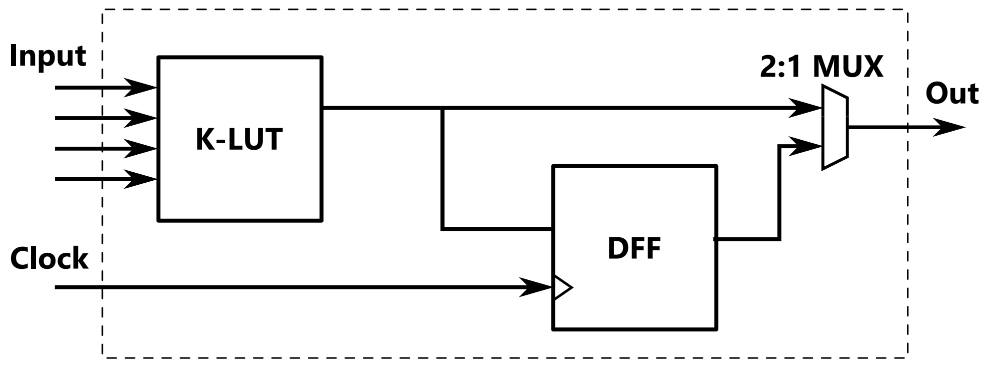


Figure 2.27: Logic block structure - Lookup table with k-inputs and a D-Flipflop. The multiplexer controls the active path for the output.

Preserving configuration on power-down and instantaneous functionality on power-up are the main advantages of the flash based type. Limited amount of reprogramming cycles and their need of none-standard CMOS processes are the weaknesses of the flash type.
The anti-fuse type creates permanent links with interconnect elements. They have a small footprint, lower on-resistances and lower parasitic capacitances compared to the other types. The need of special programming equipment and the inability of reprogramming are the reasons why they are only used for special applications in aviation and aerospace. [50], [51]
In the past years, there is the strong tendency to combine a pure FPGA core with other high-performance hardware peripherals. Dedicated RAM blocks, DSPs, serial transceivers, soft processors such as the Microblaze from Xilinx or the Nios II from Altera (see [52]), and hard processors (ARM or PowerPC) are being combined with a FPGA core, resulting in a single powerful chip. Also analog resources like ADCs and DACs are finding its way to be coupled with FPGA cores. [53]
To recap, FPGAs are flexible, have a short time-to-market, there are plenty of highly sophisticated IPs available for rapid prototyping, they can be (dynamically) reprogrammed, allow verification of ASIC designs and have many other benefits making them a good choice for various applications. [53]

2.6.4 Zynq System-on-Chip

The Zynq platform is a All-Programmable System-on-Chip (SoC) solution provided by the company Xilinx. It combines a dual-core ARM®Cortex-A9 processor with a Xilinx 7-series FPGA on a single chip. As such, the hardware is divided into two parts, namely the processing system (PS) corresponding to the ARM cores, and the programming logic

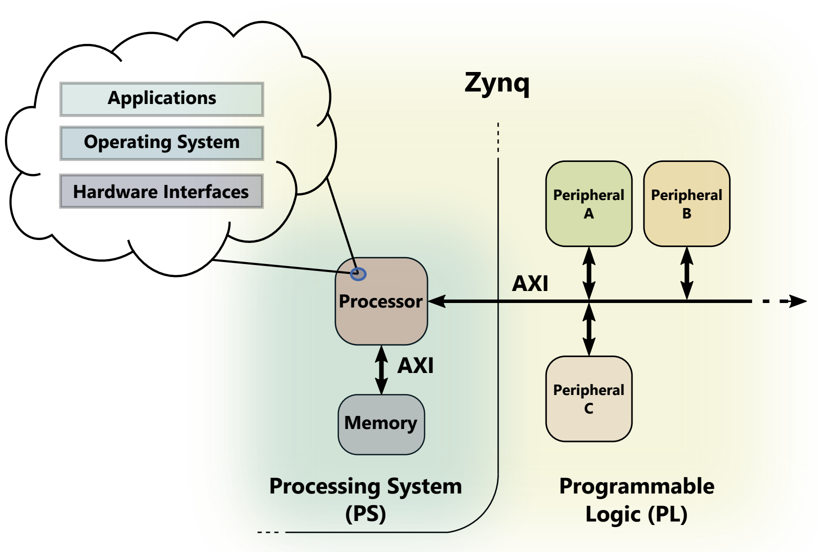


Figure 2.28: Zynq architecture: Schematic relation between the processing system and the programmable logic. Adapted from [54].

(PL) representing the FPGA. The two parts are connected by a special bus system called AXI (see section 2.6.5). A schematic relationship between the PS and PL layer is shown in figure 2.28. The peripherals shown in the figure are design blocks called Intellectual Property (IP) blocks. For the Zynq system, many IPs are already available enabling a quick and easy integration into the custom application. [54]

Xilinx provides a software suite named Vivado Design Suite to model custom IPs in VHDL or Verliog and create complete system designs for Zynq platforms. Vivado supports the simulation and synthesis of custom HDL designs, creates overall timing analysis, reports estimated power consumption and allows in-circuit debugging via JTAG and special debug-cores. The graphical interface of Vivado supports the user to design a FPGA-system in a visual manner. Custom IPs are connected to Xilinx IPs in a block diagram. Figure 2.29 visualizes a simple block diagram with a PS connected to an GPIO block.

2.6.5 Advanced eXtensible Interface

The Advanced eXtensible Interface (AXI) is part of the Arm Advanced Microcontroller Bus Architecture (AMBA). The AMBA protocol is an open standard interconnect specification for peripheral communication based on a multi-master design. AXI is extremely

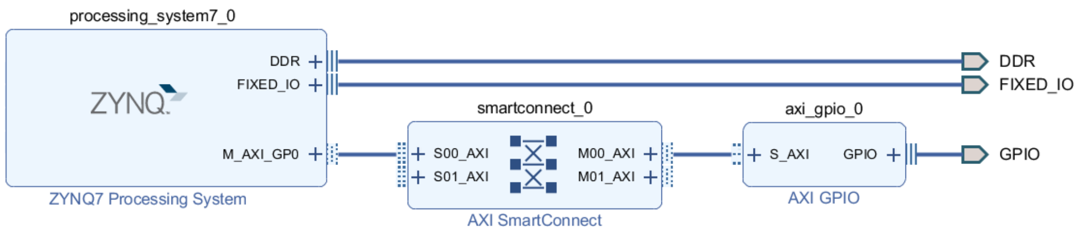


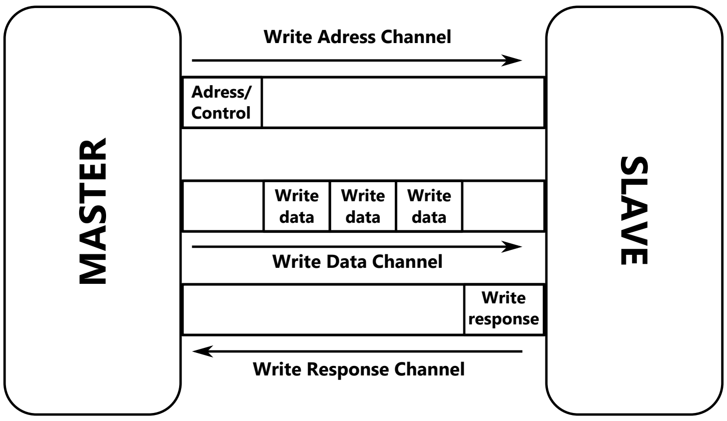
Figure 2.29: Essential blocks - Processing system connected to a AXI GPIO block via a smartconnect.

popular within FPGA designs for interfacing IP cores, enabling easy connectivity between complex designs.
The latest version is AXI5, introduced in the beginning of 2018 by ARM®. Nevertheless, all further explanations will be based on the older version AXI4, since the current version of the Vivado Design Suite (2018.3) does not offer support for latest AXI standard. [54]
Basic features include high-bandwidth, low-latency, separate control and data phase, burst transactions and separate data channels for reading and writing. [54], [55]
There are three basic variants of the AXI4 interface:

1. AXI - Is the fully featured high performance interface for memory mapped communication.
2. AXI-Lite - Is the lite variant of AXI, restricted to a few basic transaction types with small footprint and simplified implementation.
3. AXI-Stream - Is suited for data streaming, like video or audio streams, without an address phase and no restriction on the burst length.

The widths of the data bus range from 32-bits (AXI4-Lite) up to 256-bits for the AXI4 interface and 32-bits for the address bus. [56]

The AXI protocol defines a simple set of rules for communication between a master and a slave module. Multiple masters can be connected to multiple slaves with an AXI interconnect, or the newer version called AXI SmartConnect. The master transfers data over the write data channel and retrieving data via the read data channel from a slave shown in figure 2.30. For none streaming interfaces, a read or write transaction always starts with an address/control phase followed by a data phase. The write channel has an additional response line for write acknowledgments. Each AXI transaction has an ID tag, enabling out-of-order transactions to cope with fast and slow responding slaves, increasing overall system performance. [54], [55]



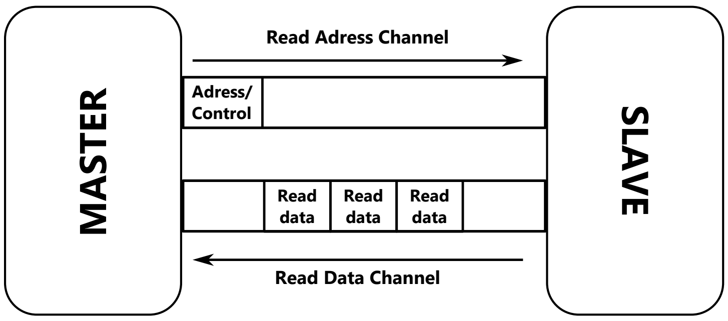


Figure 2.30: AXI write channel (on top) and read (at bottom) channel architecture.

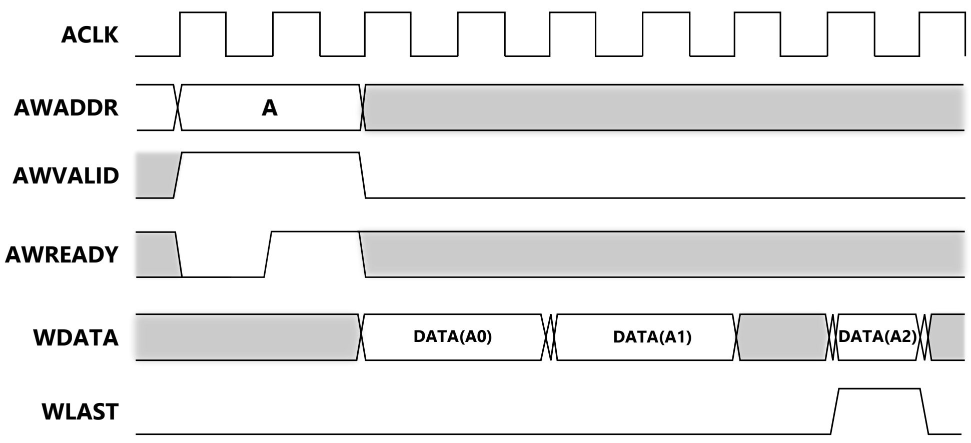


Figure 2.31: Simplified signal transitions for an AXI write burst transaction.

Figure 2.31 shows a simplified AXI write transaction between a master and a slave. In the address phase, the master sets the destination address on the *AWADDR* signal and asserts the *AWVALID* signal indicating a valid address. The slave signals to be ready to retrieve data by driving the *AWREADY* line to high. The master sends now a burst of data, whereas the last data packet is indicated by a high on the *WLAST* signal. For simplicity, several control and acknowledgment signals included in such a transfer have been omitted. A read transaction works similar with the main difference that a separate read channel is used. The explanation of all bus signals can be found in [56].
Generally, it is beneficial to design all custom IPs with an AXI interface to provide easy connectivity and reusability across various FPGA system.

2.6.6 Direct Memory Access Engine

A Direct Memory Access (DMA) engine enables peripherals like the UART accessing the data-bus independently from the CPU to read or write memory. Expensive memory transfers from I/O-devices or copying memory to memory can be executed by the DMA, and as such, heavily unload the CPU. The CPU can initiate data transfers and perform other tasks in the meantime. Usually an interrupt is triggered once a transfer is finished to let the CPU know the transfer completed.

Typically, all operating systems use the concept of virtual memory. A contiguous virtual address space is mapped to certain physical addresses in real (physical) memory. The address translation is done by the memory management unit (MMU). A DMA does not communicate with the MMU and therefore operate purely with physical addresses. This

needs to be taken into account when configuring the DMA for a data transfer.
There are two more things which needs to be considered when using a DMA: firstly, parallel access to the data-bus needs to be avoided - usually this is handled by a bus-arbiter in hardware; secondly, a DMA transfer can lead to cache coherency problems. In a system with a cache, data fetched by the CPU is read from its original location and temporarily stored in the cache. Subsequent accesses or modifications of the same data location are performed on the cached data, allowing a faster access. The modified data is written back from the cache to its original location on demand - note that certain other techniques exist.
Usually, the DMA is not aware of the caching state of single data locations and might read outdated information from the physical location which is in parallel updated by the CPU in the cache location only. This leads to coherency problems and needs to be avoided under all circumstances - a read from the same memory address shall never result in different values. There are several approaches to deal with such an issue. A detailed study about memory consistency and cache coherency is given by Sorin et al. [57].

To copy a single chunk of memory, it is sufficient to configure the DMA with a source address, a destination address, and the length of the memory to be copied. When copying multiple blocks, the CPU needs to reconfigure the source and destination address after each successfully copied single block. This slows down the throughput of the DMA and interrupts the CPU quite frequently. To deal with the problem a so called scatter-gather (SG) mode is available. In the SG mode, all parameters necessary for a transfer are stored at a certain memory location accessible by the DMA. Each memory block to be copied has its own parameters, known as SG descriptor. If starting the DMA in SG mode, it will fetch the first descriptor, evaluate its parameters and starting a memory transfer upon validation. After a successful transfer, the next descriptor is fetched, and so on, until the last descriptor is reached. The method has two benefits: while the DMA is transferring data, the CPU can continue execution without interruption, and furthermore, the physical to virtual memory mapping is done at once for all descriptors, without the need to bother the MMU all the time.

2.6.7 Pipelined ADC

The successive-approximation (SAR), the sigma-delta ($\Sigma - \Delta$), the flash, and the pipelined type are the far most popular ADC-types. The trade-off between resolution and sampling rate assists in selecting the most suitable type. The SAR ADC is suitable for all multiplexed data acquisition systems, is easy to use and has no pipeline delay. The $\Sigma - \Delta$ ADC is in place for a variety of industrial measurements offering the highest resolution of the three types with an average sampling rate. For high speed sampling, the pipelined type is the way to go. With sampling rates greater than 100 MSPS and high SNR it is used in digital oscilloscopes, spectrum analyzers, communication applications, and consumer electronics. [58]

The pipelined type is used in the implementation and will be further explained in rough details. The pipelined ADC evolved from the subranging architecture (SUA) in the 1950s [59]. The SUA, in its 6-bit form, uses a sample-and-hold block (SHA) to temporarily store the analog input value, which is discretized by a first-stage 3-bit sub-ADC (SADC). Those first 3-bits (MSB) are converted back with an sub-DAC (SDAC) and the resulting value is subtracted from the SHA value producing a residue signal. This signal is again converted by a SADC to retrieve the LSBs.

Ideal, the residual signal will never exceed the range of the second SADC. But the residual signal might exceed the range of the subsequent SADC resulting in missing codes - this can be the consequence of an inaccurate or nonlinear SDAC. An increase of the second SADC to 4-bits allows the residual signal to be out of range. An additional error-correction stage handles a possible overflow.

The error-corrected SUA requires the original value to be stored in the SHA until all SADC and SDAC stages have finished their conversion, limiting the maximum allowable sampling rate. To achieve higher sampling rates, a pipelined architecture is used shown in figure 2.32. The residue signal is delayed and stored by additional track-and-hold stages (T/H) before each SADC/SDAC pair. This enables the conversion of a new sampling value, as soon the previous value passes the first T/H stage, leading to a much higher overall sampling rate. The pipeline structure introduces a conversion delay indicated in figure 2.32. Once the delay is elapsed, each clock-cycle gives a new sampling value. [58]

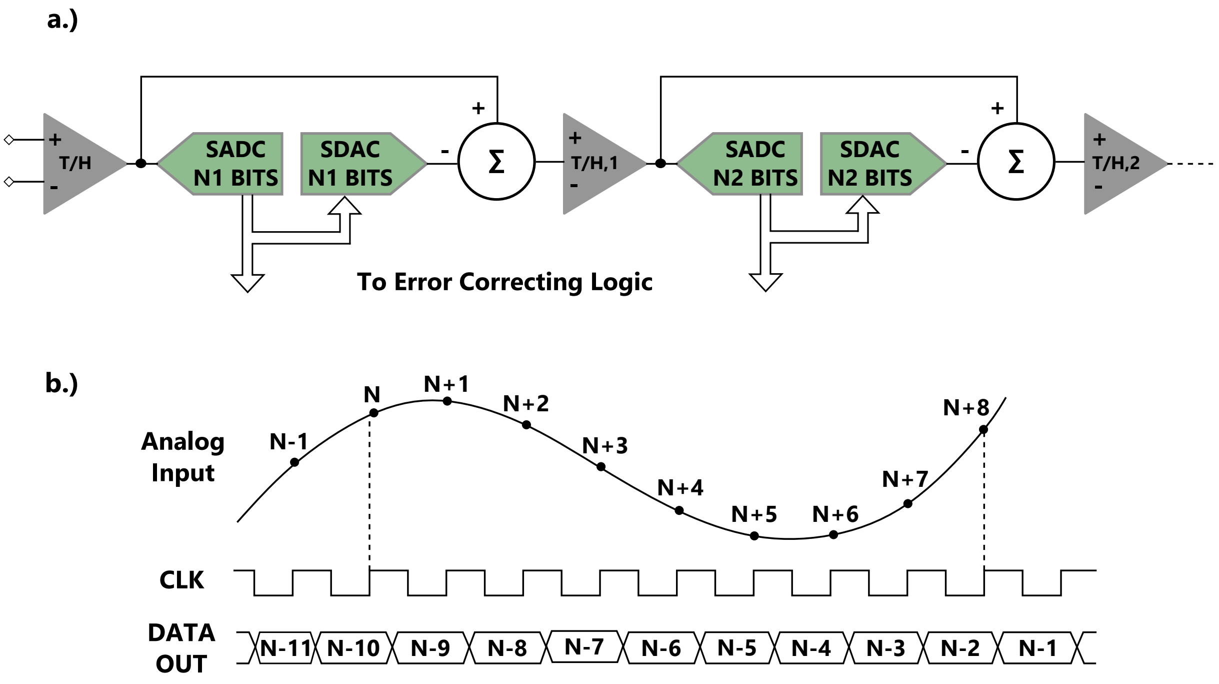


Figure 2.32: a.) Structural design of a subranging ADC. First stage SADC with second stage SDAC for back conversion of the discretized signal with a path towards the error correction. b.) Timing diagram of the SUA indicating the pipeline delay.

