Signal Processing Architectures for Automotive High-Resolution MIMO Radar Systems

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades

Doktor-Ingenieur
(abgekürzt: Dr.-Ing.)

genehmigte Dissertation

von Herrn

Dipl.-Ing. Frank Meinl

geboren 1988 in München

2020

1. Referent:	Prof. Dr.-Ing. Holger Blume
2. Referentin:	Prof. Dr.-Ing. Ilona Rolfes

Tag der Promotion: 19. Februar 2020

Abstract

To date, the digital signal processing for an automotive radar sensor has been handled in an efficient way by general purpose signal processors and microcontrollers. However, increasing resolution requirements for automated driving on the one hand, as well as rapidly growing numbers of manufactured sensors on the other hand, can provoke a paradigm change in the near future. The design and development of highly specialized hardware accelerators could become a viable option – at least for the most demanding processing steps with data rates of several gigabits per second.

In this work, application-specific signal processing architectures for future high-resolution multiple-input and multiple-output (MIMO) radar sensors are designed, implemented, investigated and optimized. A focus is set on real-time performance such that even sophisticated algorithms can be computed sufficiently fast. The full processing chain from the received baseband signals to a list of detections is considered, comprising three major steps: Spectrum analysis, target detection and direction of arrival estimation.

The developed architectures are further implemented on a field-programmable gate array (FPGA) and important measurements like resource consumption, power dissipation or data throughput are evaluated and compared with other examples from literature. A substantial dataset, based on more than 3600 different parametrizations and variants, has been established with the help of a model-based design space exploration and is provided as part of this work. Finally, an experimental radar sensor has been built and is used under real-world conditions to verify the effectiveness of the proposed signal processing architectures.

Keywords — Automotive radar, MIMO, FMCW, Chirp sequence, FFT, CFAR, Direction of arrival, DOA, FPGA, Design space exploration, Signal processing, Real-time, Maximum likelihood estimation, Fixed-point.

Kurzfassung

Bisher wurde die digitale Signalverarbeitung für automobile Radarsensoren auf eine effiziente Art und Weise von universell verwendbaren Mikroprozessoren bewältigt. Jedoch können steigende Anforderungen an das Auflösungsvermögen für hochautomatisiertes Fahren einerseits, sowie schnell wachsende Stückzahlen produzierter Sensoren andererseits, einen Paradigmenwechsel in naher Zukunft bewirken. Die Entwicklung von hochgradig spezialisierten Hardwarebeschleunigern könnte sich als eine praktikable Alternative etablieren – zumindest für die anspruchsvollsten Rechenschritte mit Datenraten von mehreren Gigabits pro Sekunde.

In dieser Arbeit werden anwendungsspezifische Signalverarbeitungsarchitekturen für zukünftige, hochauflösende, MIMO Radarsensoren entworfen, realisiert, untersucht und optimiert. Der Fokus liegt dabei stets auf der Echtzeitfähigkeit, sodass selbst anspruchsvolle Algorithmen in einer ausreichend kurzen Zeit berechnet werden können. Die komplette Signalverarbeitungskette, beginnend von den empfangenen Signalen im Basisband bis hin zu einer Liste von Detektion, wird in dieser Arbeit behandelt. Die Kette gliedert sich im Wesentlichen in drei größere Teilschritte: Spektralanalyse, Zieldetektion und Winkelschätzung.

Des Weiteren werden die entwickelten Architekturen auf einem FPGA implementiert und wichtige Kennzahlen wie Ressourcenverbrauch, Stromverbrauch oder Datendurchsatz ausgewertet und mit anderen Beispielen aus der Literatur verglichen. Ein umfangreicher Datensatz, welcher mehr als 3600 verschiedene Parametrisierungen und Varianten beinhaltet, wurde mit Hilfe einer modellbasierten Entwurfsraumexploration erstellt und ist in dieser Arbeit enthalten. Schließlich wurde ein experimenteller Radarsensor aufgebaut und dazu benutzt, die entworfenen Signalverarbeitungsarchitekturen unter realen Umgebungsbedingungen zu verifizieren.

Schlagworte — Automobilradar, Digitale Signalverarbeitung, Echtzeitsystem, Entwurfsraumexploration, Winkelschätzung, Radarsensorik, Konstante Falschalarmrate, Maximum-Likelihood-Schätzung.

Acknowledgments

I am very thankful for the support I got in the course of my work as a doctoral candidate and I want to express my gratitude to all persons and colleagues who helped me during this period.

First and foremost, I want to thank my supervisor Prof. Dr. Holger Blume for his outstanding technical and academic guidance, his open-minded nature and most importantly for his steady encouragement and support. Only with his excellent cooperation, the successful completion of this work became possible. I am also very grateful to Prof. Dr. Ilona Rolfes for her time and effort spent in her role as co-examiner, as well as for her interest in my work. In the same way I appreciate the commitment of Prof. Dr. Bernardo Wagner, acting as a chairman for the board of examiners.

Likewise, I express my deepest gratitude to all employees of the Robert Bosch GmbH who accompanied me and my work over the past years. You all helped me considerably by sharing your knowledge and expertise, by establishing an excellent working environment, and not least by allowing me enough freedom. At this point, I especially want to thank Dr. Martin Kunert, Dr. Eugen Schubert and Dr. Berthold Käferstein for their long-standing collaboration, their permanent availability for discussions, as well as their trust and confidence in my work. Furthermore I am extremely thankful to Dr. Martin Stolz and Dr. Tilman Glökler for their accurate feedback and valuable editorial remarks.

Then, a very warm thank-you goes to all employees of the Institute of Microelectronic Systems in Hannover for their great cooperation, as well as their openness and readiness to help. I felt very welcome in my role as a visiting researcher and the numerous contacts and exchanges were highly valuable. In particular, I want to thank Dr. Nico Mentzer who supported me from the first day and who was available for virtually any imaginable question I had.