2.6.7.1 Signal Theory - The Sampling theorem

To properly reconstruct a continuous-time signal sampled by an ADC, it is necessary to comply with the sampling theorem.

Sampling theorem after Shannon: "If a function $x(t)$ contains no frequencies higher than W Hz, it is completely determined by giving its ordinates at a series of points spaced $T = 1/2W$s apart." [60]

The lower bound for the sampling rate is $\geq 2W$ samples per second, known as Nyquist sample rate, whereas W is called the Nyquist frequency. Violating this rule leads to aliasing effects - distortion of the signal due to undersampling. [60]

The sampling theorem was independently developed and proven in different engineering fields and is also known as Whittaker–Nyquist–Kotelnikov–Shannon sampling theorem. [60], [61]

2.6.8 Direct Digital Synthesizer

A Direct Digital Synthesizer (DDS) is a digital version of an analog oscillator used to create arbitrary waveforms. A common method to create such a waveform employs a phase accumulator and a LUT. The LUT stores discrete samples for a signal, like a

sinusoid. The phase accumulator generates successive addresses that are translated by the LUT to the desired output waveform.
DDS are heavily used in digital communication systems as modulators, mixers and local oscillators. [62]

2.7 Software

The previous section 2.6 covered topics from the hardware domain. This section will focus on the software domain. The Zynq platform, presented in section 2.6.4, is running an embedded linux operating system (OS). Due to the special hardware configuration (FPGA combined with ARM) the bootlfow needs adaption. The difference between the classical linux bootflow and the Zynq bootflow will be explained in detail in the following section. Beforehand the concept of the device tree will be elaborated. The device tree is used to tell the OS which hardware is present in the system.

2.7.1 Device tree

Before the device tree (DT) was introduced, the linux kernel contained the complete description of the hardware. It was only possible to supply some hardware parameters to the bootloader - even slight modifications required to rebuild the kernel. A separate binary, named DT blob (DTB) was established to split up the kernel from the hardware description. A DTB is created by a DT compiler out of a device tree source (DTS) file. [63], [64]
A DT is a data structure organized in nodes specifying each driver or module available in the system. Each node contains certain properties of the module or driver and might contain child nodes. [63]
The syntax of a typical node describing a serial port in DTS looks like the following:

A node starts with a name followed by its unit address (*node-name@address*). The *compatible* string is used to bind a device to a driver. The *reg* entry specifies the address within the address space of its parent node/bus followed by the length of the register set for the device. Detailed explanations of all other entries can be found in the DT specifications document [65].

```
/ {
    serial@e0001000 {
      compatible = "xlnx,xuartps";
      reg = <0xe0001000 0x1000>;
      status = "okay";
      clocks = <0x1 0x18 0x1 0x29>;
      interrupts = <0x0 0x32 0x4>;
      device_type = "serial";
      port-number = <0x0>;
    };
};
```

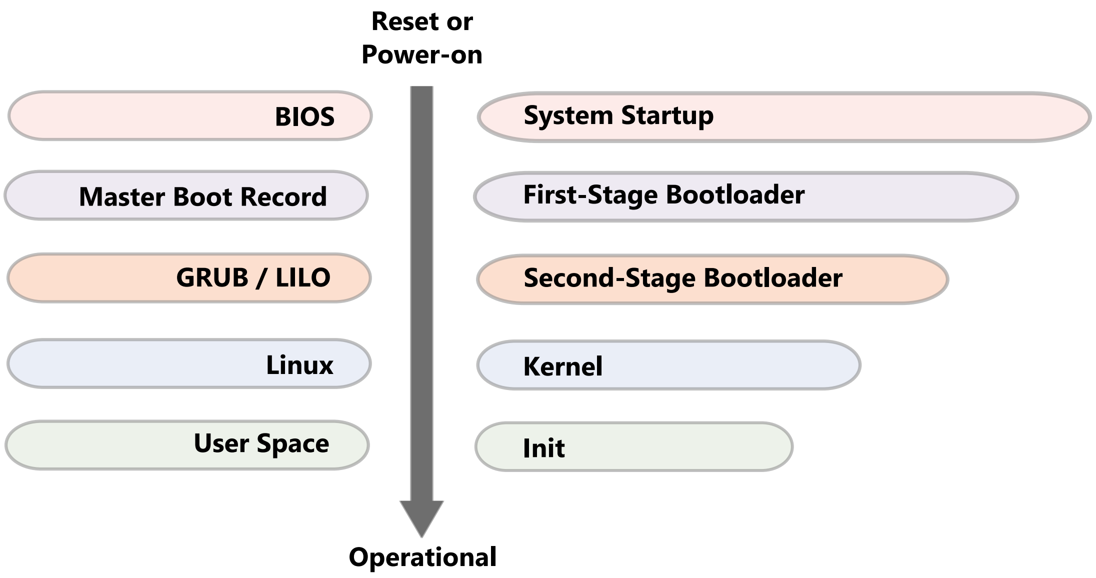


Figure 2.33: Traditional linux boot sequence starting from BIOS up to the user space. Adapted from [54].

2.7.2 Linux vs. Zynq Boot Sequence

2.7.2.1 Linux Boot Sequence

Figure 2.33 shows a traditional linux boot sequence (LBS). The BIOS performs initial system startup, configures hardware, and loads the first-stage bootloader (FSBL) into memory. The FSBL is only responsible to load the second-stage bootloader (SSBL). The SSBL detects available operating systems (OS) and decompresses the preferred one into memory, handing over the control to the kernel image, which can finally boot into user space.

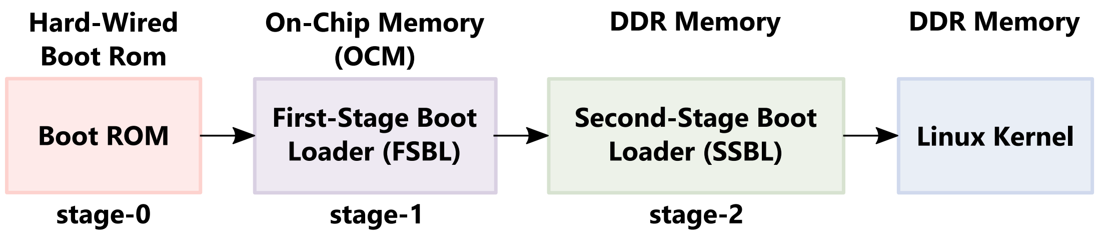


Figure 2.34: Zynq boot stages starting with hard-wired ROM code. Adapted from [54].

2.7.2.2 Zynq Boot Sequence

Figure 2.34 presents the Zynq boot stages which are similar to the LBS with some additional steps to fulfill embedded hardware requirements. In the first stage, the boot ROM evaluates the boot mode strapping pins to determine from which interface the FSBL will be loaded to the on chip memory (OCM). Once FSBL is in charge, it configures the CPU with PS data and optionally programs the PL with a bitstream retrieved from non-volatile memory. In the subsequent stage, the FSBL loads a SSBL such as the universal bootloader (U-Boot). Finally, the SSBL copies the compressed linux image, the DTB, and the ramdisk image into memory and initialize the kernel execution. [54]

The files present for a Zynq boot medium are illustrated in figure 2.35. The *BOOT.BIN* is a container file including the FSBL, the SSBL and an optional bitstream file. The DTB corresponds to the *devicetree.dtb* file. The *ramdisk8M.image.gz* is a temporary disk drive file which can be loaded into RAM and acting like a file system. The *zImage* is the compressed linux kernel. [54]

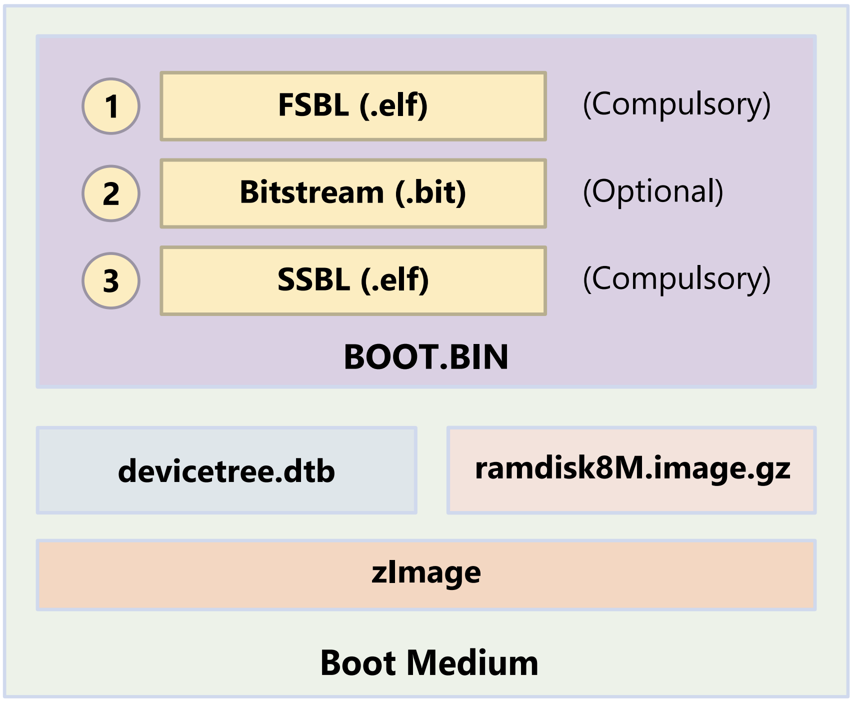


Figure 2.35: Zynq boot medium files. The bitstream file is optional - it is not mandatory to load FPGA content. Adapted from [54].

CHAPTER 3

Time-of-flight Simulation

The first part of this chapter presents different approaches how to simulate an ultrasonic wave propagation. The second part introduces a classical signal processing technique for flow determination and a new approach based on the echo energy.

3.1 Overview

The goal of the simulation is to obtain data for a time-of-flight (TOF) determination of an ultrasonic transducer pair in a moving medium. To get TOF data, two different approaches for ultrasonic wave propagation have been implemented. Based on the data obtained by the simulation a new method for flow determination in a pipe was evaluated.

3.2 Implementation

All implementations are based on Python. Additionally, a library named SciPy, an open-source python-based software designed for science and engineering is in use. SciPy brings many convenient math functions and a powerful plotting library with animation capabilities.

To get familiar with FDM algorithms, a one-dimensional FDM was implemented in the first place. A wave propagating in one-dimension corresponds a to a pulse on a rope which is fixed at both ends, moving forth and back. This very first approach is based on the

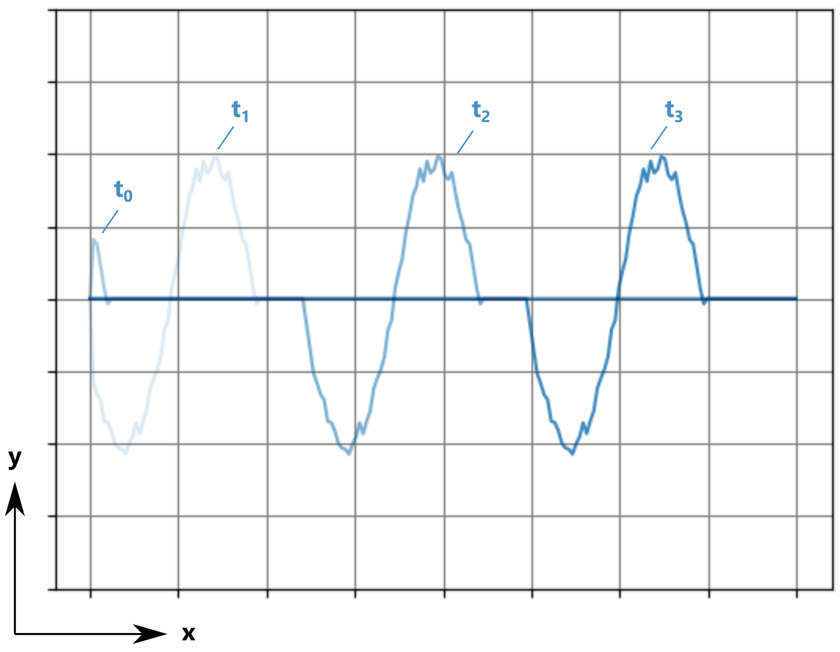


Figure 3.1: Pulse on a string: Propagation of a distorted sine wave of a one-dimensional FDM implementation at advancing time-steps t_x.

FDM represented by equation 2.48. Figure 3.1 shows the propagation of a distorted sine wave in space for different time-steps for the one-dimensional case.

The simulation concept is shown in figure 3.2. The left hand side shows a practical example with an insertion-type transducer pair. With the classical TTM, presented in section 2.3.2.1, two measurements are done consecutively to get t_u and t_d. Similarly, the simulation would require two sequential runs to obtain the necessary timing values to calculate the flow velocity. To save simulation time, the transducers T_1 and T_2 are replaced by a single sender S_x, and a receiver-pair R_u and R_d. This strategy halves the simulation time and enables the obtainment of the upstream-time and the downstream-time by a single simulation run.
Both receivers are spaced equidistant, by the distance L, to S_x under a certain angle Φ. The gas flows from top to bottom with the mean velocity v_f.
The simulation boundaries are freely configurable to be either absorbing or reflecting. To mimic a measurement in a virtually infinite stretched out pipe from top to bottom, the boundaries are configured with absorbing properties, since reflections are unexpected. The right boundary represents a rigid pipe wall and is therefore set to reflecting characteristics. Theoretically, the left pipe wall is expected to be rigid too, but needs special handling as described in the next paragraph.

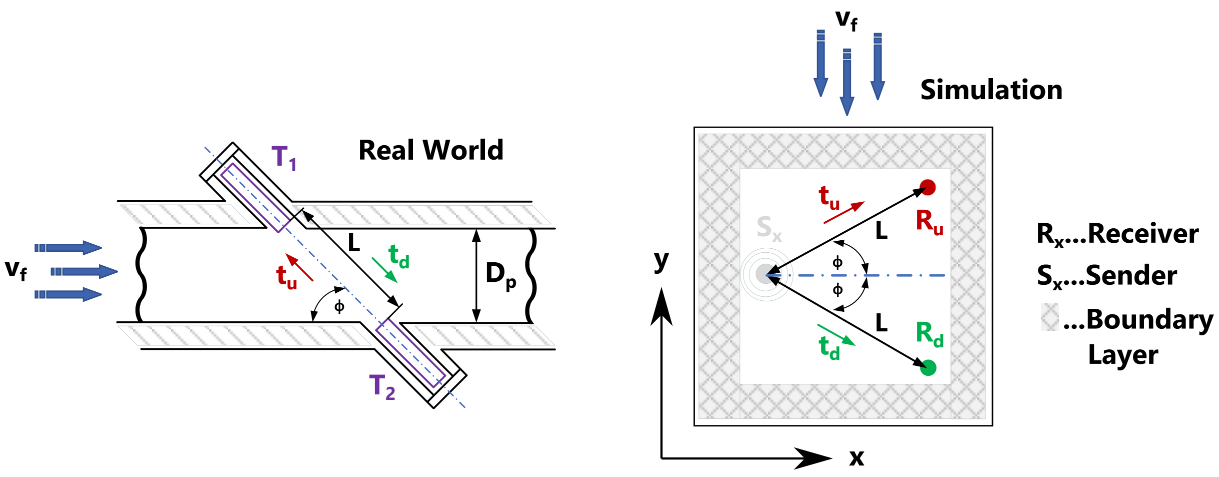


Figure 3.2: The sketch on the left hand side shows the typical arrangement of the transducers T_1 and T_2 for a transit time measurement in a pipe with diameter D_p. In analogy, the right figure represents the simulation concept with arrangement of the receiving transducers R_u and R_d and a point source S_x.

In practice, insertion-type transducers are mounted directly at the interface of the pipe wall and the flowing media, and do not experience or detect any reflections from its wall being mounted in. In the simulation environment, the sender can not be placed directly at the boundary of the grid, since it would interfere with the ghost-points (see section 2.5.4). To avoid unwanted reflections, S_x is placed a certain distance away from the left boundary layer which is configured to be absorbing in this case. The gas flow is not affected by the boundary layers at all and can pass them unchanged.
The following two sections describe the approaches implemented to simulate a wave propagation.

3.2.1 Grid-shifting Approach for Wave propagation Simulation

The grid-shifting approach is based on the one-dimensional implementation. The implementation of equation 2.48 is extended to support the two-dimensional case expressed by equation 2.49. The problem with equation 2.49 is, that it only models the wave propagation itself, without the influence of a flowing medium. None of the available literature deals with the wave equation in the second-order central FDM under influence of a moving medium. Therefore, a self-developed method for a moving medium have been combined with the two-dimensional wave equation, explained in the next paragraph.
In the simulation environment the gas flow is from top to bottom as shown in figure 3.2. Imagine a point source placed in the centre of the simulation environment under the in-

fluence of a gas flow. Emitted waves are spreading in all directions. Waves in the upwards direction are slowed down. Waves in the downwards direction are accelerated. A change of the velocity of the wave in the downwards direction is nothing else than a displacement on the grid. Assuming a plug flow, the displacement can be mimicked by a shift of the grid nodes in negative y-direction. Considering the flow velocity and the simulation step in space, a shift at proper time intervals can mimic a plug flow with specific velocity quite well. This shifting technique was implemented and is working accurately, with the limitation that only certain discrete step-sizes for flow velocity are applicable due to the connection of v_f to Δt and the grid spacing.
In reality, the gas flow is laminar and not plug like. All means to implement a laminar shifting technique, at which the central grid columns are shifted by a higher amount of steps compared to the outer ones, leaded to discontinuities between some of the neighbouring nodes. These discontinuities represent high frequency disturbances in the medium causing the central FDM to oscillate already at small v_f values.
Applying a laminar flow to this algorithm failed. This lead to the usage of another basic mathematical approach presented in the next section.

3.2.2 Staggered Grid Approach for Wave propagation Simulation

The staggered grid approach is based on the work of D. Wilson Keith and Lanbo Liu [66], published in 2004. First-order partial differential equations by Ostashev [67] and Aldridge [68] are the basis for this algorithm

$$\frac{\delta p}{\delta t} = \boxed{-(\boldsymbol{v}\cdot\nabla)\,p} - \boxed{\rho c^2 \nabla \cdot \boldsymbol{w}} + \boxed{\rho c^2 Q} \tag{3.1}$$

$$\frac{\delta \boldsymbol{w}}{\delta t} = \boxed{-(\boldsymbol{w}\cdot\nabla)\,\boldsymbol{v}} - \boxed{(\boldsymbol{v}\cdot\nabla)\,\boldsymbol{w}} - \boxed{\frac{\nabla p}{\rho}} + \boxed{\boldsymbol{F}/\rho} \tag{3.2}$$

where ρ is the ambient medium density, p is the acoustic pressure, w is the acoustic particle velocity, and $\boldsymbol{v}$ is the wind velocity. $\boldsymbol{F}$ and Q are used to model sources, whereat $\boldsymbol{F}$ acts as a force on the medium, and Q is a mass source. Symbols in bold represent vectors. The red surrounded expressions in equations 3.1 and 3.2 represent the shift of the sound wave by the wind, known as advection. The green surrounded expression is the influence on the particle velocity by spatial variations in the wind field. The blue surrounded terms model the influence of the medium density, and the black surrounded expressions are the two different types of sources available.