I further want to thank Prof. Dr. Andreas Stelzer and Dr. Reinhard Feger from the Institute for Communications Engineering and RF-Systems in Linz for their inspiring and valuable collaboration in the field of RF electronics and radar technology.

Finally, I am just as well very thankful to all other persons, to my family and my friends, who helped and supported me during this challenging time.

Frank Meinl

Contents

1 Introduction **1**

1.1 Contribution of this work 3

2 Automotive Radar Sensors **5**

2.1 Radar waveforms 6

2.1.1 Pulsed and continuous-wave (CW) radar 7

2.1.2 Frequency-modulated continuous wave (FMCW) 8

2.1.3 Chirp sequence modulation 11

2.1.4 Spectrum evaluation 14

2.2 Target detection 17

2.2.1 Constant false alarm rate (CFAR) 18

2.2.2 Non-coherent integration 22

2.3 Direction of arrival estimation 25

2.3.1 Antenna arrays 27

2.3.2 Classical beamforming 32

2.3.3 Maximum likelihood (ML) estimator 35

2.3.4 Multiple-input multiple-output (MIMO) 40

2.4 State of the art 44

2.4.1 Related work 47

3 Architectures for Real-time Signal Processing **51**
3.1 Two-dimensional fast Fourier transform (FFT) 52
3.1.1 Pipelined FFT implementations 54
3.1.2 Fixed-point arithmetic 58
3.1.3 High-throughput matrix transformation with SDRAM . 73
3.1.4 Window function 77
3.2 Target detection 79
3.2.1 Rank-only OS-CFAR 79
3.2.2 Target detection accelerator 82
3.3 Angle estimation 85
3.3.1 Single target ML estimator 85
3.3.2 Two target ML estimator 92
3.4 Summary 98

4 Design Space Exploration **103**
4.1 Methodology 103
4.1.1 Measured quantities in the design space 104
4.1.2 Design space exploration procedure 108
4.1.3 Fitting of a model function 112
4.2 Target detection module 114
4.2.1 Area 115
4.2.2 Power 119
4.2.3 Performance (Data throughput) 120
4.3 Single target ML estimator 121
4.3.1 Area 122
4.3.2 Power 127
4.3.3 Performance (Data throughput) 127
4.3.4 Numerical accuracy 129
4.4 Two target ML estimator 130
4.4.1 Area 131
4.4.2 Power 133
4.4.3 Performance (Data throughput) 134
4.4.4 Area and power efficiency 135

5 Verification with an Experimental High-Resolution MIMO Radar 139
5.1 Overall radar system architecture 140
  5.1.1 Radar sensor head 142
  5.1.2 Signal processing back end 144
5.2 Acquisition of measurement data 145
5.3 Real world evaluation 148
  5.3.1 Single snapshot performance 148
  5.3.2 Requirements for next generation radar systems 150

6 Conclusion 153
6.1 Future work 155

Appendix 157
A.1 FMCW baseband signal derivation 157
A.2 FPGA silicon area estimation 159
A.3 Power consumption data 162
  A.3.1 Target detection module 162
  A.3.2 Single target ML estimator 164
  A.3.3 Two target ML estimator 166

List of abbreviations 169

List of own publications 173

Bibliography 175

Curriculum vitae 191

1

Introduction

With over 100 years of history, radar is a major sensing technology with a large spectrum of applications. At the beginning mainly developed for military and defense, more and more civilian applications emerged gradually. In recent years, radar sensors have been successfully employed in passenger cars where they provide necessary information about the current driving situation and the surrounding environment of the vehicle. Nowadays, the prevailing applications of automotive radar sensors are advanced driver assistance systems (ADAS) which require a real-time perception of the present traffic scenario.

Over the next years, a further extension of the intended use cases for radar sensors can be anticipated. Not only due to their robustness in several environmental conditions with poor visibility, for instance during low sunlight, night, rain, fog or smoke, automotive radar sensors are considered to be one pillar of a self-driving car's perception system. The excellent range resolution as well as an instantaneous measurement of the velocity are further advantages of radar sensors compared to other sensing technologies like video or lidar.

Future developments toward automated driving create an increasing demand for high-resolution sensors, aimed to gather even more accurate and detailed information about the environment. Though, the more fine-grained maps of the environment come at the cost of a higher processing load. The evolution of automotive radar sensors has to take place on several levels whether it is a faster sampling rate of the analog-to-digital converters, more antennas and parallel channels, an extended measurement time or simply a higher measurement rate. These changes have all in common that they increase the amount of data which has to be handled by the sensor in a certain period of time. In addition, the development and employment of more sophisticated algorithms and evaluation methods create a further burden for the processing unit. Such approaches comprise for instance parametric spectral estimation, occupancy grid mapping or a semantic segmentation based on neural networks [1,2].

So far, the real-time signal processing for state of the art automotive radar sensors could be handled sufficiently by digital signal processors (DSPs) [3, pp. 379 – 398]. These devices achieve a sufficient performance for current applications, like adaptive cruise control (ACC) or autonomous emergency braking (AEB), with their relatively low requirements in terms of data rates and computational load. For more advanced approaches, the employment of multiple DSP cores and additional vector processors is still considered as a best practice [4]. The prevalence of DSP-based systems is further supported by well-established programming tools and by a wide range of available software libraries which facilitate product development.

Two undeniable facts may provoke a paradigm change in the near future. Firstly, the data rates are reaching a critical level so that conventional microprocessor architectures cannot catch up in terms of silicon area and power consumption. Secondly, a tremendous increase in the number of sold sensors and equipped vehicles justify a higher development and integration effort. In other words, the fast increasing requirements on the processing unit are hard to fulfill with conventional microprocessor devices, while at the same time a per-piece cost advantage is getting more and more important due to the larger volumes. As a consequence, a trend toward more specialized architectures is expected in the near future.

The employment of dedicated hardware accelerators confronts the radar system developer with new questions, because a larger design space is suddenly at their disposal. Optimized processing pipelines can be designed, the degree of parallelization can be defined, data word sizes can be tailored to the application, different architectural options are available and have to be chosen already during the concept phase. At the same time, early cost estimates of the envisaged final product are crucial for a successful market entry, which can be a seemingly impossible task.

A practical approach is the early traverse and exploration of the design space prior to an actual product development [5]. The focus is given to the setup of an implementation database which contains important figures and characteristics of modules with a varying parametrization. Subsequently, appropriate model functions can be derived to cover any possible parameter combination. Such tools can be used later to provide quick and accurate cost estimates across a large and multidimensional design space. Finally, it shall be feasible to predict key figures like consumed silicon area or power dissipation before the real development phase has begun.

1.1 Contribution of this work

One essential contribution of this work is to investigate signal processing concepts for future automotive radar sensors which are suitable for a hardware acceleration. The trend toward fast chirp sequence modulations, large MIMO antenna arrays and higher data rates engenders the use of new evaluation algorithms. Different approaches exist and they need to be investigated thoroughly in order to optimize a possible hardware architecture. The outcome of a literature study, including the state of the art in industry and research is presented in chapter 2.

Furthermore, suitable implementation forms have to be identified and explored in order to set the basis for a future system-on-chip integration. For this purpose, the most promising approaches which have been identified in chapter 2 will be further investigated and implemented as hardware accelerator modules. They are described and modeled with the help of the language VHDL[1] which allows a direct employment in FPGA devices. The architecture of these modules and their interaction is shown in chapter 3.

With the means of a model-based design space exploration, a radar-specific database of certain algorithm implementations has been established. The methods, as well as the results are illustrated in chapter 4. The system designer can use such models to select the best trade-off fulfilling the system's requirements at a minimum cost. So-called Pareto optimal realizations can be found and identified from these results.

In addition, early prototype realizations of a high-performance radar system can be achieved with the help of the implemented building blocks. An operational system has been built up, based on off-the-shelf parts and devices like for instance commercial available FPGAs. As part of this work, the provided implementations form the basis for a radar development framework which

---

[1] Very High Speed Integrated Circuit Hardware Description Language

is operational under real-time conditions. Unlike other experimental setups which are only stationary and cannot support a long data acquisition period, the radar prototype presented in chapter 5 can be mounted on a test vehicle and is able to operate non-stop at a high cycle rate.

A real-world operation and recording of high-resolution radar images during challenging traffic scenarios is possible. The aim is to support the research in fields like algorithm development, machine learning, system verification and requirement analysis, just to mention a few. So far, several scientific publications have been realized with the help of data acquired by this prototype setup [6–23]. Further research is still going on so that even more results can be expected in the near future.

2

Automotive Radar Sensors

The basic working principle of radar sensors is rather straightforward and similar for all kind of applications. Though, a variety of different realizations and technologies exist – each with its own focus – which makes it difficult to get a quick overview. In general, radar sensors use the physical effect of electromagnetic wave propagation and especially the reflection of such waves. The idea is to transmit a radio wave into free space and receive its possible echoes. Out of the received signals, information about the reflecting objects can be obtained. For instance, the time of flight can be measured and translated into a distance, because electromagnetic waves propagate at the speed of light, which is a natural constant.

Compared to typical air and ground surveillance applications, the intended detection ranges, and thus the power requirements of automotive radar sensors are much lower. For example, the average output power of an air surveillance radar can be several kilowatts [24, p. 3], while an automotive radar transceiver has a transmit power of only some milliwatts [25]. Another major difference is the number of present objects in the field of view of the sensor, which is usually much higher for automotive systems. A crowded city scenario with numerous traffic participants is a quite common situation which must be recognized as accurately as possible. Such different requirements entail several particularities for automotive radar sensors which will be outlined in this chapter.

In section 2.1, the most relevant radar waveforms appropriate for use in automotive radar sensors are described. Only the currently prevailing modulation scheme for automotive sensors, frequency-modulated continuous wave (FMCW) is derived in detail. Other operating modes such as pulsed or stepped waveforms exist, however, they are not described in-depth in this chapter.

The following section 2.2 describes the target[1] detection process, where all present objects in the vicinity of the sensor are detected and extracted from the received signals. During this step, a decision is made if a certain measurement cell contains an object echo or only noise components. The data output after this step is already a sparse representation and differs significantly from other sensing technologies like for instance a video camera, which always outputs an image of a predefined size, no matter if the pixels contain useful information.

In the next section 2.3, the task of estimating the direction of arrival is explained. In this context, direction of arrival refers to the origin of an impinging electromagnetic wave or target echo. The result is an indication about the object's position in the three-dimensional space. This kind of information cannot be extracted from a single time-of-flight measurement so that additional steps have to be taken. Besides, the angular measurement is mainly determined by the used antenna arrays and direction of arrival estimation algorithms which will be further detailed in section 2.3.3.

Finally, the state of the art regarding the development of automotive radar sensors is presented briefly in section 2.4, with a special focus on the signal processing hardware.

Parts of this chapter were also presented in [26,27].

2.1 Radar waveforms

Receiving the echoes of a previously transmitted radio wave does not provide any information about the surrounding environment by itself. Consider a pure sine wave for example, which is not adequate to deduce the time of flight due to its ambiguity after the phase has increased by a value of 2π. Even an estimation of the reflected power would become difficult in the case of multiple interfering targets. Only with the help of certain modulation techniques, it becomes possible to extract the necessary information out of the received signals.