In the two-dimensional space, equations 3.1 and 3.2 can be written as

$$\frac{\delta p}{\delta t} = -\left(v_x \frac{\delta}{\delta x} + v_y \frac{\delta}{\delta y}\right) p - \left(\frac{\delta w_x}{\delta x} + \frac{\delta w_y}{\delta y}\right) \kappa + \kappa Q \tag{3.3}$$

$$\frac{\delta \boldsymbol{w_x}}{\delta t} = -\left(w_x \frac{\delta}{\delta x} + w_y \frac{\delta}{\delta y}\right) v_x - \left(v_x \frac{\delta}{\delta x} + v_y \frac{\delta}{\delta y}\right) w_x - b\frac{\delta p}{\delta x} + bF_x \tag{3.4}$$

$$\frac{\delta \boldsymbol{w_y}}{\delta t} = -\left(w_x \frac{\delta}{\delta x} + w_y \frac{\delta}{\delta y}\right) v_y - \left(v_x \frac{\delta}{\delta x} + v_y \frac{\delta}{\delta y}\right) w_y - b\frac{\delta p}{\delta y} + bF_y \tag{3.5}$$

where $b = 1/\rho$ is the mass buoyancy (or upthrust), which is a force contrary to the gravity, acting on an object residing in a fluid or a gas. The adiabatic bulk modulus, $\kappa = \rho c^2$, is defined as the ratio of an all-sided pressure on an object to its resulting volume change. Before applying the FDM discretization steps to equations 3.3-3.5, the concept of the staggerid grid will be introduced.

3.2.2.1 Staggered grid

A collocated grid is used in most discretization steps of various FDMs (introduced in section 2.5.2). The so called odd-even decoupling problem might occur when a uniform collocated grid is used with a central difference FDM. In such a situation, the update step for odd-numbered nodes, only uses information from even-numbered neighbour nodes, and vice versa. Without countermeasures, this decoupling will cause checkerboard like errors and in worst case leading to high-frequency oscillations. There are several techniques to avoid the decoupling problem, summarized in [69] and [70]. Here we focus only on the solution based on the so called staggered grid (SG).
On a SG, dependent variables are distributed to different nodes and not placed at one common grid point like it is the case for a collocated grid. To obtain a SG, the collocated grid is subdivided into one or multiple subsections, depending on the needs of the algorithm in use. As such, not all the variables need to be calculated for the same grid point.

The spatially staggered finite-difference grid for the staggered grid approach is shown in figure 3.3. The step size of the grid is halved, whereas whole-number nodes ($x = i\Delta x$ and $y = j\Delta y$) store the pressure values p. The variables v_x and F_x are stored at w_x nodes, being normal to the x-direction, staggered by $\Delta x/2$. Similarly, the variables v_y and F_y

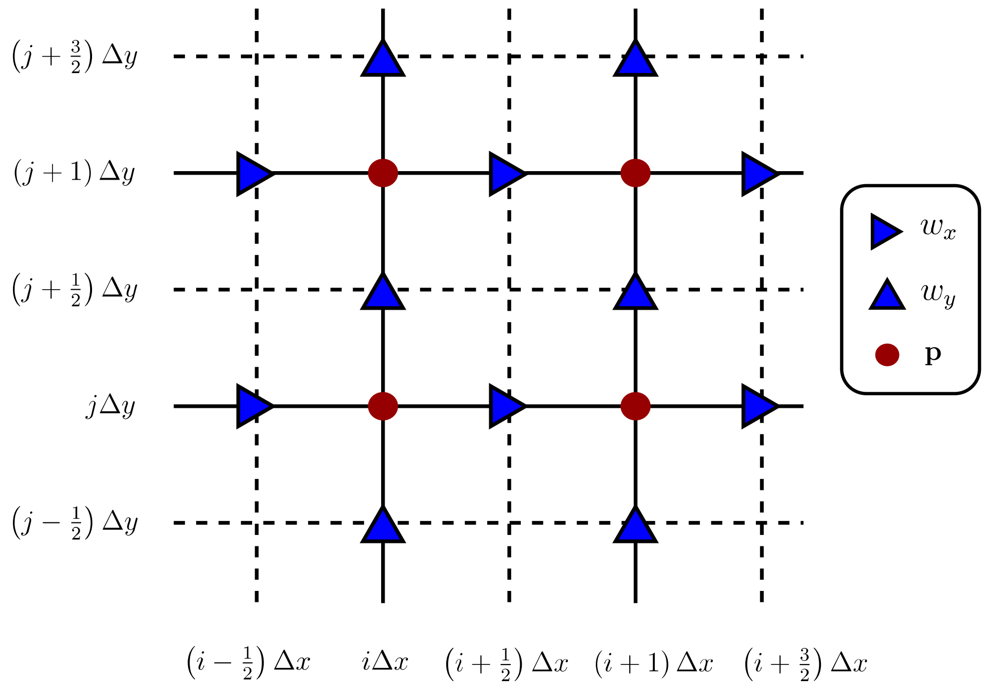

$\left(j+\frac{3}{2}\right)\Delta y$
$(j+1)\,\Delta y$
$\left(j+\frac{1}{2}\right)\Delta y$
$j\Delta y$
$\left(j-\frac{1}{2}\right)\Delta y$
$\left(i-\frac{1}{2}\right)\Delta x$
$i\Delta x$
$\left(i+\frac{1}{2}\right)\Delta x$
$(i+1)\,\Delta x$
$\left(i+\frac{3}{2}\right)\Delta x$
w_x
w_y
p

Figure 3.3: Staggered grid with halved step size.

are stored at w_y nodes, staggered by $\Delta y/2$ in the y-direction.
A direct consequence of the SG is, that the pressure nodes p can be calculated without the interpolation of the influencing velocity components. The essential advantage of the SG is that the pressure difference between two adjacent nodes are the driving strength for the velocity values located between these nodes, avoiding checkerboard errors.
Due to the usage of half-steps, the indexing within the algorithm becomes rather complex. Another disadvantage is the higher memory consumption needed to store empty node values. [71]

3.2.2.2 Approximation with the centered FDM

With the concept of the SG in mind, the derivatives from equations 3.3-3.5 are discretized with a central FDM staggered in space across two grid intervals, resulting in

$$\frac{\partial w_x[(i+1/2)\Delta x, j\Delta y, t]}{\partial x} \simeq \frac{w_x[(i+3/2)\Delta x, j\Delta y, t] - w_x[(i-1/2)\Delta x, j\Delta y, t]}{2\Delta x} \tag{3.6}$$

$$\frac{\partial w_x[(i+1/2)\Delta x, j\Delta y, t]}{\partial y} \simeq \frac{w_x[(i+1/2)\Delta x, (j+1)\Delta y, t] - w_x[(i+1/2)\Delta x, (j-1)\Delta y, t]}{2\Delta y} \tag{3.7}$$

$$\frac{\partial w_y[i\Delta x,(j+1/2)\Delta y,t]}{\partial x} \simeq \frac{w_y[(i+1)\Delta x,(j+1/2)\Delta y,t]-w_y[(i-1)\Delta x,(j+1/2)\Delta y,t]}{2\Delta x} \tag{3.8}$$

$$\frac{\partial w_y[i\Delta x,(j+1/2)\Delta y,t]}{\partial y} \simeq \frac{w_y[i\Delta x,(j+3/2)\Delta y,t]-w_y[i\Delta x,(j-1/2)\Delta y,t]}{2\Delta y} \tag{3.9}$$

and for the pressure updates expresses to

$$\frac{\partial p(i\Delta x,j\Delta y,t)}{\partial x} = \frac{p[(i+1)\Delta x,j\Delta y,t]-p[(i-1)\Delta x,j\Delta y,t]}{2\Delta x} \tag{3.10}$$

$$\frac{\partial p(i\Delta x,j\Delta y,t)}{\partial y} \simeq \frac{p[i\Delta x,(j+1)\Delta y,t]-p[i\Delta x,(j-1)\Delta y,t]}{2\Delta y} \tag{3.11}$$

Obviously, approximations like $\delta w_x/\delta y$ require w_y and v_y to be evaluated at grid node $[(i+1/2)\Delta x, j\Delta y]$. Although those nodes are not existing on the grid (see figure 3.3), they can be averaged by the closest neighbours with

$$\begin{aligned} &w_y[(i+1/2)\Delta x,j\Delta y,t] \simeq \\ &\quad \frac{1}{4}\{w_y[(i+1)\Delta x,(j+1/2)\Delta y,t]+w_y[i\Delta x,(j+1/2)\Delta y,t]+ \\ &\quad\quad w_y[(i+1)\Delta x,(j-1/2)\Delta y,t]+w_y[i\Delta x,(j-1/2)\Delta y,t]\} \end{aligned} \tag{3.12}$$

Similar it is done for w_x and v_x at nodes $[i\Delta x,(j+1/2)\Delta y]$.

3.2.2.3 Solution on a Grid Staggered in Space

For the derivative $\delta p/\delta t$ we define a function f_p

$$\frac{\partial p(i\Delta x,j\Delta y,t)}{\partial t} = f_p\left[i\Delta x,j\Delta y,\boldsymbol{p}(t),\boldsymbol{w}_x(t),\boldsymbol{w}_y(t),\boldsymbol{s}(t)\right] \tag{3.13}$$

where bold symbols represent matrices for all dependent variables, and $\boldsymbol{s}(t)$ represents a source term. Replacing the approximations for w_x, w_y, v_x and v_y in equation 3.1, f_p

expresses as

$$
\begin{aligned}
f_p\left[i\Delta x, j\Delta y, \boldsymbol{p}(t), \boldsymbol{w}_x(t), \boldsymbol{w}_y(t), \boldsymbol{s}(t)\right] \approx \\
&- \frac{1}{4\Delta x}\left\{v_x[(i+1/2)\Delta x, j\Delta y, t] + v_x[(i-1/2)\Delta x, j\Delta y, t]\right\} \\
&\quad \times \left\{p[(i+1)\Delta x, j\Delta y, t] - p[(i-1)\Delta x, j\Delta y, t]\right\} \\
&- \frac{1}{4\Delta y}\left\{v_y[\Delta x, (j+1/2)\Delta y, t] + v_y[i\Delta x, (j-1/2)\Delta y, t]\right\} \\
&\quad \times \left\{p[i\Delta x, (j+1)\Delta y, t] - p[i\Delta x, (j-1)\Delta y, t]\right\} \\
&- \kappa(i\Delta x, j\Delta y, t)\left\{\frac{w_x[(i+1/2)\Delta x, j\Delta y, t] - w_x[(i-1/2)\Delta x, j\Delta y, t]}{\Delta x}\right. \\
&\quad \left. + \frac{w_y[i\Delta x, (j+1/2)\Delta y, t] - w_y[i\Delta x, (j-1/2)\Delta y, t]}{\Delta y}\right\} \\
&\qquad + \kappa(i\Delta x, j\Delta y, t)Q(i\Delta x, j\Delta y, t)
\end{aligned} \tag{3.14}
$$

The derivatives of the particle velocities are replaced by functions as well

$$
\frac{\partial w_x[(i+1/2)\Delta x, j\Delta y, t]}{\partial t} = f_x\left[(i+1/2)\Delta x, j\Delta y, \boldsymbol{p}(t), \boldsymbol{w}_x(t), \boldsymbol{w}_y(t), \boldsymbol{s}(t)\right] \tag{3.15}
$$

and

$$
\frac{\partial w_y[i\Delta x, (j+1/2)\Delta y, t]}{\partial t} = f_y\left[i\Delta x, (j+1/2)\Delta y, \boldsymbol{p}(t), \boldsymbol{w}_x(t), \boldsymbol{w}_y(t), \boldsymbol{s}(t)\right] \tag{3.16}
$$

The detailed expressions for f_x and f_y can be found in [66].

3.2.2.4 Solution on a Grid Staggered in Space and Time

For a staggered central FDM in time, the pressure is stored on whole time steps, and the velocities at half time steps. Full time steps are noted as $l\Delta t$ and future half time steps expressed as $(l+1/2)\Delta t$. As such, equation 3.13 becomes

$$
\begin{aligned}
&\frac{\partial p[i\Delta x, j\Delta y, (l+1/2)\Delta t]}{\partial t} \\
&\quad = f_p\left[i\Delta x, j\Delta y, \boldsymbol{w}_x[(l+1/2)\Delta t], \boldsymbol{w}_y[(l+1/2)\Delta t], \boldsymbol{s}[(l+1/2)\Delta t]\right] \\
&\qquad \simeq \frac{p[i\Delta x, j\Delta y, (l+1)\Delta t] - p[i\Delta x, j\Delta y, l\Delta t]}{\Delta t}
\end{aligned} \tag{3.17}
$$

Solving for the future time-step $(l+1)\Delta t$ gives

$$
\begin{aligned}
p[i\Delta x, j\Delta y, (l+1)\Delta t] &= p[i\Delta x, j\Delta y, l\Delta t] \\
&+ \Delta t f_p\left[i\Delta x, j\Delta y, \boldsymbol{w}_x[(l+1/2)\Delta t], \boldsymbol{w}_y[(l+1/2)\Delta t], \boldsymbol{s}[(l+1/2)\Delta t]\right]
\end{aligned} \tag{3.18}
$$

D. Aldridge [1] proposed a central FDM storing two steps in time for the pressure field, which results in a future time-step $(l+1)\Delta t$ of p in

$$\begin{aligned} p[i\Delta x, j\Delta y, (l+1)\Delta t] &= p[i\Delta x, j\Delta y, (l-1)\Delta t] \\ &+ 2\Delta t f_p \left[i\Delta x, j\Delta y, \boldsymbol{p}(l\Delta t), \boldsymbol{w}_x(l\Delta t), \boldsymbol{w}_y(l\Delta t), \boldsymbol{s}(l\Delta t)\right] \end{aligned} \tag{3.19}$$

This expression requires the value of the velocities at whole time steps $t = l\Delta t$, which can be retrieved by

$$\boldsymbol{w}_x(l\Delta t) = \frac{\boldsymbol{w}_x[(l+1/2)\Delta t] + \boldsymbol{w}_x[(l-1/2)\Delta t]}{2} \tag{3.20}$$

and similar for $\boldsymbol{w}_y$. The update step for the velocities at half time steps is

$$\begin{aligned} w_x[(i+1/2)\Delta x, j\Delta y, (l+1/2)\Delta t] &= w_x[(i+1/2)\Delta x, j\Delta y, (l-3/2)\Delta t] \\ &+ 2\Delta t f_x \left[\begin{array}{l} (i+1/2)\Delta x, j\Delta y, \boldsymbol{p}[(l-1/2)\Delta t], \boldsymbol{w}_x[(l-1/2)\Delta t] \\ \boldsymbol{w}_y[(l-1/2)\Delta t], \boldsymbol{s}[(l-1/2)\Delta t] \end{array}\right] \end{aligned} \tag{3.21}$$

and

$$\begin{aligned} w_y[i\Delta x, (j+1/2)\Delta y, (l+1/2)\Delta t] &= w_y[i\Delta x, (j+1/2)\Delta y, (l-3/2)\Delta t] \\ &+ 2\Delta t f_y \left[\begin{array}{l} i\Delta x, (j+1/2)\Delta y, \boldsymbol{p}[(l-1/2)\Delta t] \\ \boldsymbol{w}_x[(l-1/2)\Delta t], \boldsymbol{w}_y[(l-1/2)\Delta t], \boldsymbol{s}[(l-1/2)\Delta t] \end{array}\right] \end{aligned} \tag{3.22}$$

The pressure at past-half-time steps is averaged by

$$\boldsymbol{p}[(l-1/2)\Delta t] \simeq \frac{\boldsymbol{p}[l\Delta t] + \boldsymbol{p}[(l-1)\Delta t]}{2} \tag{3.23}$$

The implementation of the algorithm including staggering in space and time is very tricky.

3.2.2.5 Stability Analysis

To guarantee stability, the maximum time step is restricted to

$$\Delta t < \frac{\Delta r}{u} \tag{3.24}$$

where $u = c + v_f$ is the total speed of propagation and Δr is the grid spacing on a two-dimensional grid defined as $\Delta r = 1/\sqrt{\Delta x^{-2} + \Delta y^{-2}}$. The CFL condition, introduced in section 2.5.5, becomes

$$CFL = u\frac{\Delta t}{\Delta r} < 1 \tag{3.25}$$

[1] Personal communication between authors

where Δt and Δr must be chosen in a way to fulfill equations 3.24 and 3.25 - this will satisfy stability criteria but does not imply accuracy.

In a uniform flow, u differs between the downwind direction $u = u_d = c + v$ and the upwind direction $u = u_u = c - v$. The resulting wavelength is shorter in the upwind direction, noted as λ_u, and defines the grid spacing as

$$\Delta r = \frac{\lambda_u}{N} = \frac{\lambda}{N}(1 - M) \tag{3.26}$$

where N is the number of grid points per wavelength, M is the Mach number (v/c) and λ is the wavelength (c/f). Obviously, an increasing value of M requires a finer grid. Combining equations 3.24 and 3.26 gives

$$\Delta t < \frac{\lambda_u}{N u_u} = \frac{1}{Nf}\frac{1-M}{1+M} \tag{3.27}$$

for the time step condition.

As mentioned in section 2.5.4, a PDE is fully described with proper initial and boundary conditions. A medium in rest under zero flow is chosen as an initial condition. The boundary conditions are described in the next section.

3.2.2.6 Boundary Conditions

The wall conditions were implemented as NBCs according to [41]. The amplitudes reaching the boundary strip with the width of B grid nodes are attenuated by a factor G

$$G = e^{-[\epsilon(B-i)]^2} \tag{3.28}$$

where ϵ is a constant, and i is the distance between the grid border and a grid node within B. The attenuation layer causes the wave to be weakened towards the grid boundary and as a consequence only slight reflections will appear. The layer will also cause multiple impedance changes of the medium - each node in B has a different impedance and thus, reflecting the incident wave back. Hence, there are two kinds of reflections, both influenced by ϵ. Large values for ϵ will cause stronger attenuation on the reflections from the grid boundary, but stronger reflections from the different impedance layers. The width B is mainly influencing the boundary reflections and has only slight influence on impedance layer reflections.

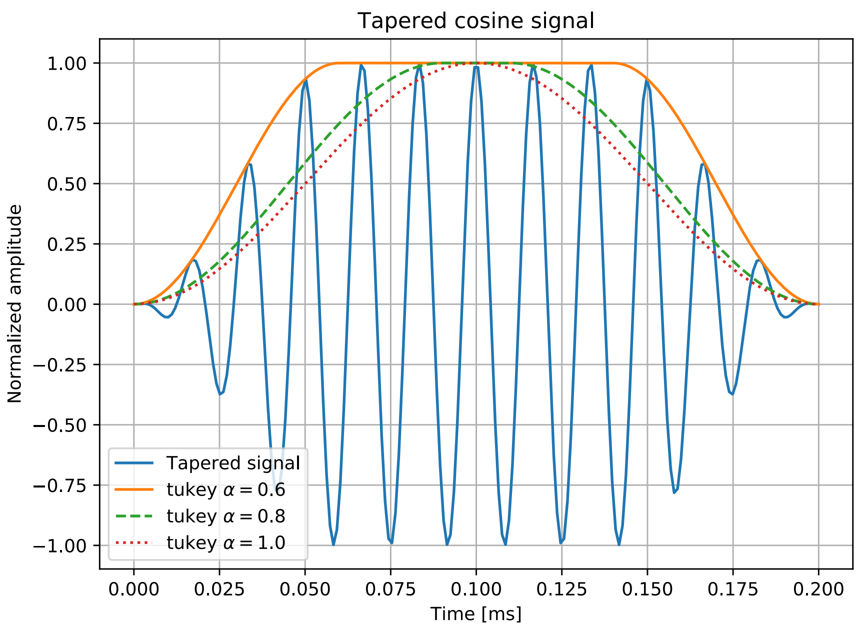


Figure 3.4: Tapered cosine signal for different α-values.

3.2.2.7 Source Modeling

To avoid numerical inaccuracies in simulation, sources in FDMs should not introduce abrupt changes to the medium. Ostashev and Symons [72] proposed to apply a cosine taper window on the source signal, which alleviates numerical dispersion of high frequencies. In case of a sinusoidal, the tapered source function is

$$Q(t) = \begin{cases} (A/2)\left[1 - \cos\left(\pi t/T_1\right)\right]\cos(2\pi f + \phi), & 0 \leq t < T_1 \\ A\cos(2\pi f + \phi), & T_1 \leq t \leq T - T_2 \\ (A/2)\left[1 + \cos\left(\pi(t-T)/T_2\right)\right]\cos(2\pi f + \phi), & T - T_2 < t \leq T \\ 0, & \text{otherwise} \end{cases} \tag{3.29}$$

where A is the source amplitude, Φ the source phase, T_1 and T_2 are the duration of the taper at beginning and end of the signal. SciPy has a built-in function, named *tukey* to create the tapered cosine function, with a parameter α to control the steepness of the window shown in figure 3.4. An α value of zero gives a rectangular window.

In the past sections all principles and conditions to implement the staggered grid approach for simulating a wave propagation have been elaborated. Considering stability criteria explained in section 3.2.2.5, boundary conditions from section 3.2.2.6 and modeling of the source explained in section 3.2.2.7, the fundamentals are in place and ready for the implementation part following in the subsequent chapter.

CHAPTER 4

Implementation

This chapter describes the details of the software implementation. First the introduction of the overall system architecture is given followed by the aspects in soft- and hardware. Critical paths and challenges of the system implementation are addressed in detail.

4.1 System Architecture

This section covers the high-level system architecture for all domains. The first domain deals with the structure of the mechanical construction. The mechanical structure is a laboratory setup with a short pipe to perform TOF and all sort of other experimental measurements. For signal generation, an ultrasonic transducer array is embedded into the pipe. Additional transducers are mounted for signal reception. A fan pushes air through the pipe. Basically, the setup mimics a gas flowing in an ordinary pipe with transducers measuring the flow rate.
The next two domains address the hardware structure that is split into an analog and a digital part. The last domain gives an overview of the software structure.

4.1.1 Mechanical Structure

A drawing of the mechanical structure is show in figure 4.1. The ultrasonic transducer array is mounted on a metal pipe of the length of 100 cm. Several indentations at different locations allow the placing of receiving ultrasonic transducers. A 50 W fan in combination with a flow straightener (see section 2.3.2.3) is installed at one end of the pipe to generate

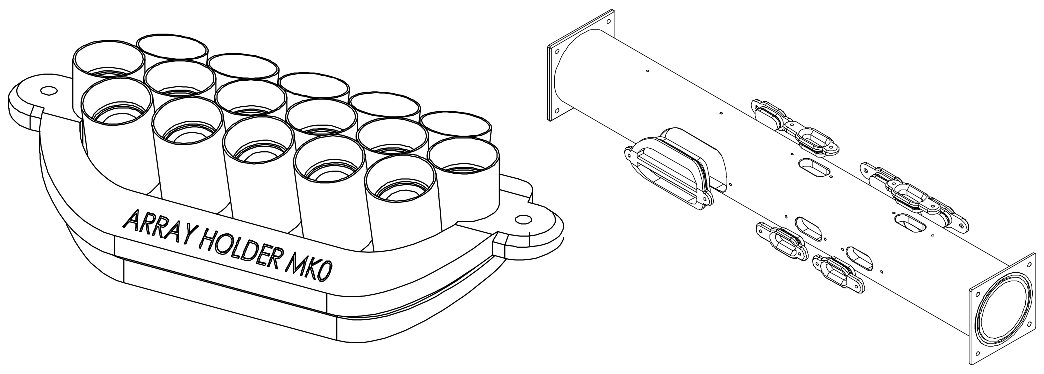


Figure 4.1: The mechanical form of the transducer array holder and the metal pipe used for measurements in the laboratory. (After R. Klambauer, A. Bergmann: *A new Principle for an Ultrasonic Flow Sensor for Harsh Environment.* 2017 IEEE SENSORS. Reproduced with permission.)

flow with an alterable velocity. The other end of the pipe is open. Beside the transducers, the pipe does not contain any obstacles or other things that might disturb the air flow.

4.1.2 Analog Module Structure

The high level module structure of the analog part is shown in figure 4.2. The FPGA generates digital signals that are transformed by a digital-to-analog converter (DAC). The converted analog signals are amplified in a two-way power-stage and driving the ultrasonic transducers. In total, sixteen transducers are in place, each being driven by its own amplifier stage and corresponding DAC.
The signal of the receiver is sensed by four receiving transducers. This analog sampling signal is converted by one of the four pipelined analog-to-digital converters (ADCs) of the type AD9237. The converted digital signal is wired to the FPGA.

4.1.3 High-level Digital System View

Figure 4.3 gives a simplified idea of the digital structure. The AD9237s are connected via the PL layer to the custom IP, named *ADC Sampler* (ADCS). The ADCS is connected as slave to the PS layer with an AXI-Lite interface. The data send back and forth from the slave interface is minor, therefore a lite version of AXI is sufficient.
The ADCS controls the AD9237s, retrieves the sampled data, and streams them to a DMA via an AXI-Stream as master. The DMA, acting in SG mode (see section 2.6.6), streams the data further to a memory controller in the PS layer, targeting a dedicated

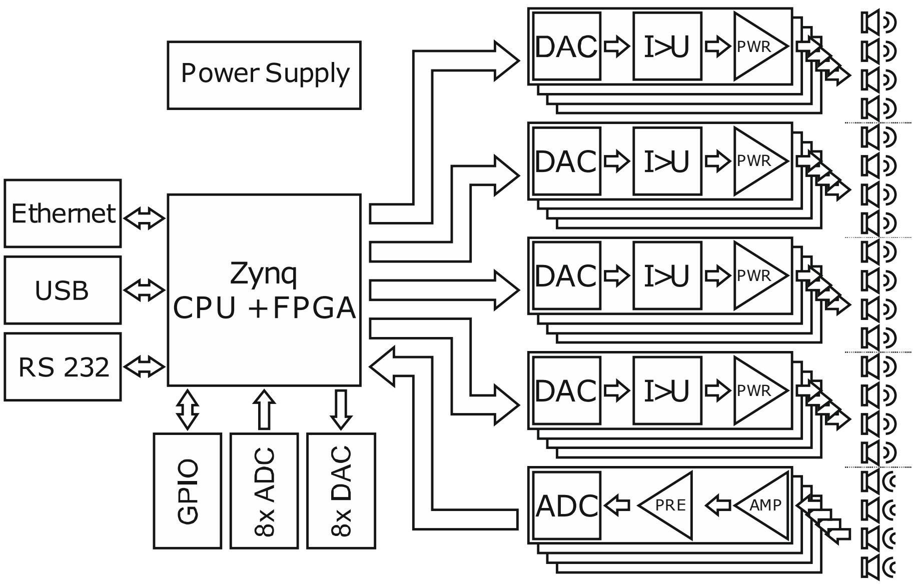


Figure 4.2: Schematic of the Zynq board with paths for ultrasonic signal generation (DAC with power-stages) and sensing (ADC with pre-amplification). (After R. Klambauer, A. Bergmann: *A new Principle for an Ultrasonic Flow Sensor for Harsh Environment*. 2017 IEEE SENSORS. Reproduced with permission.)

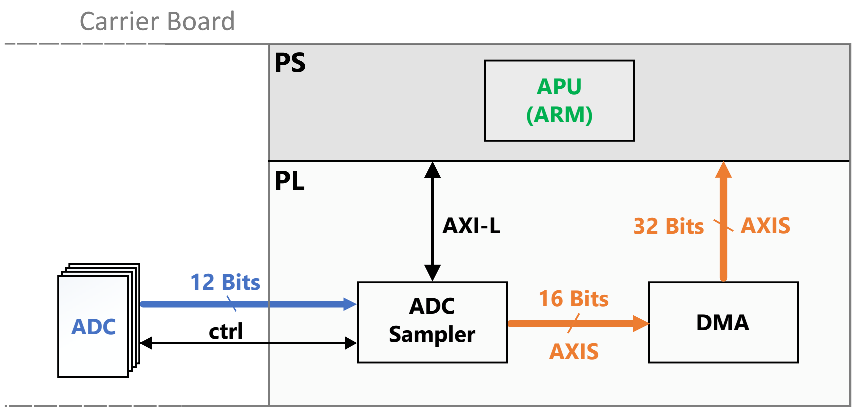


Figure 4.3: Simplified digital system view. The data-bus from the ADC is connected to the *ADC Sampler* in the PL layer of the Zynq. The sampled data is forwarded by a DMA to the PS layer. All PL modules are controlled by AXI interfaces.

RAM (DDR3) region used in a cyclic manner.

4.1.4 Software Application Structure

The software applications are split into two parts: (1) the application running on the Zynq, further referred to as the user space application (USAPP) and (2) the host application (HOAPP) running on a conventional windows or linux machine.
The communication between the two applications is separated into a command path and a data path. For the command path a reliable TCP connection is in use. For the data path a lightweight UDP connection is preferred, being better suited for streaming, but risking the chance of packet loss.

4.1.5 The Critical Paths

One critical path in the system is definitely the data transfer from the ADCS to the dedicated DDR3 memory region at high data rates. Subsequently, the data needs to be transfered via network to the HOAPP before the cyclic buffer wraps around, making it another critical path.
Before starting with the complete system design, various methods for both critical paths have been evaluated.

4.2 Software

This section describes the software part needed to run custom applications on the Zynq board. The Zynq board is driven by an embedded linux operating system described in the first place.

4.2.1 Xilinx PetaLinux

Xilinx provides a complete software suite, named PetaLinux Tools, to customize and deploy embedded linux solutions for their processing systems. The suite supports full configuration and building of a tiny linux kernel with everything that is needed to make it boot. Moreover, a rich support of drivers for Xilinx standard IPs and some useful third-party libraries like Qt is given. Qt is a free open-source software development kit for creating GUI and cross-platform applications written in C++. The HOAPP and the

USAPP are both written with Qt. The PetaLinux version used is *4.14.0-xilinx-v2018.3*.

4.2.2 Memory Layout

As mentioned in 2.6.6, the DMA engine only operates with physical addresses. Therefore, it is advised to reserve one physical contiguous block of memory, further referred to as scratch or reserved memory, for the DMA transfers. From user space (function *malloc*) it is impossible to allocate contiguous physical memory without the use of special kernel space drivers.
In the kernel space, two memory allocation functions are available: *kmalloc* and *vmalloc*. Only the former can reserve contiguous physical memory. *kmalloc* works on page-sized chunks usually 32 or 64 bytes in size. The page-size depends on the system architecture which also determines the maximum memory size allowed to be allocated at once. *kmalloc* is usually only suited to allocate memory up to 128 kilobytes (upper limit in some architectures). The main problem is the system memory fragmentation - after boot-up all kind of kernel space applications and drivers will try to allocate memory either with *kmalloc* or *vmalloc*, and the probability to get a large contiguous memory block of several megabytes decreases drastically after a certain system uptime. [64]

Figure 4.4 visualizes the desired physical RAM layout. There are a couple of options to obtain larger physical contiguous buffers accessible from the user space [64]. The following three options have been examined:

1. **Use *mem=* parameter to limit available system RAM**
   Linux accepts boot time parameters to supply the kernel with specific hardware informations or to provide special boot flags (see linux man pages 'bootparam' for a complete overview). One of the non-device specific boot arguments is the *mem* = parameter. This parameter can overwrite the amount of total installed system memory which is usually reported by the BIOS. If this parameter is modified, *fdt_high* and *initrd_high*, responsible for the location of the device tree and the initial root file system, needs adjustment too. On a parameter change, the DTB as well as the uboot needs to be rebuild.

2. **Use Scatter-gather Mappings**

The scatter-gather principle is already explained in section 2.6.6. For acquiring memory, accessible from user space, it is probably the most flexible of all three examined types in terms of variable memory space allocation. It does not come for free - the implementation requires a kernel space driver to convert the virtual addresses to physical addresses.
Using this method, there is no need to have a fixed predefined size of reserved memory in boot stage. The required memory can be allocated during runtime from user space. The acquired virtual memory addresses are passed to the kernel driver for conversion to physical addresses needed by the DMA.

A SG descriptor of the Xilinx AXI DMA v7.1 requires 40 Bytes of space to store information about a DMA transaction for a chunk of 16 kB of data. The memory requirements for the SG descriptor to address different amount of RAM are listed in table 4.1.

RAM size [MB]	Num. of Descriptors	Memory for Descriptors [kB]
16	1024	40
32	2048	80
64	4096	160
128	8192	320
256	16384	640
512	32768	1280

Table 4.1: Memory requirements for SG descriptors

3. **Dedicated Buffer in early Boot Stage via DTB**
   Similar to option one (*mem=*), a DTB entry can be used to reserve memory during boot-up. Listing 4.1 shows the device entries for such a reserved memory. The first entry is the storage for the SG descriptors, starting at address 0x37E00000 with a size of 2 MB, large enough to handle at least 512 MB of RAM (see table 4.1). The second entry is for the DMA scratch memory, starting at address 0x38000000 with a size of 128 MB.
   Both entries are using the *shared-dma-pool* keyword for the compatible property inside the *reserved-memory* node. This keyword invokes a special memory management driver. The *linux,cma-default* property let the operating system know, that this region shall be reserved by the contiguous memory allocator. [73]

```
reserved-memory {
  ...

  dma_desc@37E00000 {
    compatible = "shared-dma-pool";
    linux,cma-default;
    reg = <0x37E00000 0x200000>; };

  dma_buf@38000000 {
    compatible = "shared-dma-pool";
    linux,cma-default;
    reg = <0x38000000 0x8000000>; };
};
```

Listing 4.1: DTB entry to reserve memory for DMA usage.

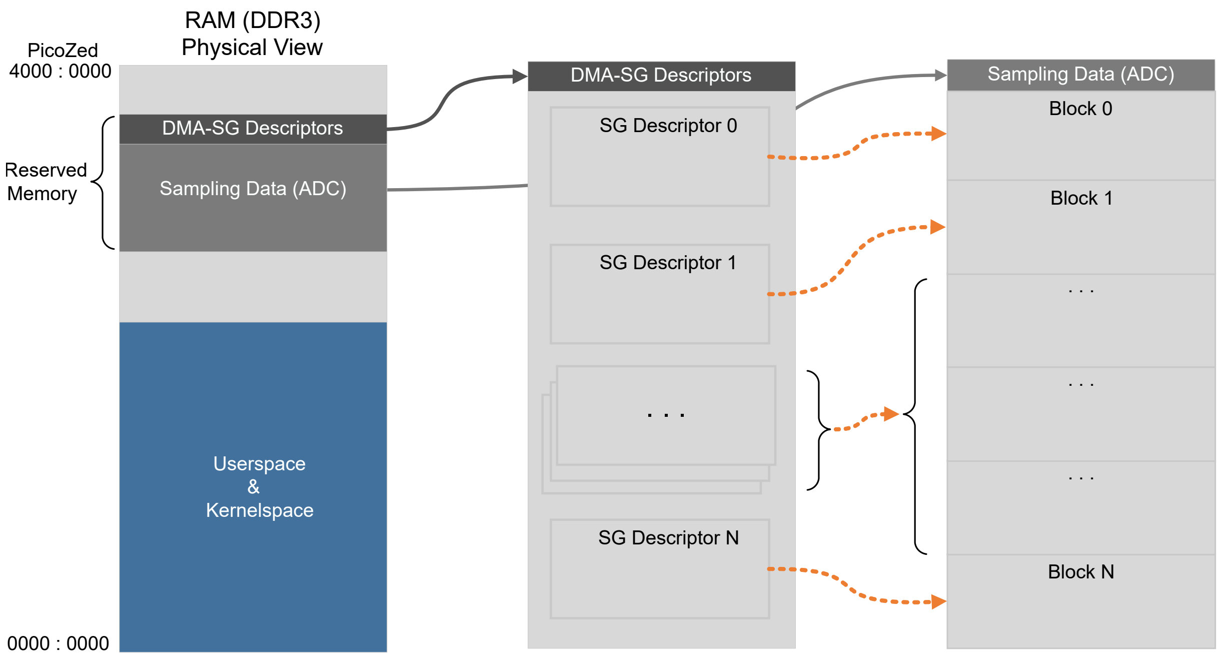


Figure 4.4: Memory layout showing reserved area for sampling data and SG descriptors necessary for the DMA.

4.2.3 Software application

Figure 4.5 shows the high-level block view of the HOAPP and the USAPP. They act on a client-server model. The HOAPP is the service requester (client), and the USAPP is the service provider (server). The communication between the client and the server is based on an Ethernet connection. The communication is split into a command-stream channel based on TCP, and a data-stream channel based on UDP.

On startup, the USAPP waits for an incoming client connection. A connection is established by some kind of handshake. A connecting client sends its version number to the server on the command-stream channel, which itself responds by its own version number. Once this is done, the data-stream channel is set up. The connection state is monitored by heartbeat packets.
The USAPP is also responsible to configure and control the IPs in the PL layer. Partly this is done by calling interface functions of the available kernel space drivers. The rest of the IPs are controlled by its static physical address. The system's physical memory can be accessed by *dev/mem/*, a character device file representing the main memory. It is a convenient way to directly address registers and memory of IPs. This approach is ideally suited for rapid prototyping, saving the needed to develop time-consuming and complex kernel drivers.

The HOAPP is responsible for the interaction with the user via a graphical user interface (GUI) shown in figure 4.6. Commands issued in this manner are translated and forwarded to the USAPP over the command-stream channel. The commands are encoded in JSON format - a format to encode object data, universally used for data exchange. The settings of all IPs in the PL layer can be configured in a visual manner in the GUI.