Below, a summary of the most important waveform principles for automotive radar is given. The reasoning is based on basic knowledge which can be found in established standard literature, such as [3,24,28,29].

---

[1] The term target is used as a synonym for object in many radar related publications even though no military setting exists.

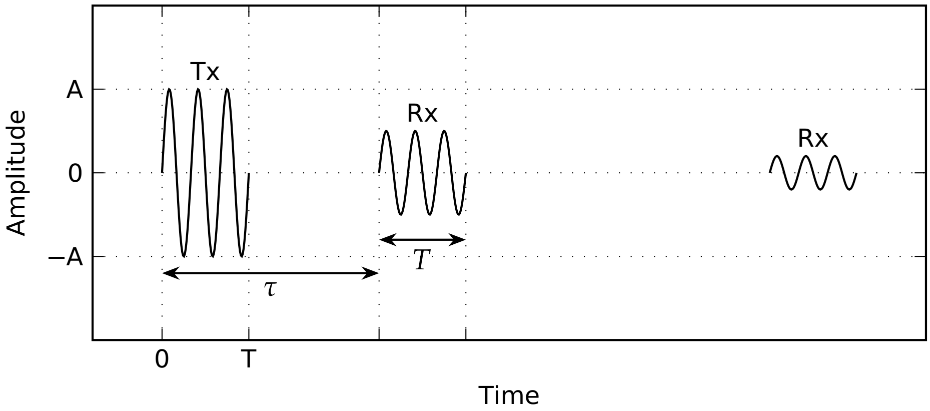


Figure 2.1: Pulsed signal waveform recorded at the radar system. The transmitted pulse *Tx* has been reflected by two different objects whose echoes *Rx* arrive with a delay τ.

2.1.1 Pulsed and continuous-wave (CW) radar

A conventional pulsed radar system uses a very basic amplitude modulation by simply switching the transmitter on and off which results in short radar pulses or bursts with the length T (cf. Fig. 2.1). The transmitted pulse is marked with Tx in Fig. 2.1 and the received echoes are marked with Rx. They are basically a delayed copy of the transmitted pulse with a smaller amplitude. Typically, the receiver uses a matched filter to detect reflections of the transmitted pulses in the received waveform [24, pp. 9,276]. By measuring the time delay τ between transmission and reception, it is possible to estimate the distance of a detected target.

Considering two objects which have a similar distance to the radar system, it is obvious that their radar echoes could arrive at the same instant of time at the receiver. In this case, the two signals could superimpose to a single pulse which can't be separated anymore. The two objects are blended into a single detection and the information about the environment is reduced. The capability of resolving closely spaced targets is referred to as range resolution and can be improved by reducing the pulse duration T.

At the same time, the transmitted energy is directly proportional to the pulse length and it follows that if T is shortened, also the energy and hence the signal-to-noise ratio (SNR) at the receiver is decreased. This is often not desired and especially in the particular case of automotive radar sensors, which require a range resolution well below 1 m, a pulsed radar system reaches its limits.

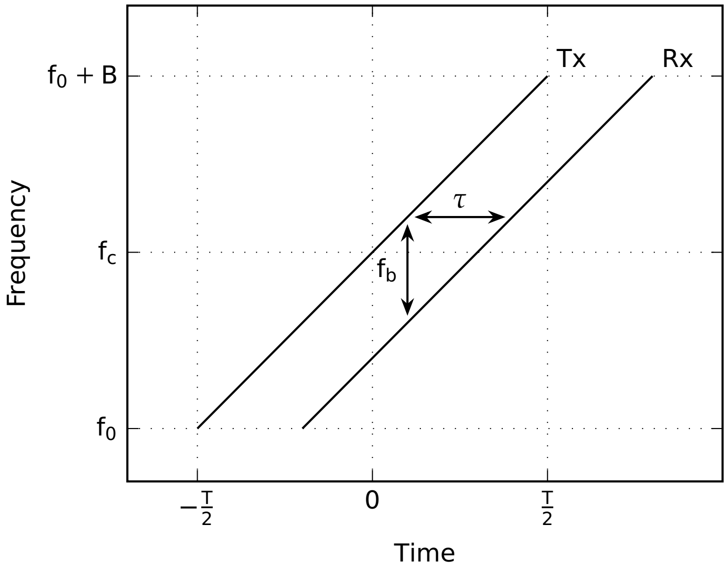


Figure 2.2: Linear frequency ramp recorded at the radar system. The time delay τ of the reflected pulse *Rx* can be translated into a frequency difference f_b.

In contrast, a continuous-wave (CW) radar transmits a waveform and receives its echoes simultaneously. In general, this has the advantage that the transmitted energy can be increased, because the burst length T is decoupled from the achievable range resolution and can thus be maximized independently [28, p. 188]. This important property can be realized by adding some kind of time-varying information to the transmitted waveform. Then, the echoes of the long pulses, which will certainly superimpose in time, can be separated by appropriate filtering techniques at the receiver. This method is often referred to as pulse compression, because a matched filter output at the receiver is much shorter than the actual pulse length. Modulated CW radar is thus very suitable for all applications where a high range resolution as well as high SNR levels are desired.

2.1.2 Frequency-modulated continuous wave (FMCW)

A common modulation scheme uses linear FMCW waveforms, which lead to a so-called frequency chirp or frequency ramp signal. Typically, the transmitted waveform consists of a narrowband signal whose frequency is permanently increased (or decreased), which can be accomplished by a relatively simple voltage-controlled oscillator (VCO). Moreover, the received echoes can be mixed with the transmitted signal, which is also advantageous in terms of hardware effort. Accordingly, a preference of FMCW waveforms over pulsed waveforms can be observed across all manufacturers of automotive radar sensors (cf. section 2.4). More details about prevailing electronic circuits in an FMCW radar system are given in section 5.1.1.