4.2.3.1 ADCS Data-stream

In this paragraph the software mechanism to meet the performance requirements for the data-stream channel is described. The data stream from the ADCS via DMA to the USAPP, and further to the HOAPP is a critical path.
The DMA in SG mode triggers an interrupt (IR) on completion of a packet transfer or if

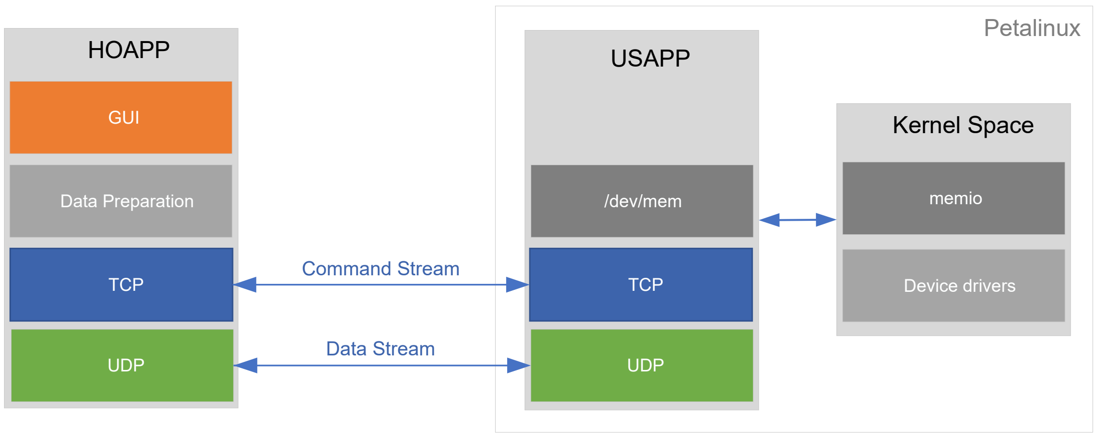
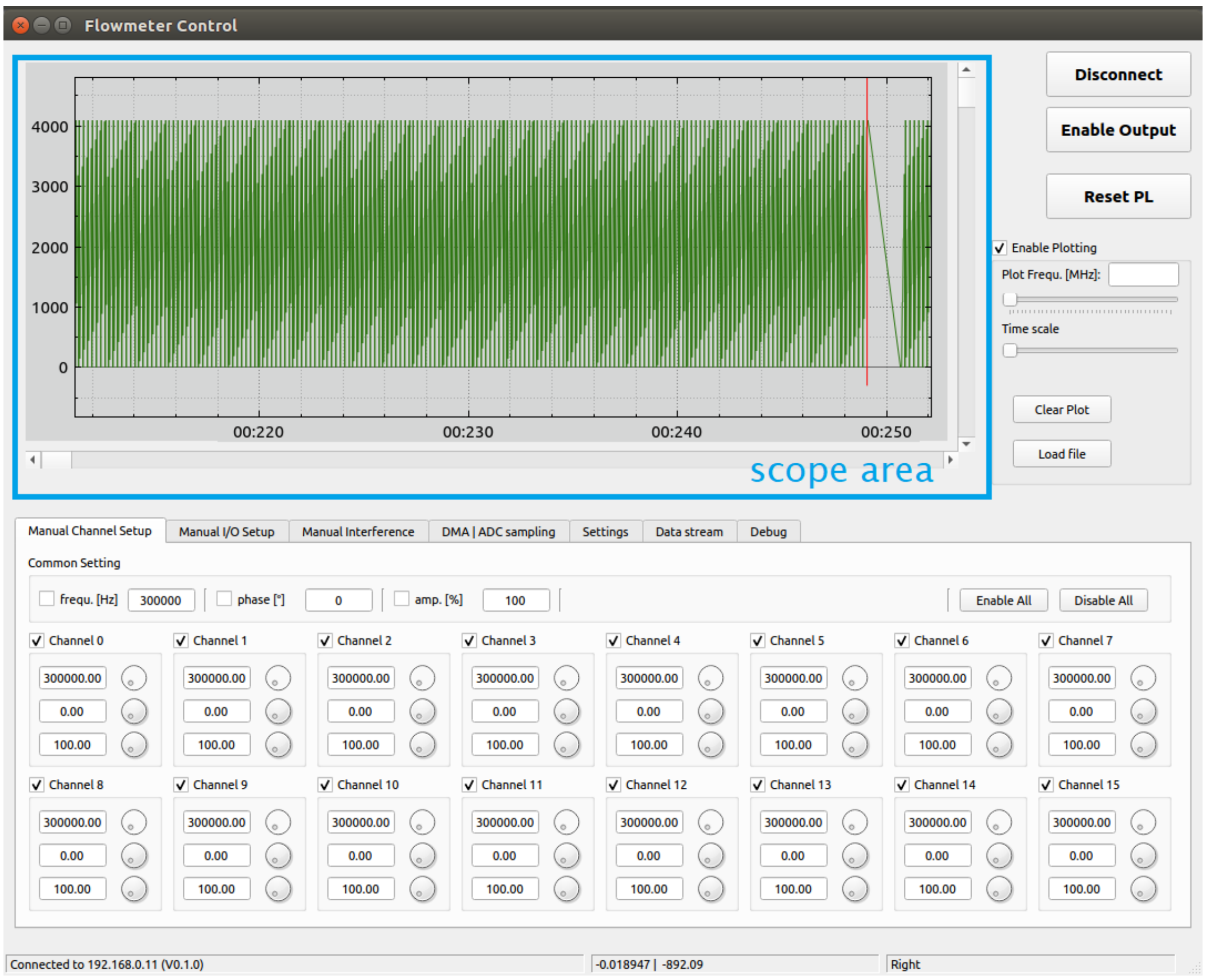


Figure 4.5: Block view of the host and user space application. The communication is based on an Ethernet connection.



Common Setting

frequ. [Hz] 300000 | phase [°] 0 | amp. [%] 100

	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
	300000.00	300000.00	300000.00	300000.00	300000.00	300000.00	300000.00	300000.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

	Channel 8	Channel 9	Channel 10	Channel 11	Channel 12	Channel 13	Channel 14	Channel 15
	300000.00	300000.00	300000.00	300000.00	300000.00	300000.00	300000.00	300000.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Connected to 192.168.0.11 (V0.1.0) | -0.018947 | -892.09 | Right

Figure 4.6: Graphical user interface of the host application in connected state with a sampled signal in green in the scope area. There are several GUI tabs with specialized functions for the single components. At this figure the *Manual Channel Setup* tab is active where settings for the sending transducers can be modified.

an error occurs. The kernel driver is invoked and clears the IR on demand. Theoretically, this is the hook for a user space application - waiting for an IR, evaluating the descriptor in charge, and processing the data in RAM which was transfered by the DMA. This implies a switch from user space to kernel space and back for each completed descriptor. Additionally, descriptor information needs to be copied from kernel to user space. The space switch and data copy takes time. Since we are designing for speed, the interrupt approach was switched by a polling approach.
Once a DMA transfer is initiated, the USAPP starts polling on the complete bit of the first descriptor. As soon the bit gets set, the associated memory location is transfered via network to the HOAPP. On transfer completion the complete bit is cleared again. Subsequently, the complete bit of the next descriptor is polled. This is repeated until the DMA stops transferring or an error occurs.
Beside the ADC data, the SG descriptors are placed in RAM too. As such, the user space has direct and fast access to the descriptors and also to their complete bits. Taking a higher CPU load into account, it turned out that the polling approach is faster compared to the IR approach.

4.3 Hardware

This section describes the step by step evolution of the system block design inside Vivado. Details about the ADCS IP development, registers in charge, and most important signals to interface the ADC are given. The section is starting with an introduction of the Zynq-board used for prototyping and for the final implementation.

4.3.1 Xilinx Zynq®-7000 System-on-Chip

The IP development was done on a ZedBoard, which is an evaluation board based on the Xilinx Zynq-7000 All Programmable System-on-Chip (SoC). With a memory of 512 MB DDR3, a swappable SD Card, essential connectors like USB, UART, Ethernet and JTAG, some push buttons, and LEDs for visual feedback makes this board ideal for rapid prototyping.
The final target was a PicoZed board that uses the same Zynq architecture like the ZedBoard, but comes with more memory (1 GB) and an on-board eMMC with 4 GB. The main difference to the ZedBoard is the lack of any physical connector since it is designed to be used with a carrier board only. Programming or any kind of communication is not

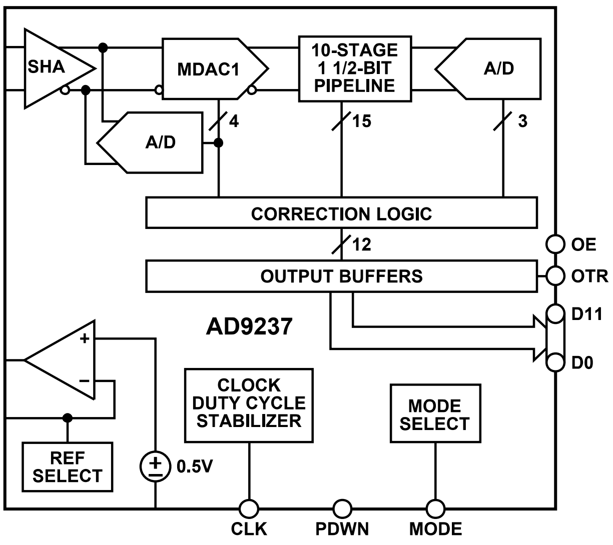


Figure 4.7: Functional block diagram - digital I/O lines only. (Simplified figure from ANALOG DEVICES: AD9237. Data Sheet Rev.C 2014)

possible without a carrier board. Specifications for the ZedBoard and PicoZed are found on the homepage of the manufacturer DIGILENT.

4.3.2 ADC AD9237 - Digital View

The digital ports for AD9237 are shown in figure 4.7. The power down pin *PDWN*, controls the power state of the ADC. By asserting the *PDWN* pin to low, the ADC will operate in normal mode. The output enable pin *OE*, manages the three-state ability of the AD9237. If the *OE* pin is high, the output data drivers are in high impedance state. Asserting the *OE* pin to low enables the output data drivers. The *OTR* pin indicates when the sampled signal is out of range.
The sampling rate is controlled with the clock input *CLK*. Sampled data appear on the digital output *D0:D11* after a pipeline delay of nine clock cycles, if the ADC is in normal operating mode and *OE* is driven to low. The AD9237 has a resolution of 12-bit and can convert twenty million samples per second (20 MSps) which corresponds to a data rate of thirty megabyte per second (30 MBps). As such, the minimum required speed on the data path channel is $\geq$ 30 MBps if targeting full speed conversion.
The ports of the AD9237 predefine I/O connections necessary on the ADCS. An output of the ADC corresponds to an input on the ADCS and vice versa.

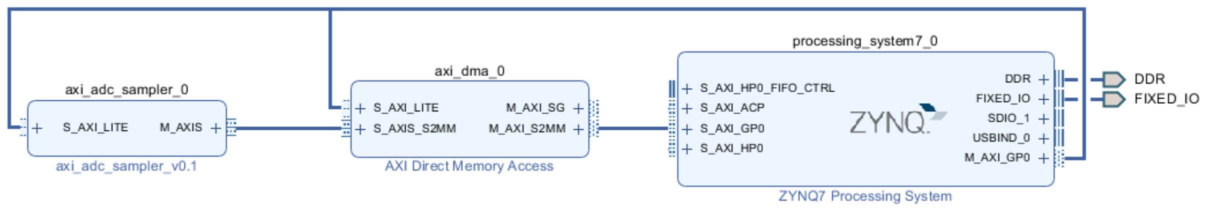


Figure 4.8: First stage of the ADCS evolution.

4.3.3 ADC Sampler and Zynq System Evolution

The Zynq system, as well as the ADCS IP was developed in several evolution stages. To get out of the critical path, the data stream from the ADCS to the DDR3 was implemented in the first place.
In the beginning, the ADCS was only acting as a 16-bit counter, simplifying the verification. The block structure of the first system design is shown in figure 4.8. Note that for all block designs only essential parts are shown in this chapter - for simplicity, smart connects and signals like clock and reset are omitted. In the first design, the master AXI port *M_AXI_GP0* connects the ADCS and the DMA to the Zynq PS. The ADCS streams the counter values via its master AXI streaming port *M_AXIS* to the slave port of the DMA, labeled as *S_AXIS_S2MM*. In the beginning, the DMA engine was working in direct register mode. In this mode, there is no need to supply SG descriptors. The DMA registers contain the destination address and length of the streaming data. The DMA streams the received data further to the slave port of the Zynq PS *S_AXI_GP0*.

Although the AD9237 has only a data width of 12-bits, the bus width of the ADCS streaming interface was increased to 16-bits to encode additional information. Beside the 12-bit data from the ADC, an additional OTR bit indicates an out of range detection. Two more bits (ADC_SEL) are reserved to encode from which of the four ADCs the data is retrieved. Bit number twelve is reserved for future use and is always set to zero. The bit fields are visualized below and further on referred to as ADC packet (ADCP).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OTR	ADC SEL		0	ADC DATA											

Initially, all IPs were clocked with the same frequency. The counter mode of the ADCS and equal data clocks eased the verification of the data stream. In the next stage, the

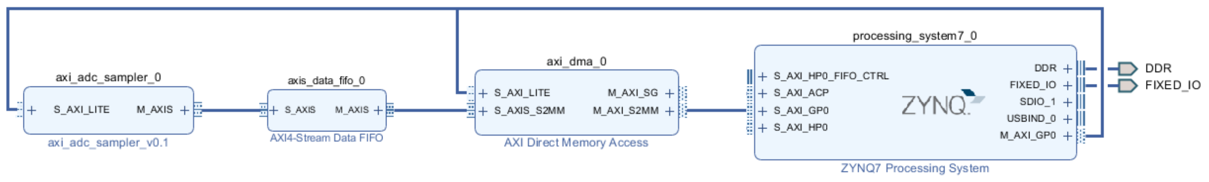


Figure 4.9: Second stage of the ADCS evolution: A FIFO is added for clock synchronization, acting as a data-buffer between the slow input clock and the fast clock of the DMA.

challenge with the two different clock domains was addressed. The ADCS needs to handle two different clockings, one for the slave AXI-L, and one for clocking and receiving data from the ADC. The AXI clock frequency for the reamainging IPs in the system is set to 100 MHz. The maximum frequency for the AD9237 is 20 MHz, which also corresponds to the data-rate on the streaming interface.
A clock domain crossing mechanism is needed to connect the streaming master to the DMA slave interface, running at a five times higher speed. Crossing clock domains is always critical - data loss and high current flow are only some of the effects appearing without proper synchronization [74].
A dual-clock first-input first-output (FIFO) architecture is the way to go for transfering data between crossing clock domains. A FIFO supports correct transfers between two modules producing and consuming data at different clock cycles. Vivado has a built-in IP, named *AXI4-Stream Data FIFO*, supporting multiple clocks with independent read-/write access. Internally, a block RAM structure with up to 32 kB gives enough space to temporary buffer sampled data. The simplified block design for the second evolution stage is presented in figure 4.9. Additionally to the AXI ports, the FIFO includes status signals giving feedback to the ADCS. For simplicity these signals are omitted in the block design. The status signals of the FIFO and their description are listed in table 4.2.

Signal	**Description**
s_fifo_empty_threshold	FIFO empty flag with configurable threshold
s_fifo_full_threshold	FIFO full flag with configurable threshold
s_fifo_wirte_data_count	32-bit counter for written data
s_fifo_read_data_count	32-bit counter for read data

Table 4.2: FIFO status signals.

The FIFO starts streaming as soon it is full or if $TLAST$, a special signal line on the incoming AXI interface, is asserted. In practice, the FIFO shall never overrun - data loss

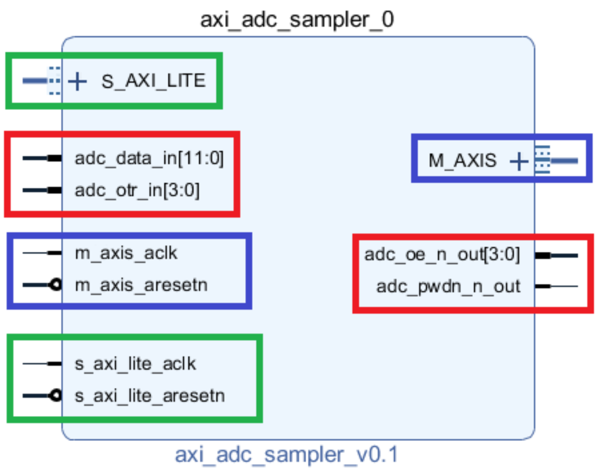


Figure 4.10: ADCS-IP with dual clock capabilities and ADC I/O-signals. Interfaces with equally coloured boxes belong together.

and decreased throughput will be the consequence. To prevent an overrun, the ADCS keeps track of the streamed data amount and monitors the FIFO status signals.
The streamed data from the ADCS is divided into bursts. Each burst consists of multiple ADCPs. *TLAST* is asserted after each burst, forcing a flush of the data stored in the FIFO. The amount of total bursts and the number of ADCPs within a burst are configurable via registers in the ADCS.

In the next evolution stage the ADCS was enhanced to interface the AD9237 with all its digital signals (signals shown in figure 4.7). Figure 4.10 illustrates the ADCS with dual-clock capability and all signals for the ADC interfacing.

At this point in time, the DMA was still operating in direct register mode. A switch to SG mode requires the *M_AXI_SG* port of the DMA to be connected to a memory port to retrieve SG descriptors. In the example design of the product guide of the DMA [75], the SG interface is connected to a block RAM (BRAM). The BRAM contains the values for the SG descriptors. The BRAM can be configured as dual port RAM, allowing simultaneous read and write from memory. One port is connected to the SG port of the DMA for reading SG descriptors. The other port is driven by the Zynq PS for writing the descriptor values. The block diagram for this configuration is shown in figure 4.11.

The memory requirements to store the SG descriptors are listed in table 4.1. The max-

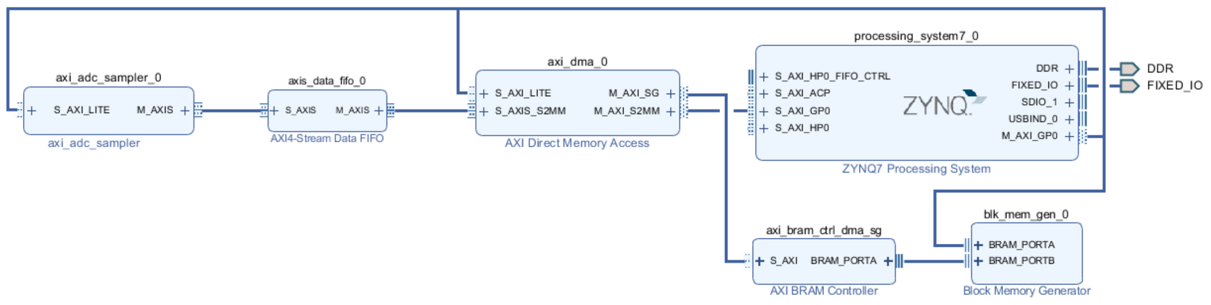


Figure 4.11: Next evolution stage: DMA configured to SG mode fetching descriptors from a BRAM.

imum available BRAM size for the Zynq 7z015 is 380 kB [76]. A reserved RAM size of 128 MB would already require 85 % of the available memory resources of the PL layer for the SG descriptors. Occupying nearly all memory resources restricts future enhancements and modifications of the system. To counter this potential limitation, the BRAM was removed and the *M_AXI_SG* port connected to the PS layer targeting DDR3 memory. There, additional space is reserved for the SG descriptors. Compared to the BRAM in the PL layer, there is plenty of space in the RAM of the PS layer, offering enough space for descriptors. The disadvantage of this method is the loss of the exclusive access to the SG descriptor location. In case of the BRAM, the DMA engine can fetch the SG descriptors without sharing the bus with any other IP. The connection to the RAM implies sharing the data path with the PS and the PL layer when fetching descriptors, taking a potential delay into account.

To minimize access time and prioritize descriptor fetching, the *M_AXI_SG* port is connected to the High Performance port *HP0* of the PS system. The high performance AXI interface allows priority management and high data rate communications. Although the data fetched per descriptor is limited and the available bandwidth of *HP0* is not outbid at all, the benefit lies clearly in the decreased access time due to configurable higher priority.

At this implementation stage the data path between ADCS and RAM was fully operable. The verification was based on the memory content in the RAM with ADCS operating in counter mode.

As soon the RAM content was transfered via network, memory seemed partially corrupted on the receiving end of the USAPP - noticed by missing counter values in the data sequence. At the very first glance, the corrupted memory seemed to be a consequence

of loosing data packets due to the usage of UDP as a network transferring protocol. Wireshark, an open-source network protocol analyzer application was used to inspect the network traffic to exclude faults in the USAPP. After extensive examination it was clear that the packet loss of UDP is only partial responsible for the faulty data. Jumps of the counter value in multiples of the data packet size occur in such a case. The rest of the faulty data had another origin explained in the next paragraph.
As mentioned in section 2.6.6, the DMA is not aware of the caching state of memory locations. This fact has led to faulty counter values inside the data stream. Data from the PS was still cached, while the DMA was updating the memory location in the background leading to cache coherency problems.
The Zynq PS features the Accelerator Coherency Port (ACP), having similar features as the *HP0*, additionally allowing cache-coherent transfers from IPs in the PL layer to main memory. The data stream coming from the DMA is connected to this ACP port of the PS to make the transfer cache aware. Additionally to this connection, two signals of the AXI write channel need to be set properly. Namely this is *AWCACHE* and *AWPROT*. *AWCACHE* controls the transaction type (normal, non-cacheable, modifiable, or bufferable). *AWPROT* specifies additional data attributes for the transaction. According to [56], the recommended binary value for *AWCACHE* is 0011 and 000 for *AWPROT*. This will generate a normal, non-cacheable, bufferable and modifiable transaction.
The connection of the SG port to *HP0* and the DMA stream connection to *ACP* is shown in figure 4.12. The recommended values for *AWCACHE* and *AWPROT* are tied to constants at the AXI smart connect between the DMA and the PS.
Despite the effort getting a cache coherent data stream, the issue with the corrupted data still remained. It seemed likely that either the ACP port or the PS layer is not properly handling the *AWCACHE* signals. Facing the supplier XILINX with this issue remained unanswered.
The state with the corrupted data was unacceptable. *AWCACHE* is the only variable in the system controlling the caching of the ACP port. An outtake of possible *AWCACHE* values is given in table 4.3. The complete list and description of all memory type settings can be found in [56]. The ADCS was extended with additional output signals to dynamically change the value of *AWCACHE* and *AWPROT*. Systematically, all meaningful values for *AWCACHE* were examined to check if the corrupted data is gone. It turned out that *0111* as a value for *AWCACHE* was solving the issue. The third bit of *AWCACHE* controls the read-allocate of a write-transaction. When asserted, the location of

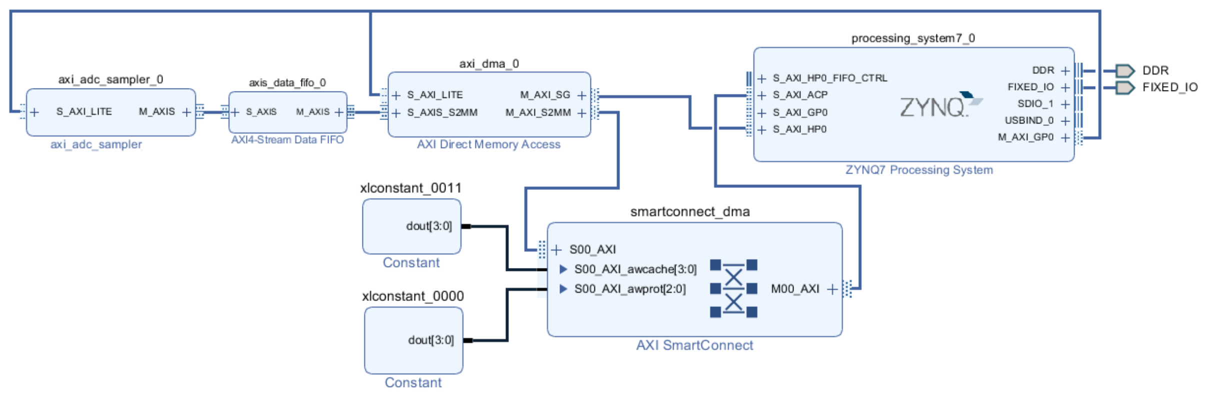


Figure 4.12: Simplified block design considering caching with *AWCACHE* and *AWPROT* control lines.

a subsequent read-transaction needs to be checked in the cache in first place, preferred to the backing store location. Using these setting, excessive long-term testing did not show a single corrupted value anymore.

AWCACHE[3:0]	Memory type
0000	Device Non-bufferable
0011	Normal Non-cacheable Bufferable
0111	Write-back No-allocate

Table 4.3: Memory type encoding

The fully evolved ADCS module with all inputs and outputs is shown in figure 4.13. In this final ADCS version 1.0, the FIFO status lines have been added as inputs. *s_awcache* and *s_awport* are added as outputs to control the caching mode.

The design for the signal generation, driving the ultrasonic transducers, was already existing. It consists of a custom IP (*signalgen_ctrl*), responsible to control four DDS (see section 2.6.8) units generating the digital values for the DACs of the type AD9707. The AD9707 requires differential clocking. This is established by a clocking wizard in combination with utility buffers. The utility buffers are configured in a way to produce differential signal pairs. The final block design is shown in figure 4.14.

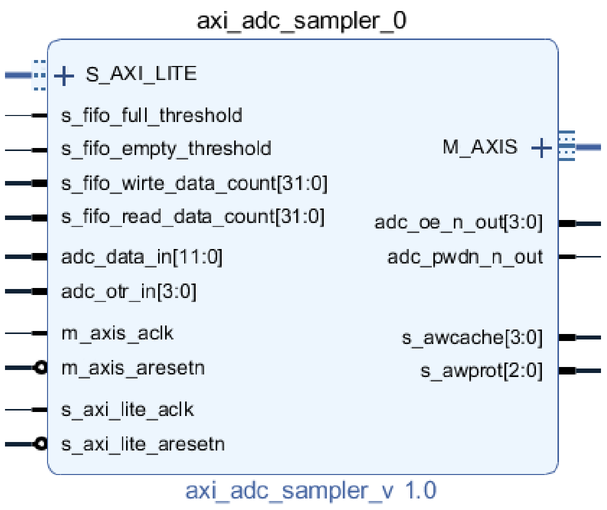


Figure 4.13: Final ADCS stage with all I/O signals including dual clock capabilities, ADC I/O-signals and cache control lines.

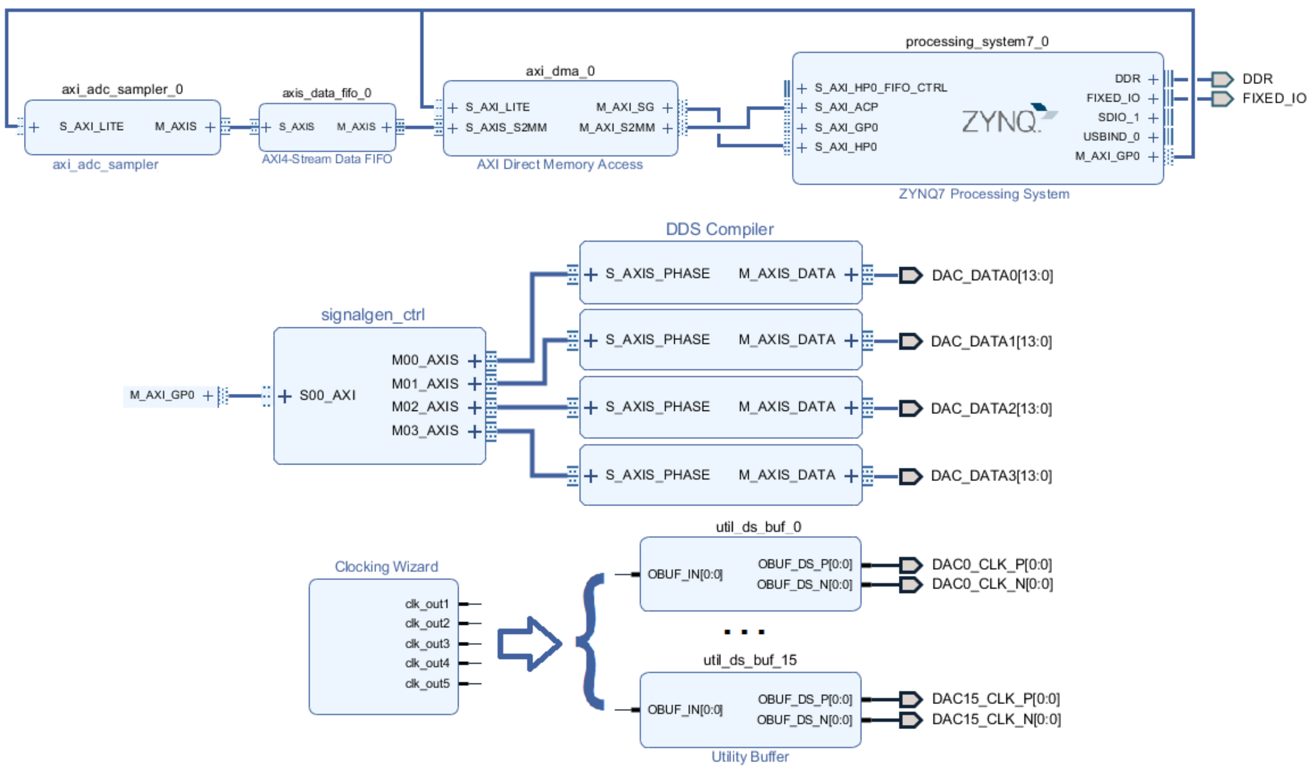


Figure 4.14: Final evolution stage: merged block diagram with ADC sampling path and signal generation with *signalgen_ctrl*.

CHAPTER 5

Results

This chapter concludes with the results of the master thesis. It covers the outcome of the comparison between two different approaches to determine the flow rate based on the implementation of the simulation presented in chapter 3. The chapter completes with the results on the software implementation evolved in chapter 4.

5.1 Simulation

This section presents the results on the flow rate measurement based on the cross-correlation method (CCM - see section 2.4.3) and the echo energy method (EEM - see section 2.4.4). The section starts with the results of the TOF wave propagation of both algorithms described in section 3.2.

5.1.1 Grid-shifting approach

In the grid-shifting approach the algorithm from section 3.2.1 is in use. Figure 5.1 shows a propagating wave originating from a sinusoidal point source with a frequency of 20 kHz placed at the middle-left side. The dimensions of the simulation area is 60 cm × 60 cm. The dashed line is only an aid for orientation. The coloured dots represent receiving transducers. The left boundary is configured to be absorbing. All other boundaries have reflecting properties. The wave propagates uniformly in all directions. A wave through is coloured yellowish, a wave crest is coloured dark-purple. The medium is steady, there is no flow (v_f =0 m/s).

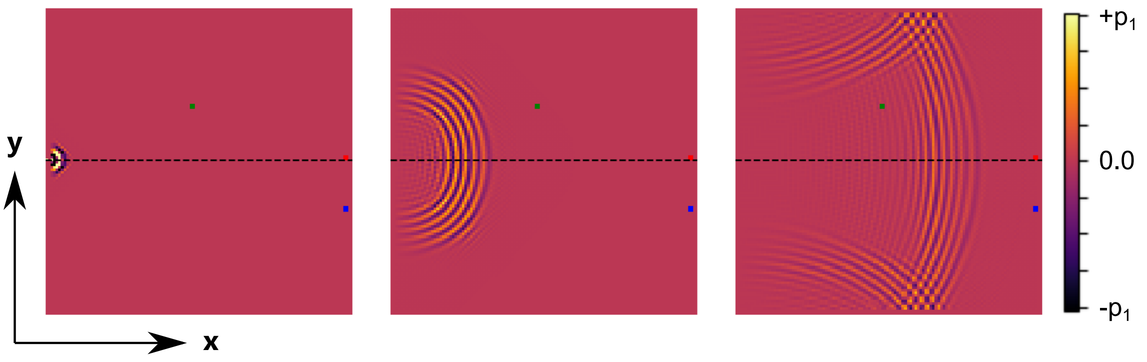


Figure 5.1: Grid-shifting approach: Propagating wave in steady medium. Arbitrary pressure amplitude p_1.

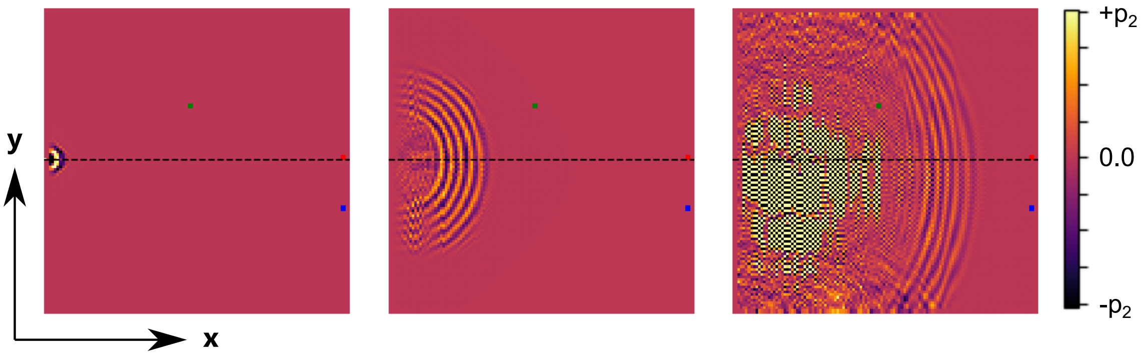


Figure 5.2: Grid-shifting approach: Oscillations under flowing medium - useless simulation data. Arbitrary pressure amplitude p_2.

Figure 5.2 indicates the failing approach to mimic a flowing medium in combination with a two-dimensional wave equation. The self-developed shifting technique, described in section 3.2.1, introduces disturbances and lead to oscillations. The oscillations diverge over the complete simulation area and make the obtained data useless.

5.1.2 Staggered Grid approach

The staggered grid approach uses the algorithm from section 3.2.2. Figure 5.3 shows a propagating wave originating from the centre of the simulation area at three different time steps. The wave is under an influence of a moving medium (gas flow) from top to bottom with v_f=60 m/s. Obviously, the wave propagating towards the top border is slowed down (compressed). The wave propagating towards the bottom is accelerated by v_f (stretched). This algorithm is used to gather the data for the TOF measurements.

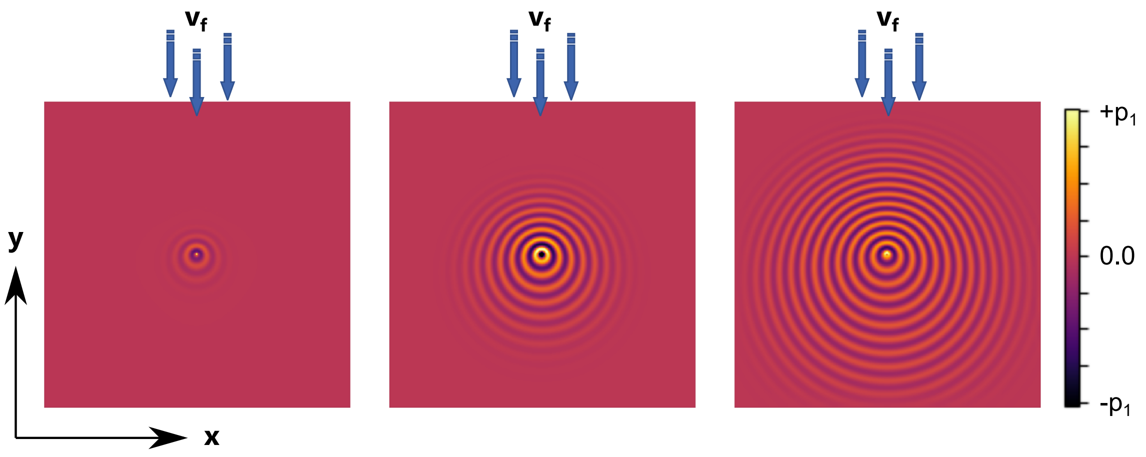


Figure 5.3: Staggered Grid approach: Propagating wave under the influence of a laminar gas flow from top to bottom. Arbitrary pressure amplitude p_1.

5.1.3 Correlation vs. Echo Energy Method

The data for the flow rate measurement is obtained from the simulation-results gained by the implementation of the algorithm described in section 3.2.2.
Recalling the simulation concept shown in figure 3.2, the settings for the simulation are listed in table 5.1.

Type	Value	Description
Source	60 kHz	Sinusoidal point-source, tapered at begin and end.
Receiver	L=10 cm $\Phi = 37°$	Parameters for receivers R_u and R_d distance and inclination to source.
Simulation	Δt=229.8 ns $\Delta x = \Delta y$=256.2 µm c=346 m/s κ=1.01 × 10^5 Pa ρ=1.1839 kg m^{-3} Runtime: 0.8 ms	Simulation parameter with time-step, grid distance, speed of sound, adiabatic bulk-modulus, mass buoyancy, and simulation run-time.

Table 5.1: Table with simulation setup parameters.