In the case of a frequency-modulated radar system, the transmitted waveform $s_t(t)$ can be modeled by the following equation, where a linearly polarized, plane wave has been assumed. The amplitude A_T is supposed to be constant and a dependency on the location has been omitted. The signal is always referred to the radar system and a monostatic[2] system is presupposed.

$$s_t(t) = A_T \cos\left(\phi_t(t)\right) \tag{2.1}$$

$$\text{with} \quad \phi_t(t) = 2\pi \int_{-\frac{T}{2}}^{t} f\left(\tilde{t}\right) d\tilde{t} \tag{2.2}$$

For a linear frequency chirp or ramp, the modulation is simply modeled with a starting frequency f_0, a bandwidth B and a ramp duration T. For a better understanding, refer also to Fig. 2.2. The term $\frac{B}{T}$ determines the ramp slope m which is an important parameter for system design. Instead of f_0, the *center* or *carrier frequency* of the ramp f_c will be used in the successive derivation, so that the lower integration boundary is chosen as $-\frac{T}{2}$ in (2.2). This choice is motivated by a slight advantage in the deduction of the baseband signal (cf. appendix A.1).

$$f(t) = f_c + \frac{B}{T}t = f_c + mt \tag{2.3}$$

By substituting (2.3) in (2.2), the transmitted waveform can be written in the following form:

$$\phi_t(t) = 2\pi \int_{-\frac{T}{2}}^{t} f_c + m\tilde{t}\, d\tilde{t} = 2\pi f_c t + \pi m t^2 + \phi_0 \tag{2.4}$$

$$s_t(t) = A_T \cos\left(\phi_t(t)\right) = A_T \cos\left(2\pi f_c t + \pi m t^2 + \phi_0\right) \tag{2.5}$$

The waveform $s_t(t)$ will propagate away from the radar sensor until it has been reflected by a surrounding object. Supposing that no phase jump occurs, the reflected signal $s_r(t)$ which is again received by the radar sensor is just a

---

[2] A radar system for which the transmitter and the receiver are at the same location.

damped, time delayed version of the sent signal.

$$s_r(t) = \alpha_2 s_t(t - \tau) = A_R \cos(2\pi f_c(t - \tau) + \pi m(t - \tau)^2 + \phi_0) \tag{2.6}$$

where $\alpha_2 = \frac{A_R}{A_T}$ accounts for the two-way free space loss and any absorption. The time duration τ expresses the wave propagation delay and depends on the target distance.

From Fig. 2.2 it can be observed that the delay τ is related to the frequency difference f_b between transmitted and received signal. This property can be used in a smart way by the receiver, because it is sufficient to measure only this difference, the so-called beat frequency f_b. In practice, a frequency mixer could be employed, which directly generates the frequency difference of the two signals $s_t(t)$ and $s_r(t)$ by simply multiplying them. Further mixing products like the signal component at the sum of the two input frequencies will also occur, but usually they can be effectively filtered out. Remarkably, the frequency f_b is much lower compared to the carrier frequency f_c which simplifies subsequent circuit and component design.

Obviously, the delay τ remains constant as long as a target is stationary. For a moving target, τ is varying during the chirp which induces an additional frequency shift, commonly known as Doppler effect. Considering a moving target at a distance r, with a constant velocity v_r, in a direction radially away from the sensor, the two-way propagation delay τ can be written as

$$\tau = \frac{2r}{c} + \frac{2v_r t}{c} \tag{2.7}$$

by using the propagation speed of radio waves, the speed of light c.

The resulting signal at the mixer output can be derived by multiplying (2.5) with (2.6) and inserting (2.7) for τ. The outcome is described by (2.8) and its complete derivation is included in the appendix A.1.

$$s_b(t) = s_t(t)\, s_r(t) = A_B \cos\left(2\pi\left(\frac{2f_c r}{c} + \underbrace{\frac{2f_c v_r}{c}}_{f_v} t + \underbrace{\frac{2mr}{c}}_{f_r} t\right)\right) \tag{2.8}$$

The frequency f_b of the baseband signal is composed of two components, a range dependent part f_r and a velocity dependent part f_v:

$$f_b = f_r + f_v = \frac{2mr}{c} + \frac{2v_r}{\lambda} \tag{2.9}$$

where the carrier frequency f_c has been substituted by the wavelength λ for convenience. In (2.9) it can be seen that two unknowns, namely the target range r and the radial target velocity v_r contribute to the resulting beat frequency. Thus, the estimation of both parameters from a single frequency measurement is ambiguous and additional measures have to be taken in order to find the respective values.

2.1.3 Chirp sequence modulation

In automotive applications, a simultaneous measurement of range and velocity of all targets is desired, but not possible with a single FMCW measurement as outlined in the previous section. From (2.9), it can be observed that the two parameters r and v_r have a different weighting in the composition of the beat frequency. More precisely, the range-dependent contribution is influenced by the ramp slope m, while the velocity-dependent part is not. This enables methods to estimate both, range and velocity, from a single beat frequency. Two approaches can be identified, each with its own advantages and disadvantages.

Matching Several measurements with different ramp slopes m are used. This results in a linear equation system, which has to be solved. In general, the more targets are present, the more frequency components will superimpose. In the end, even more ramps are required in order to find an unambiguous solution in multi-target scenarios. Furthermore, the computational complexity grows drastically in this case so that this approach is only suitable for a bounded number of reflections [30].

Fast chirp The ramp slope m is made steep enough, so that the velocity-dependent term in (2.9) gets so small that it can be neglected [31]. Then, the target range can be estimated directly from the measured beat frequency. Even more important, the estimation of the frequency components does not get more complicated when multiple targets are present, so that this approach scales very well in scenarios with a high target count. However, the velocity is not available from the frequency measurement and additional steps have to be performed in order to measure v_r.

With regard to automotive scenarios, the target count is usually large due to present roadside infrastructure and traffic participants. Furthermore, the growing resolution capabilities engender a higher number of targets because even closely located objects can be separated. This trend is one reason for the adoption of chirp sequence modulation schemes by automotive radar sensors, because a matching based approach cannot keep up with an increased number of targets, as mentioned above.

A chirp sequence uses a series of fast chirps, each with the same ramp slope m and starting frequency f_0. Furthermore, a constant time relation between all ramps, the ramp repetition interval T_{rri} should be adhered in order to simplify the following signal processing tasks. The usage of multiple chirps has mostly two reasons:

- A fast chirp is usually a very short pulse ($T < 100\,\mu\text{s}$) due to its high ramp slope m. Hence, the transmitted energy is rather small which deteriorates the SNR in the first place. Though, if multiple chirps are evaluated together, the SNR can finally be regained after signal processing.

- The velocity cannot be estimated from a single fast chirp. By using a chirp sequence the estimation of v_r becomes possible by evaluating the phase information of the resulting beat signals.

For the subsequent derivation according to [32], the distance of a moving target is modeled by using a non-fixed distance r_k which varies from chirp to chirp k.

$$r_k = r_0 + v_r T_{rri} k \tag{2.10}$$

Reusing the expression of the baseband signal from a single chirp (2.8) and inserting the equation for a varying target distance from (2.10), the baseband signal of the k-th chirp can be written in the following form:

$$
\begin{aligned}
s_{b,k}(t) &= A_B \cos\left(\frac{2\pi}{c}\left(2f_c r_k + 2m r_k t\right)\right) && (2.11)\\
&= A_B \cos\left(\frac{2\pi}{c}\left(2f_c r_0 + 2f_c v_r T_{rri} k + 2m(r_0 + \underbrace{v_r T_{rri} k}_{<< r_0})\, t\right)\right) && (2.12)\\
&= A_B \cos\left(\frac{2\pi}{c}\left(\overbrace{2f_c r_0 + \underbrace{2f_c v_r T_{rri} k}_{\text{Velocity}}}^{\text{Phase}} + \overbrace{\underbrace{2m r_0 t}_{\text{Range}}}^{\text{Frequency}}\right)\right) && (2.13)
\end{aligned}
$$

For this expression, a sufficient high ramp slope m has been supposed so that the Doppler-dependent frequency part f_v from (2.8) can be neglected. A typical Doppler frequency for fast moving objects in a traffic situation might be in the order of 10 kHz, while the range-dependent part of the beat frequency f_r amounts easily to several megahertz or more [31].

Furthermore, parts of the last term in (2.12) can be omitted if the velocity v_r is small enough, so that the additional distance $v_r T_{rri}$ between two chirps k and $k+1$ does not induce a significant different beat frequency. Consequently, the measured frequency can be considered as constant from chirp to chirp which simplifies the required signal processing steps. This assumption is true if the target's movement during the full measurement cycle, i.e. during all K ramps of the whole chirp sequence, is below the range resolution of the system. It is appropriate for most state-of-the-art sensors and for usually encountered object velocities in automotive scenarios.

However, with increasing bandwidths and the augmented range resolution of future radar sensors, the influence of the movement on the range measurement can become significant. The effect is also referred to as *range cell migration* because the beat frequency is not constant and traverses several range cells during the chirp sequence. Advanced spectrum evaluation methods can help to compensate for the diverging beat frequency [33], however they are not covered by this work.

Apart from the beat frequency, the phase of the baseband signal carries additional information. From (2.13) it can be seen that a moving target modifies the measured phase value of a single chirp. Even though the measured phase term depends also on the absolute distance r_0, an estimation of v_r is possible when evaluating the variation of the phase over time, i.e. from chirp to chirp.

The phase measurement is in general very sensitive to any radial movements. Already a small motion in the order of the wavelength, which means around 4 mm at 77 GHz, provokes a phase shift of 2π. Such micro-movements can be captured over a certain period of time so that an accurate estimation of the radial target velocity becomes possible. A high velocity resolution of well below 1 m/s is an important feature of this measurement principle [7,34].

For a constant radial velocity, the phase shift between two consecutive chirps is also constant as long as the ramp repetition interval T_{rri} is not varied. This property simplifies the subsequent signal processing steps which will be further explained in the next section. The spacing between two consecutive frequency chirps T_{rri} determines the maximum unambiguous velocity which can be measured. This property is a consequence of the 2π phase ambiguity. A shorter T_{rri} is in general desired, because it increases the unambiguous velocity.

2.1.4 Spectrum evaluation

The classical approach to separate multiple targets and to estimate their corresponding distances to the sensor is to transform the measured baseband signals into the frequency domain. Every single target will cause a discrete beat frequency component which can directly be converted into a distance if a fast chirp modulation has been used (cf. section 2.1.3). A conventional fast Fourier transformation (FFT) in combination with an appropriate window function can be used as efficient and robust estimation algorithm.

Furthermore, it is possible to estimate the chirp to chirp phase offset and hence the velocity by a second FFT which can be applied to the result of the first transformation. The outcome is a two-dimensional spectrum which is often referred to as *range-Doppler (r-D) matrix*.

A schematic describing the signal flow using FFTs can be seen in Fig. 2.3. Each frequency chirp is separately transformed by the first FFT which will also be referred to as *Range FFT*. In the resulting data vectors, which look similar for all frequency ramps, three targets are marked with different colors. For the moment, two targets share the same frequency bin so that they can only be recognized as a single target.

The second FFT is also referred to as *Velocity FFT* and gets directly applied to the result of the first FFT, but in another dimension. For each frequency bin of the Range FFT, the second transform is computed over all ramps of the chirp sequence. An important prerequisite is a constant ramp-to-ramp spacing so that a possible phase offset of a constant velocity will appear equal over the whole sequence. With this technique, the two targets sharing the same range bin can be separated due to their different velocities, which can be seen in the resulting range-Doppler matrix on the right of Fig. 2.3. This separation capability helps to detect moving targets in scenarios with many overlapping objects. Even if multiple targets with the same distance share a common range bin, they can be further separated according to their relative radial velocities.

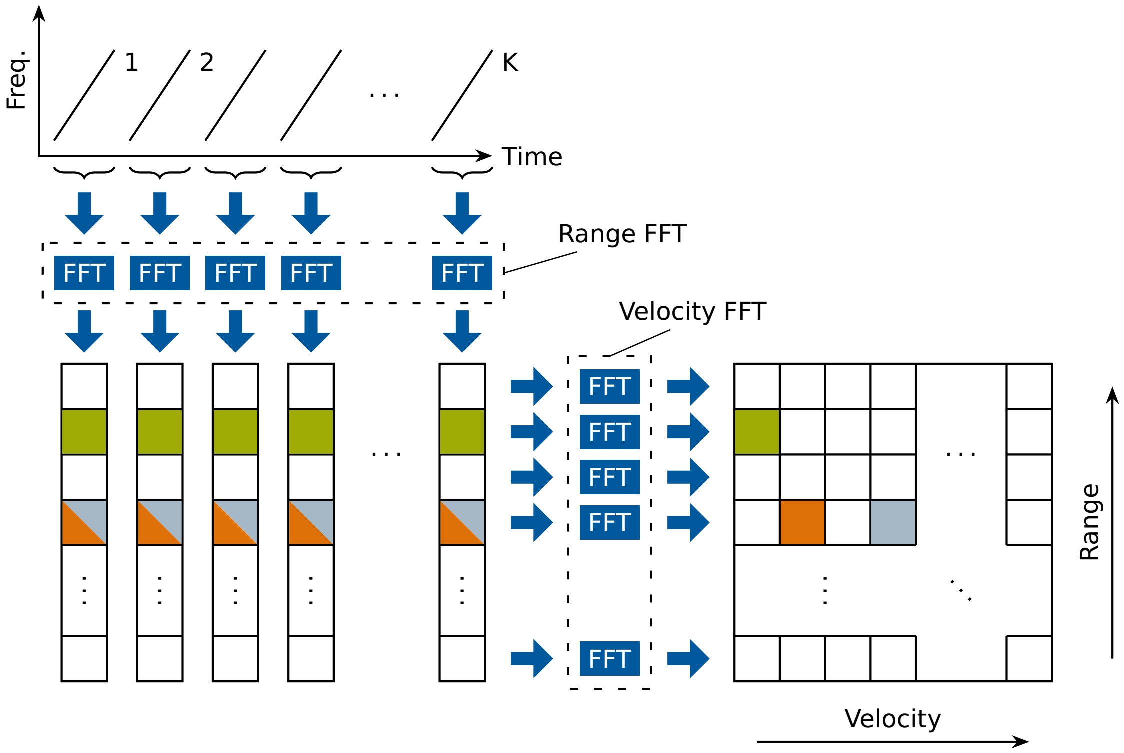


Figure 2.3: Chirp sequence modulation waveform and corresponding spectral evaluation using FFTs. Cells containing a target echo are colored.

A further advantage of the two-dimensional spectrum evaluation is an additional gain in SNR because the transformation into the frequency domain behaves like a matched filter. After the frequency transformation, a single target maps to a discrete frequency, so that the whole signal energy of the target echo is concentrated into one cell of the FFT while the present noise is distributed equally across the whole spectrum. The signal-to-noise ratio increases and the effect is often referred to as *processing gain* [35].

Consequently, the two-dimensional frequency estimation should be carried out for all received signals of the chirp sequence and the full spectrum needs to be evaluated even though the occupied frequency cells are rather sparse. In real world scenarios only about one percent of the frequency bins contain valid targets (cf. chapter 5.3.2) and in the first place it seems inefficient to compute all bins. Nevertheless, this step is crucial for an enhanced noise margin and improved detection performance so that the primary drawback of a short chirp duration in terms of low SNR can be avoided.

The costs of a full spectrum evaluation by using a 2D-FFT become apparent when regarding the demands placed on the processing system. The FFT by itself does not reduce the amount of data in any way, i.e. it only transforms an input vector to an output vector of the same length. Hence, the data throughput equals the raw data rate at the input, originating from multiple analog-to-digital converters operating at rates of several tens or hundreds of megasamples per second (MSPS) [3, p. 387]. This sample rate gets multiplied by the number of parallel channels as well as the word size (resolution), so that a data rate of 10 GBit/s and above is to be expected.

Furthermore, the data of a full 2D matrix has to be stored in memory before the Velocity FFT can be computed, because it requires data from chirp 1 to chirp K. For certain high resolution applications, the typical length of one FFT in either dimension might be in the order of 1024, or even greater [36]. In this case, the total storage size of the full matrices can amount up to 100 MByte, because the number of cells exceeds clearly one million and multiple matrices have to be stored, one for each channel.

The maximum length of the chirp sequence, i.e. the number of ramps K is in general only bounded by the available memory and processing capabilities. The advantage of a longer chirp sequence is certainly the improved velocity resolution as well as an improved SNR due to the larger processing gain. On the downside, a long measurement time can provoke blurring effects in dynamic scenarios which is also not desirable. Furthermore, the duration of a full chirp sequence defines the duration of one measurement snapshot or cycle. Consequently, a large number of ramps will decrease the maximum possible cycle rate of the system. A typical value for automotive applications is a maximum measurement time of 40 ms for all K ramps, so that a cycle rate of 25 Hz can be achieved. [3, pp. 388,393,397]

2.2 Target detection

The radar echoes at the receiver are usually superimposed by random noise components or disturbing interference from other signal sources. Hence, a decision has to be made, if the received signals contain valid radar detections or just random noise components. In general, a wrong decision deteriorates the sensor's performance and it is desired to minimize the following two faults which could arrive:

False positive A random signal component is classified as valid radar echo. The sensor outputs a wrong detection, i.e. it detects an object which does not exist. This case is also referred to as *false alarm*, dating back to the days where radar was primarily used to recognize an approaching enemy.

Missed detection The received radar echo is too weak and gets masked by other signal components like noise or stronger targets in the vicinity, so that it cannot be detected anymore. This occurs frequently for tiny targets with a small reflective surface, for which the SNR is simply too low. The ability to detect such weak radar echoes can be crucial when a detailed and fine-grained map of the environment is required.

A trade-off between the two properties explained above is required, because both cannot be improved at the same time. For the system designer it can be important to limit the probability of false alarm, so that the reliability of the data can be guaranteed to a certain degree. The common approach is to assure a constant false alarm rate, which normally requires some dedicated processing techniques prior to the detection decision. The goal is to eliminate the effect of locally and temporally different noise characteristics so that the detection performance remains the same irrespective of changing operating conditions. In the following section, the theory and different realizations of a *constant false alarm rate (CFAR)* processing will be introduced.

The starting point for the target detection is the two-dimensional frequency spectrum which can be obtained if a chirp sequence modulation is used as it has been explained in section 2.1.3. The actual detection is performed in the power spectrum, which means that the absolute square values of the complex FFT data have to be computed first.

The basic principle is to observe the power level of each frequency bin compared to its local environment in the spectrum. A cell with a larger amplitude is supposed to contain a target echo and should be considered as valid detection. At the same time, all cells whose power level does not stand out from their neighborhood will most likely contain unimportant noise components.

Furthermore, it is possible to benefit from the existence of multiple receiving channels by integrating the data prior to the detection. One such possibility is referred to as *non-coherent integration (NCI)* and is described more in detail in section 2.2.2.

2.2.1 Constant false alarm rate (CFAR)

A very basic approach for a decision if a certain cell contains a valid target would be to use a fixed threshold. Accordingly, if the power level of a cell exceeds this threshold it can be considered as valid target, otherwise it would be considered as noise. However, this procedure imposes some limitations due to the varying operational and environmental conditions.

For instance warmer ambient temperatures can cause all component's temperatures to rise equally which in turn increases the influence of thermal noise components. Furthermore, so-called signal clutter originating from distributed reflections on the ground surface or during rain and snowfall, can increase the local signal power level in certain ranges [28, pp. 82 – 88]. This is caused by the superposition of many small echoes which resemble noise components because an extraction of a distinct frequency of a single target is impossible. In summary, the noise floor of a radar system is changing over time and it can also be locally different in the spectrum. Such effects have to be eliminated prior to the detection if a constant false alarm rate (CFAR) is desired and a simple fixed threshold decision is not sufficient.

The design of a CFAR detector has been of major interest since an automated target detection by means of digital signal processing became possible. The basic principle of all CFAR detectors is more or less identical even though a wide range of different classes and variants have been reported in the past decades. The main idea is to divide the detection space into many small cells and to use a locally different threshold for the detection decision. Consequently, an adaptive threshold based on the local noise characteristics is calculated individually for every cell under test (CUT) in the spectrum. The procedure is repeated in each measurement cycle so that it can adapt to changing operating conditions.

Generally, the input cells for the CFAR detector originate from the power spectrum, which is often referred to as *square law detector* in literature. The cell size can be chosen identical to the size of one frequency bin, so that the result of an FFT can be directly fed into the CFAR algorithm. More precisely, a single cell of the CFAR processor corresponds to exactly one absolute square value of the FFT output in this case.

A major challenge is the accurate estimation of the noise level, which is required to achieve a constant-false alarm rate. In general, the neighboring cells around the CUT are considered to contain mostly noise so that they can provide enough information to estimate the local noise distribution. All cells which are used for the noise estimation are referred to as *window cells*. Other cells usually located in direct vicinity to the CUT do not contribute to the noise estimation and are referred to as *guard cells*. These cells with immediate proximity to the CUT are not taken into account because the energy from a possible target in the CUT could expand to the neighboring cells which in turn could distort the result. The noise estimation procedure is often implemented as a sliding window filter, similar to a convolution, so that parts of the previous estimation results can be reused.

An exemplary schematic of a CFAR detector is illustrated in Fig. 2.4. The window cells are taken from the left and the right neighborhood of the CUT in an one-dimensional manner. If the input cells originate from a 2D-FFT, then every column or row can be processed separately. In general, for a two-dimensional spectrum, a rectangular window is likewise imaginable.

The size and the shape of the CFAR window are important design parameters because they determine mainly the estimation accuracy and the adaptiveness to altering noise conditions. On the one hand, a larger window size reduces the statistical estimation error, but on the other hand local differences in the noise level can be blurred by a large window. A trade-off has to be made between a statistical estimation error and the local sensitivity of the adaptive threshold due to smoothing effects. Furthermore, the computational effort becomes more relevant with increasing window sizes.

The major distinctive feature of all CFAR detectors is the method to estimate the local noise distribution and to derive a decision threshold which maintains a constant-false alarm rate. In the following subsections, two of the most prominent variants are presented, the cell-averaging (CA) and the ordered-statistic (OS) CFAR.