Bar charts 5.4 show the absolute error in meters per second of the flow velocity between the cross-correlation method (orange bars) and the echo energy method (blue bars). For all results a linear and zero offset correction is applied. The results in bar chart 5.4a.) are based on a excitation signal with properties listed in table 5.1. This signal does not have ideal autocorrelation properties. As mentioned in section 2.4.3, the CCM works best on signals with ideal autocorrelation properties. As such, the CCM shows a higher absolute

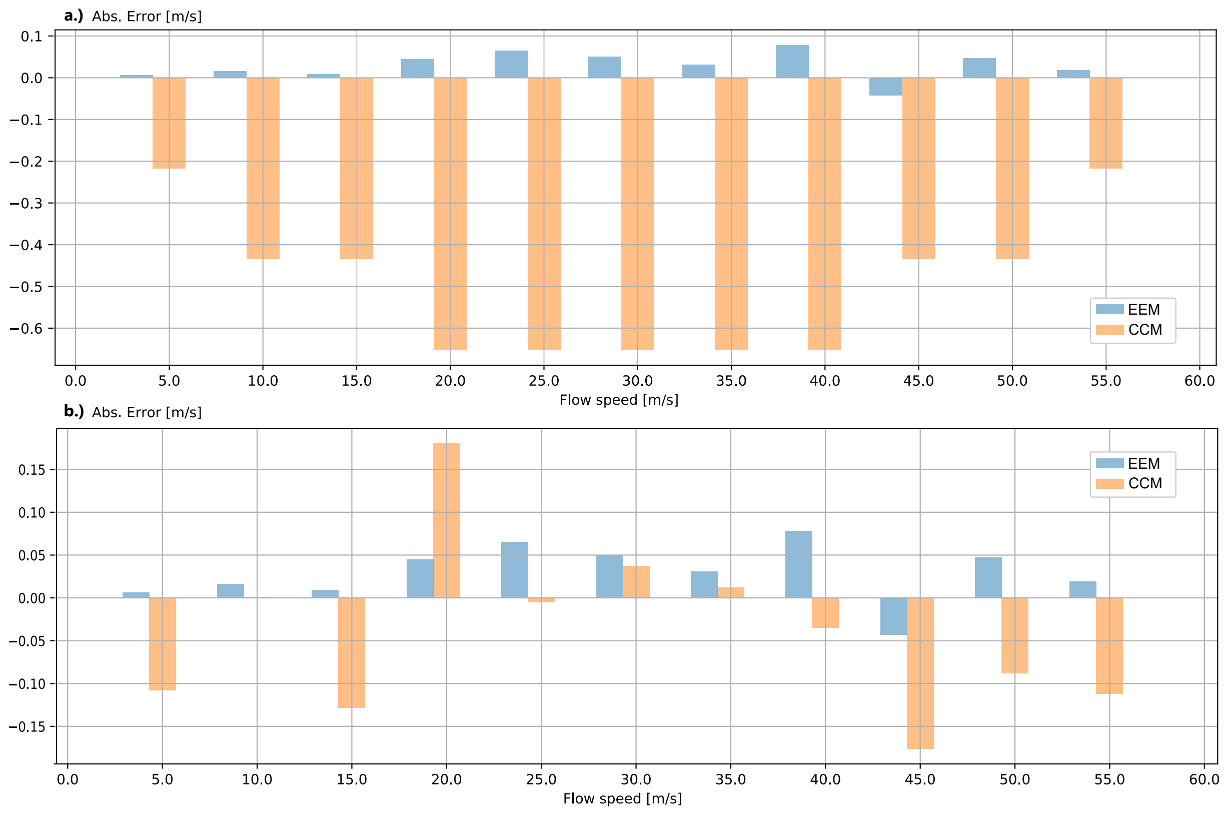


Figure 5.4: Bar chart a.) showing absolute error between the CCM and the EEM method with equal excitation signals. In bar chart b.) a Barker code is used for the CCM - tremendous decrease of the absolute error.

error compared to the EEM.

Bar chart 5.4b.) shows the absolute error of the CCM compared to the EEM. This time 'fair' conditions are in place - now, the CCM uses a Barker code of the length eleven (see table 2.1) having ideal autocorrelation properties. As expected, the CCM works now much more accurate compared to 5.4a.). Overall, the EEM works well for low flow velocities. Compared to the CCM, it shows a smaller absolute error over nearly the complete simulated range.

In a perfect simulation environment, one would generally expect no deviation in the calculated flow velocity from its nominal value. Especially at zero flow rate, the send out wave should reach both receivers (R_u, R_d) at the same time with equal pressure amplitudes (recall simulation environment in figure 3.2 - receivers are are spaced equidistant to the sender). Nevertheless, the simulation results show a different behaviour. Even at zero flow rate, a slight deviation in the received pressure amplitude is observed. The plots in figure 5.5 serve as an explanation for the numerical inaccuracy. Figure 5.5a.) shows the received normalized amplitude for the upstream (blue signal) and the downstream (orange signal). Timewise there is a perfect overlap. As such, only the downstream is visible.

Taking a closer look at one of the amplitude peaks shown in figure 5.5b.) makes a slight deviation visible. This is a result of the truncation error (see FDM truncation error in section 2.5.3) and a general rounding error present on each computer system (rounding error on floating point representation). Figures 5.5c.) and d.) shows the impact of the amplitude deviation on the normalization step. In figure 5.5c.) the unnormalized amplitudes of the two streams are subtracted. The deviation between the two streams is in the range of 10^{-5}. In contrast, looking at the subtraction of the normalized amplitudes in 5.5d.) it becomes obvious that the normalization step, due to the difference in the amplitude peaks, is moving the error range in the region of 10^{-2}. This has a tremendous impact on the EES and its area for the feature point determination. To minimize the impact on the flow velocity results, a zero offset correction is applied for all simulation runs.

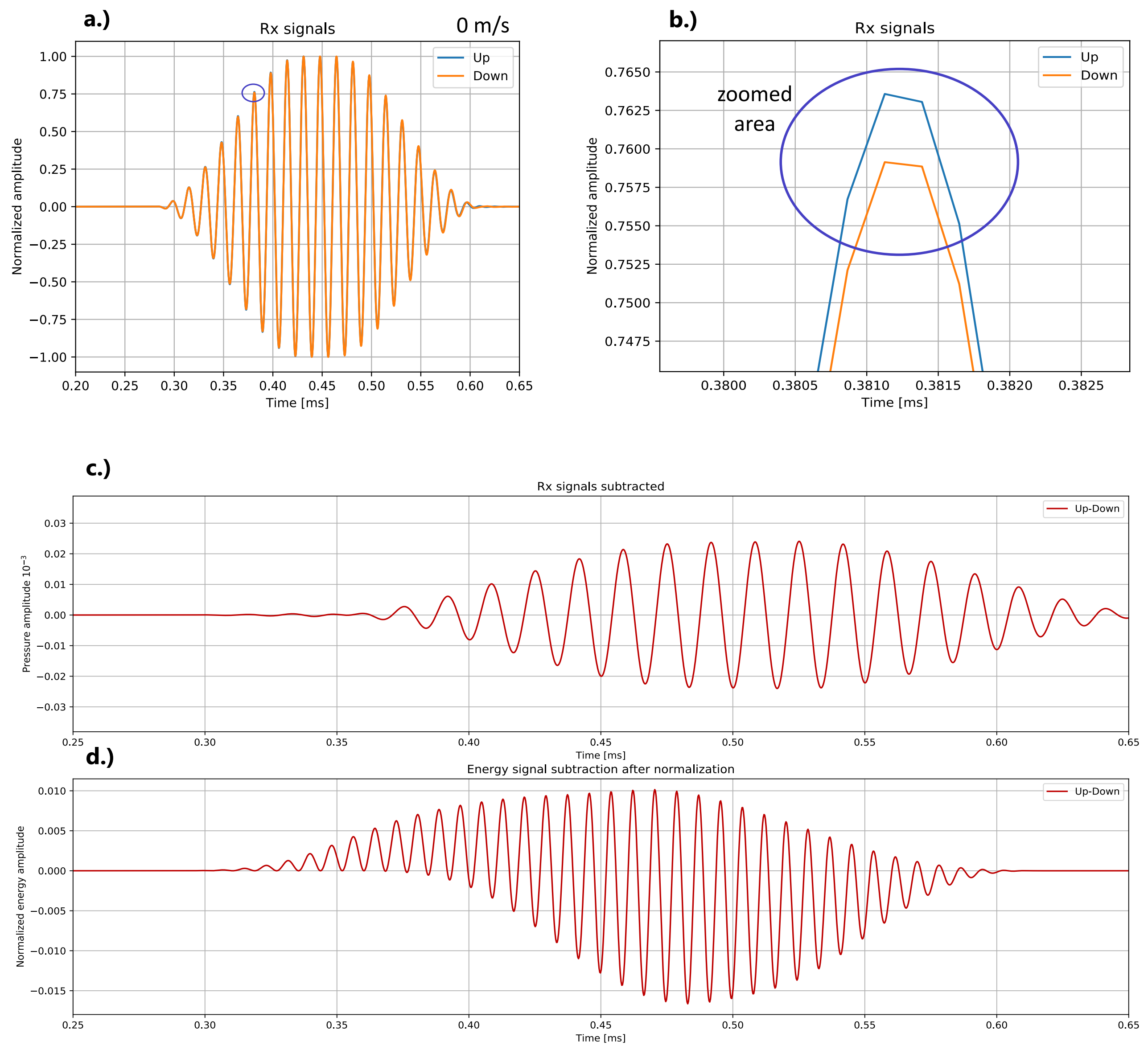


Figure 5.5: a.) Normalized received signals for up- and downstream. b.) Zoomed area showing amplitude deviation. c.) Subtracted up- and downstream. d.) Subtracted up- and downstream after normalization.

5.2 Zynq Implementation Results

This section provides the results of the hardware and software implementations described in chapter 4. For the evaluation of the hardware an unit of the carrier board was constructed. The carrier board was equipped by a manual pick and place procedure and soldered in a reflow-oven. This was time-consuming due to the massive amount of SMD components ($> 1k$). Figure 5.6 shows the carrier board with an attached PicoZed.

5.2.1 Zynq Design Results

In this subsection the timing analysis and overall FPGA utilization is given. The design timing summary is shown in table 5.2. All slack timings are positive indicating that the timing constraints are met. The requirement is 10 ns for a 100 MHz clock. The slack is 2.731 ns and positive. This means that we can speed up the design by $10-2.731 = 7.269$ ns, which is about 137 MHz.

The utilization of the full zynq block design (see figure 4.14) is shown in table 5.3 and visualized in diagram 5.7. The utilization is divided into the following resources: lookup tables (LUTs), RAM necessary for LUTs (LUTRAM), flip-flops (FF), block-RAM (BRAM), input-output connections (IO) and mixed-mode clock manager (MMCM) responsible for clock generation. Obviously, there is still quite some space within the PL for future extensions. Except for the IO resource, the utilization for all others is not breaching one-third of the maximum available resources.

Summary		
Setup	**Hold**	**Pulse Width**
Worst Neg. Slack: 2.731 ns	Worst Hold Slack: 0.080 ns	Worst Pulse Width Slack: 4.020 ns
Total Neg. Slack: 0.000 ns	Totals Hold Slack: 0.000 ns	Total Pulse Width Neg. Slack: 0.000 ns
# Failing Endpoints: 0	# Failing Endpoints: 0	# Failing Endpoints: 0
# Total Endpoints: 2223	# Total Endpoints: 2223	# Total Endpoints: 1002
All user specific timing constraints met.		

Table 5.2: Design timing summary - all constraints are met.

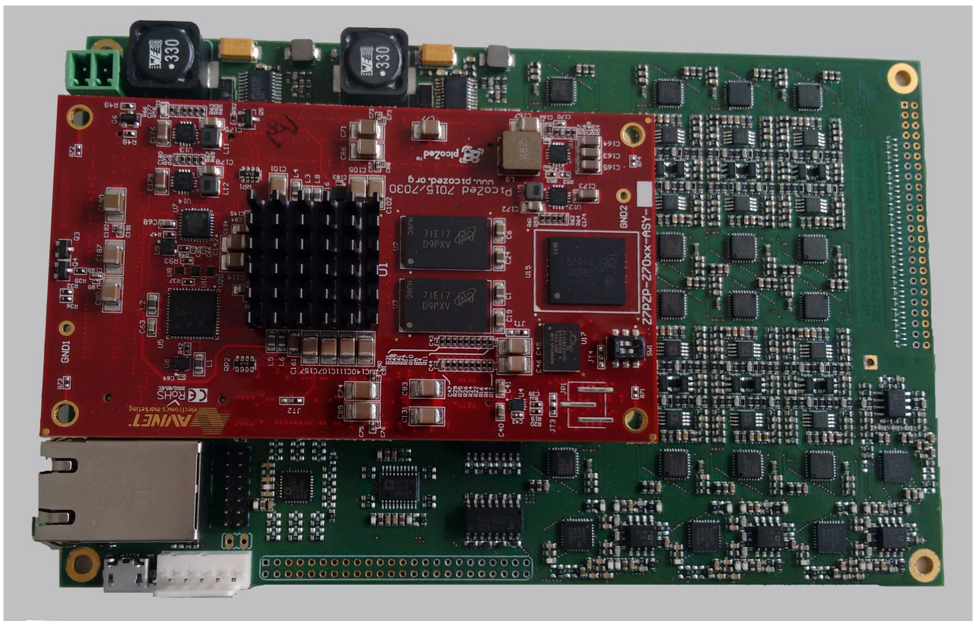


Figure 5.6: Carrier board (green PCB) with attached PicoZed (red PCB).

Resource	Utilization	Available	Utilization %
LUT	7132	46200	15.44
LUTRAM	849	14400	5.90
FF	12100	92400	13.10
BRAM	26.5	95	27.89
IO	118	150	78.67
MMCM	1	3	33.33

Table 5.3: Utilization summary.

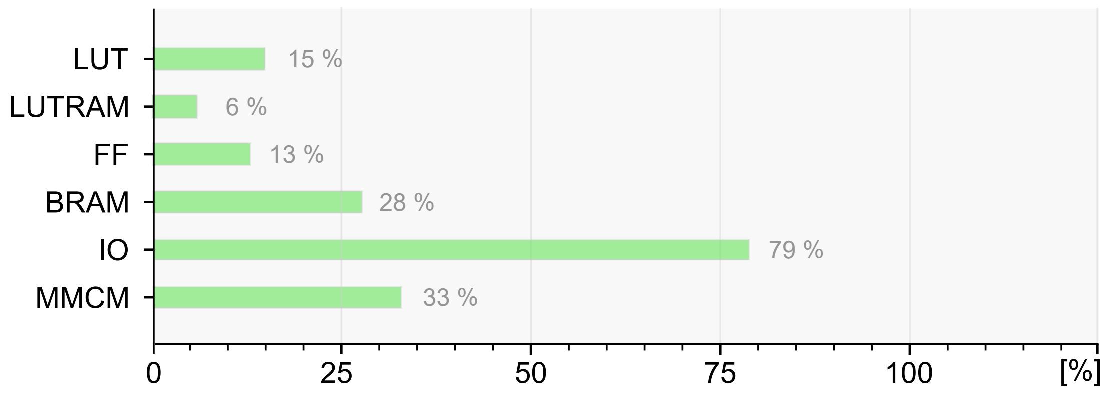
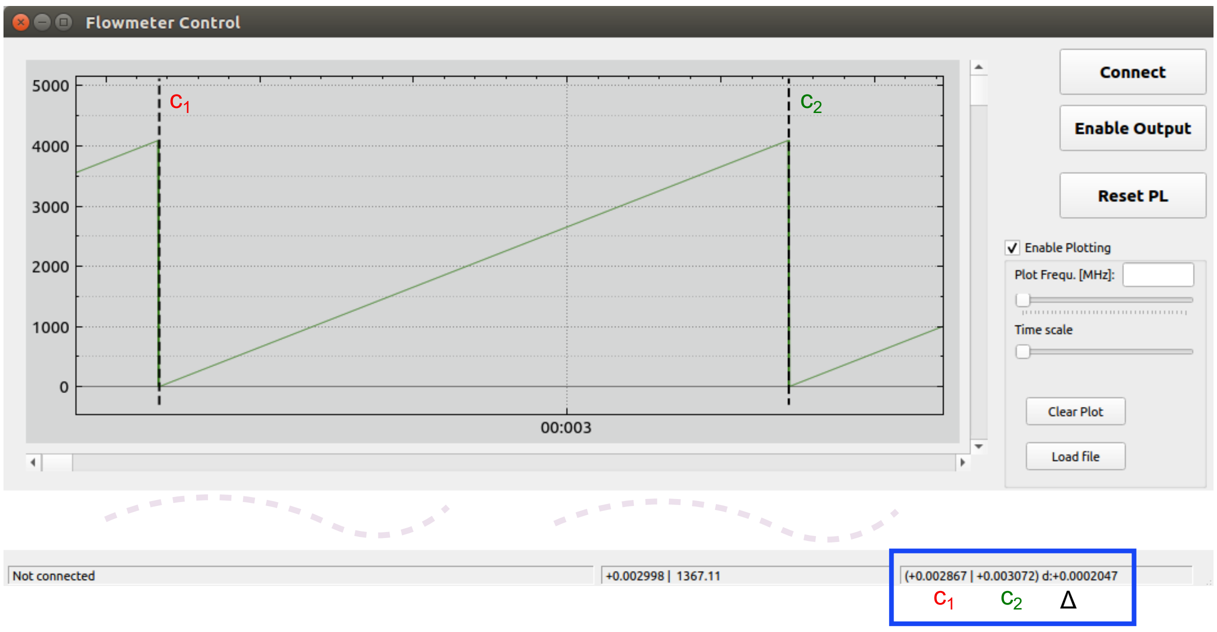


Figure 5.7: Utilization of the PL in percent.



Not connected

+0.002998 | 1367.11

(+0.002867 | +0.003072) d:+0.0002047

Figure 5.8: Measurement capabilities with mouse-placed cursors in the scope area of the GUI. The signal shows the sawtooth pattern of the ADCS in counter mode. The status bar displays timing values of the placed cursors.

5.2.2 Performance and Accuracy

Figure 5.8 shows a capture of the signal generated by the ADCS module in counter mode. The counter has 12-bits which corresponds to a decimal range from 0 to 4095 running at 20 MHz. This is noticeable as a repeating sawtooth signal. At the endpoints (counter overflows) two measurement cursors c_1 and c_2 are placed. The status bar displays the timing values of the placed cursors and the calculated delta timing.
A free running counter with 12-bits and a frequency of 20 MHz overflows after 204.8 µs. The measurement with the cursors in the status bar of figure 5.8 displays a delta of 204.7 µs. The very small divergence is a result of the placement accuracy of the cursors - even at the highest zoom-level in the scope area, a single pixel-offset will cause a small divergence in the measured versus expected timing value. However, looking at the logged sampling data the exact expected timing value can be found. Overall, this confirms the timing accuracy of the system at maximum frequency.

The network performance is a critical path in the design. To test the overall network performance, a UPD speed test routine is implemented. The results of multiple speed tests are shown in figure 5.9. The test is executed with a direct 1 Gbps LAN connection

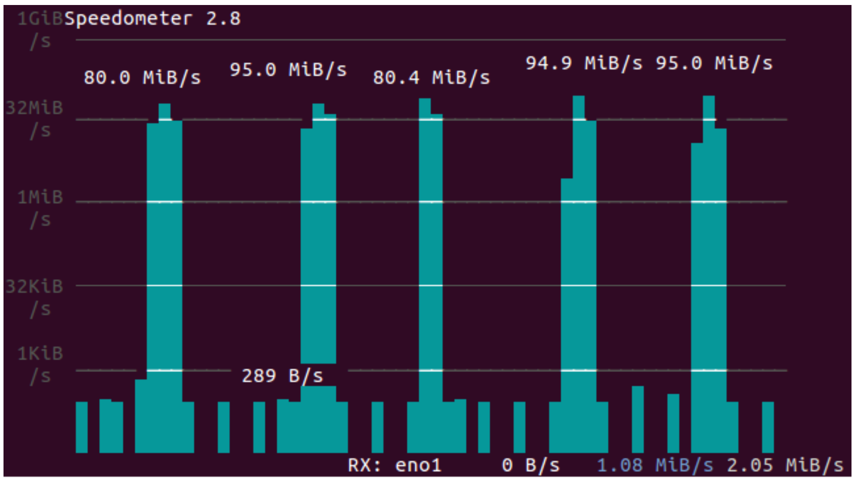


Figure 5.9: Average speeds on transfers of 128 MBps data chunks.

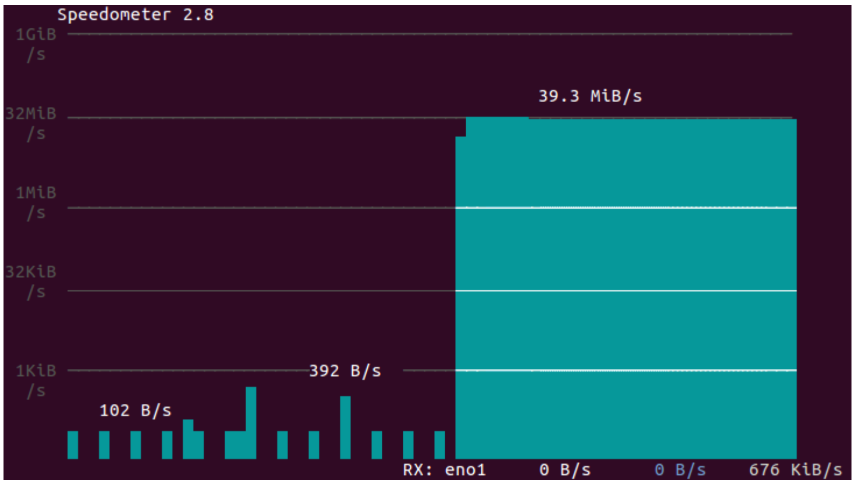


Figure 5.10: Bandwith requirement in ADCS continuous mode.

transferring 128 MB chunks of data from the reserved memory area. The average speed achieved is 745 Mbps, which is close to the theoretical maximum of 1 Gbps. Connected to the university internal network with a huge number of peers, the average speed decreases to ∼600 Mbps. This is still far enough bandwidth to continuously stream sampled data from the UAPP to the HOAPP.

In continuous mode at least 30 MBps are required just for the data itself (see section 4.3.2). Figure 5.10 shows the overall bandwidth requirement for the continuous mode including network overhead, which sums up to roughly 40 MBps. This is pretty much the maximum achievable transfer rate. The limiting factor is not the network bandwidth, but rather the software logic needed for polling on completed SG descriptors and triggering the network transfers on the Zynq. However, the design fulfills the targeted maximum sampling rate of the ADC and runs flawlessly in continuous mode.

CHAPTER 6

Conclusion

This master thesis aimed to address two parts in the area of ultrasonics flow measurement: (1) the evaluation of a novel signal processing method for flow velocity calculation and (2) a feature implementation for an alternative flow measurement principle. For the first part, the evaluation, a simulation environment was developed from scratch in python. The simulation for wave propagation is open source based and, as such, does not require any licensing. The achieved accuracy and results are more than sufficient for trivial scenarios. Based on the results of the simulation in section 5.1.3, users should consider the novel echo energy method as a a good alternative to the common established ultrasonic gas flow signal processing methods.

The second part of the master thesis is dealing with the sampling of the transducer signals. The implementation of the receiving path, makes the existing ultrasonic flow measurement system more feature complete. The extended capabilities of the system can now be used for ultrasonic signal generation and detection. There is no need for additional hardware for the sampling process anymore.
The developed receiving path allows high bandwidth data to be transfered from the digital logic to the Ethernet uplink. The usage of standardized AXI interfaces for all implemented modules enables modularity and future proof enhancement capabilities.

Bibliography

[1] R. Klambauer and A. Bergmann, "A new principle for an ultrasonic flow sensor for harsh environment," in *2017 IEEE SENSORS*, Oct 2017, pp. 1–3.

[2] A. N. S. Institute., *American national standard: acoustical terminology; ANSI S1.1-1994 (ASA 111-1994). Revision of ANSI S1.1-1960 (R 1976).*, 1994.

[3] H. M. Merklinger and D. D. Ellis, "Fessenden and boyle: Two canadian sonar pioneers," *Proceedings of Meetings on Acoustics*, vol. 30, no. 1, p. 070001, 2017. [Online]. Available: https://asa.scitation.org/doi/abs/10.1121/2.0000564

[4] "Paul langevin," *Ultrasonics*, vol. 10, no. 5, pp. 213 – 214, 1972. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0041624X72902946

[5] L. Bjørnø, "Developments in sonar technologies and their applications," in *2013 IEEE International Underwater Technology Symposium (UT)*, March 2013, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/6519839

[6] Y. Yang and S. Lin, "Medical ultrasonic treatment and its applications in medicine," in *2012 Symposium on Piezoelectricity, Acoustic Waves, and Device Applications (SPAWDA)*, Nov 2012, pp. 447–450. [Online]. Available: https://ieeexplore.ieee.org/document/6464129

[7] M. Möser, *Technische Akustik*, ser. VDI-Buch. Springer Berlin Heidelberg, 2015. [Online]. Available: https://books.google.at/books?id=RIfDCgAAQBAJ

[8] Q. Zhou, K. H. Lam, H. Zheng, W. Qiu, and K. K. Shung, "Piezoelectric single crystal ultrasonic transducers for biomedical applications," *Progress in Materials Science*, vol. 66, pp. 87 – 111, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0079642514000541

[9] R. Krimholtz, D. A. Leedom, and G. L. Matthaei, “New equivalent circuits for elementary piezoelectric transducers,” *Electronics Letters*, vol. 6, no. 13, pp. 398–399, June 1970.

[10] D. Ensminger and L. Bond, *Ultrasonics: Fundamentals, Technology, Applications, Third Edition*, ser. Dekker Mechanical Engineering. Taylor & Francis, 2012. [Online]. Available: https://books.google.at/books?id=1MTm8qiY1fQC

[11] H. Shinoda, T. Nakajima, K. Ueno, and N. Koshida, “Thermally induced ultrasonic emission from porous silicon,” *Nature*, vol. 400, pp. 853–855, 08 1999.

[12] O. Oralkan, A. Y. Ergun, J. Johnson, M. C. Karaman, U. Demirci, K. Kaviani, T. Lee, and B. T. Khuri-Yakub, “Capacitive micromachined ultrasonic transducers: next-generation arrays for acoustic imaging?” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 49, pp. 1596–1610, 2002.

[13] B.-Y. Hsieh, J. Kim, J. Zhu, S. Li, X. Zhang, and X. Jiang, “A laser ultrasound transducer using carbon nanofibers–polydimethylsiloxane composite thin film,” *Applied Physics Letters*, vol. 106, no. 2, p. 021902, 2015. [Online]. Available: https://doi.org/10.1063/1.4905659

[14] R. Tech, *Introduction to Phased Array Ultrasonic Technology Applications: R/D Tech Guideline*, ser. Advanced practical NDT series. R/D Tech, 2004. [Online]. Available: https://books.google.at/books?id=0f1BPwAACAAJ

[15] R. C. Baker, *Flow Measurement Handbook: Industrial Designs, Operating Principles, Performance, and Applications.* Cambridge University Press, 2000.

[16] C. H. M. H. W. H. M. K. G. L. W. M. U. M. M. O. F. O. K.-H. R. D. S. A. T. H.-J. W. F. B. C. K. L. D. E. H. A. S. U. P. B. J. H. L. F. L. G. R. S. P. A. K. T. M. F. Frenzel, H. Grothey, *Industrial flow measurement - Basics and practice.* ABB Automation Products GmbH, 2011.

[17] I. Y. Han, D.-K. Kim, and S. J. Kim, “Study on the transient characteristics of the sensor tube of a thermal mass flow meter,” *International Journal of Heat and Mass Transfer*, vol. 48, no. 13, pp. 2583 – 2592, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0017931005001286

[18] P. Bruschi, M. Dei, and M. Piotto, “A single chip, double channel thermal flow meter,” *Microsystem Technologies*, vol. 15, no. 8, pp. 1179–1186, Aug 2009. [Online]. Available: https://doi.org/10.1007/s00542-008-0741-x

[19] M. M. E. C. Flow and D. Meters, “Product data sheet, ps-00374, rev ai,” Tech. Rep., January 2019. [Online]. Available: https://www.emerson.com/documents/automation/product-data-sheet-elite-series-coriolis-flow-density-meters-micro-motion-en-66748.pdf

[20] W. P. Mason, *Physical Acoustics V14: Principles and Methods 1st Edition.* ACADEMIC PRESS, INC. (LONDON) LTD., 1979.

[21] K. J., “An ultrasonic flowmeter for liquids,” *Progress in Materials Science*, vol. 55-16-3, pp. 1–6, 1955a.

[22] Z. Kaneko, “First steps in the development of the doppler flowmeter,” *Ultrasound in Medicine & Biology*, vol. 12, no. 3, pp. 187 – 195, 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0301562986903091

[23] I. research company. (2019) Ultrasonic flowmeter market: Global industry trends, share, size, growth, opportunity and forecast 2019-2024. [Online]. Available: https://www.imarcgroup.com/ultrasonic-flowmeter-market

[24] Q. Chen, W. Li, and J. Wu, “Realization of a multipath ultrasonic gas flowmeter based on transit-time technique,” *Ultrasonics*, vol. 54, 06 2013.

[25] G. Rajita and N. Mandal, “Review on transit time ultrasonic flowmeter,” pp. 88–92, Jan 2016.

[26] D. V. Mahadeva, R. C. Baker, and J. Woodhouse, “Further studies of the accuracy of clamp-on transit-time ultrasonic flowmeters for liquids,” *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 5, pp. 1602–1609, May 2009.

[27] ——, “Studies of the accuracy of clamp-on transit time ultrasonic flowmeters,” in *2008 IEEE Instrumentation and Measurement Technology Conference*, May 2008, pp. 969–973.

[28] L. Xingtai Kunlun Machinery Co. Specialized ultrasonic flow measurement instrumentation manufacturer. [Online]. Available: http://www.xtflowmeter.com

[29] E. Mandard, D. Kouame, R. Battault, J. Remenieras, and F. Patat, "Methodology for developing a high-precision ultrasound flow meter and fluid velocity profile reconstruction," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 55, no. 1, pp. 161–172, January 2008. [Online]. Available: https://ieeexplore.ieee.org/document/4454311

[30] B. Barshan, "Fast processing techniques for accurate ultrasonic range measurements," *Measurement Science and Technology*, vol. 11, p. 45, 12 1999.

[31] J. C. Jackson, R. Summan, G. I. Dobie, S. M. Whiteley, S. G. Pierce, and G. Hayward, "Time-of-flight measurement techniques for airborne ultrasonic ranging," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 60, no. 2, pp. 343–355, February 2013. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6416489

[32] Z. Fang, L. Hu, K. Mao, W. Chen, and X. Fu, "Similarity judgment-based double-threshold method for time-of-flight determination in an ultrasonic gas flowmeter," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, no. 1, pp. 24–32, Jan 2018.

[33] B. Barshan and B. Ayrulu, "Performance comparison of four time-of-flight estimation methods for sonar signals," *Electronics Letters*, vol. 34, no. 16, pp. 1616–1617, Aug 1998. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/707177

[34] M. Mirghani and A. Awad Babiker, *Adaptive Coding, Modulation and Filtering of Radar Signals*, 05 2018.

[35] B. Liu, K.-J. Xu, L.-B. Mu, and L. Tian, "Echo energy integral based signal processing method for ultrasonic gas flow meter," *Sensors and Actuators A: Physical*, vol. 277, pp. 181 – 189, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924424718302760

[36] F. Alauzet, X. Li, E. S. Seol, and M. S. Shephard, "Parallel anisotropic 3d mesh adaptation by mesh modification," *Engineering with Computers*, vol. 21, no. 3, pp. 247–258, May 2006. [Online]. Available: https://doi.org/10.1007/s00366-005-0009-3

[37] P. Causon and P. Mingham, *Introductory Finite Difference Methods for PDEs*. Bookboon. [Online]. Available: https://books.google.at/books?id=9wrZ4XL6gGMC

[38] B. Engquist and A. Majda, "Absorbing boundary conditions for numerical simulation of waves," *Proceedings of the National Academy of Sciences*, vol. 74, no. 5, pp. 1765–1766, 1977. [Online]. Available: https://www.pnas.org/content/74/5/1765

[39] R. Clayton and B. Engquist, "Absorbing boundary conditions for acoustic and elastic equations," *Bulletin of the Seismological Society of America*, vol. 67, pp. 1529–1540, 12 1977.

[40] C. Cerjan, D. Kosloff, R. Kosloff, and M. Reshef, "A nonreflecting boundary-condition for discrete acoustic and elastic wave-equations," *Geophysics*, vol. 50, pp. 705–708, 04 1985.

[41] Z. Jiang, J. C Bancroft, and L. R Lines, "Combining absorbing and nonreflecting boundary conditions for elastic wave simulation," 01 2010.

[42] J.-P. Berenger, "A perfectly matched layer for the absorption of electromagnetic waves," *Journal of Computational Physics*, vol. 114, no. 2, pp. 185 – 200, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999184711594

[43] F. Nataf, "Absorbing boundary conditions and perfectly matched layers in wave propagation problems," in *Direct and Inverse problems in Wave Propagation and Applications*, ser. Radon Ser. Comput. Appl. Math. de Gruyter, 2013, vol. 14, pp. 219–231. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00799759

[44] S. G. Johnson, "Notes on perfectly matched layers (pmls)," 2010. [Online]. Available: https://www.semanticscholar.org/paper/Notes-on-Perfectly-Matched-Layers-(PMLs)-Johnson/675fe8176ebbdb23f5d464b12e0c68970d60e028

[45] R. Bernatz, *Fourier Series and Numerical Methods for Partial Differential Equations*, 08 2010.

[46] G. Strang, "18.086 mathematical methods for engineers ii, massachusetts institute of technology: Mit opencourseware," 2006. [Online]. Available: https://ocw.mit.edu.

[47] S. Heath, Ed., *Index*, second edition ed. Oxford: Newnes, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780750655460500165

[48] T. B. Group. Embedded systems glossary. [Online]. Available: https://barrgroup.com/Embedded-Systems/Glossary-E

[49] E. Garnsey, G. Lorenzoni, and S. Ferriani, "Speciation through entrepreneurial spin-off: The acorn-arm story," *Research Policy*, vol. 37, no. 2, pp. 210 – 224, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0048733307002363

[50] H. Yang, J. Zhang, J. Sun, and L. Yu, "Review of advanced fpga architectures and technologies," *Journal of Electronics (China)*, vol. 31, no. 5, pp. 371–393, Oct 2014. [Online]. Available: https://doi.org/10.1007/s11767-014-4090-x

[51] E. Monmasson and M. N. Cirstea, "Fpga design methodology for industrial control systems—a review," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 4, pp. 1824–1842, Aug 2007.

[52] P. B. Minev and V. S. Kukenska, "Implementation of soft-core processors in fpgas," in *UNITECH'07 International Sceintific conference*, 2007.

[53] J. J. Rodríguez-Andina, M. D. Valdés-Peña, and M. J. Moure, "Advanced features and industrial applications of fpgas—a review," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 4, pp. 853–864, Aug 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7104117

[54] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart, *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc*. UK: Strathclyde Academic Media, 2014.

[55] A. Shrivastav, G. S. Tomar, and A. K. Singh, "Performance comparison of amba bus-based system-on-chip communication protocol," in *2011 International Conference on Communication Systems and Network Technologies*, June 2011, pp. 449–454. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5966487

[56] XILINX, "Axi reference guide ug761 (v13.1)," Tech. Rep. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf

[57] D. J. Sorin, M. D. Hill, and D. A. Wood, "A primer on memory consistency and cache coherence," *Synthesis Lectures on Computer Architecture*, vol. 6, no. 3, pp. 1–212, 2011. [Online]. Available: https://doi.org/10.2200/S00346ED1V01Y201104CAC016

[58] W. Kester, W. Kester@analog, and C. , "Which adc architecture is right for your application," 07 2005. [Online]. Available: https://www.researchgate.net/publication/228761926_Which_ADC_architecture_is_right_for_your_application

[59] R. Staffin and R. D. Lohman, "Signal amplitude quantizer," Patent U.S. Patent 2,869,079, filed December 19, 1956, issued January 13, 1959.

[60] Y. Shmaliy, *Continuous-Time Signals*, 09 2006.

[61] A. J. Jerri, "The shannon sampling theorem—its various extensions and applications: A tutorial review," *Proceedings of the IEEE*, vol. 65, no. 11, pp. 1565–1596, Nov 1977.

[62] L. Cordesses, "Direct digital synthesis: a tool for periodic wave generation (part 1)," *IEEE Signal Processing Magazine*, vol. 21, no. 4, pp. 50–54, July 2004. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1311140

[63] T. Petazzoni, "Device tree for dummies," Tech. Rep., 2015. [Online]. Available: https://ohwr.org/project/embedded-linux/wikis/Documents/Device-Tree-for-Dummies

[64] J. Corbet, A. Rubini, and G. Kroah-Hartman, *Linux Device Drivers, 3rd Edition.* O'Reilly Media, Inc., 2005.

[65] D. community. Devicetree specification, release v0.2. [Online]. Available: www.devicetree.org

[66] L. L. B. Wilson D. Keith., "Finite-difference, time-domain simulation of sound propagation in a dynamic atmosphere," United States. Army. Corps of Engineers. Engineer Research and Development Center (U.S.) Cold Regions Research and Engineering Laboratory (U.S.), Tech. Rep., 2004. [Online]. Available: http://acwc.sdp.sirsi.net/client/en_US/default/search/detailnonmodal/ent:$002f$002fSD_ILS$002f0$002fSD_ILS:154352/ada/?rt=CKEY%7C%7C%7CCKEY%7C%7C%7Cfalse

[67] S. Abdullaev and V. Ostashev, "On propagation of sound waves in three-dimensional inhomogeneous moving (randomly inhomogeneous) media," *Izvestiya Atmospheric and Oceanic Physics*, vol. 24, pp. 310–316, 04 1988.

[68] A. D. F., "Acoustic wave equations for a linear viscous fluid and an ideal fluid." [Online]. Available: https://digital.library.unt.edu/ark:/67531/metadc741848/

[69] D. Kleine, de, B. Esch, van, J. Kuerten, and A. Vreman, “Calculation of unsteady flow in a centrifugal pump with vaned diffuser using staggered and collocated grid methods,” in *Proceedings of the ASME 2009 Fluids Engineering Division Summer Meeting, (FEDSM2009),August 2-6,2009, Vail,Colorado*, 2009, pp. 78 113–1/8.

[70] P. Wesseling, *Principles of Computational Fluid Dynamics.* Berlin, Heidelberg: Springer-Verlag, 2000.

[71] S. V. Patankar, *Numerical heat transfer and fluid flow*, ser. Series on Computational Methods in Mechanics and Thermal Science. Hemisphere Publishing Corporation (CRC Press, Taylor & Francis Group), 1980. [Online]. Available: http://www.crcpress.com/product/isbn/9780891165224

[72] V. Ostashev, D. Wilson, L. Liu, D. F Aldridge, N. Symons, and D. Marlin, “Equations for finite-difference, time-domain simulation of sound propagation in moving inhomogeneous media and numerical implementation,” *The Journal of the Acoustical Society of America*, vol. 117, pp. 503–17, 03 2005.

[73] L. K. documentation. [Online]. Available: https://www.kernel.org/doc/Documentation/devicetree/bindings/reserved-memory/reserved-memory.txt

[74] S. Verma and A. S. Dabare, “Understanding clock domain crossing issues,” *EE Times*, 2007.

[75] XILINX, “Axi dma v7.1 - logicore ip product guide, pg021,” Tech. Rep. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf

[76] ——, “Zynq-7000 soc - technical reference manual, ug585 (v1.12.2,” Tech. Rep. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf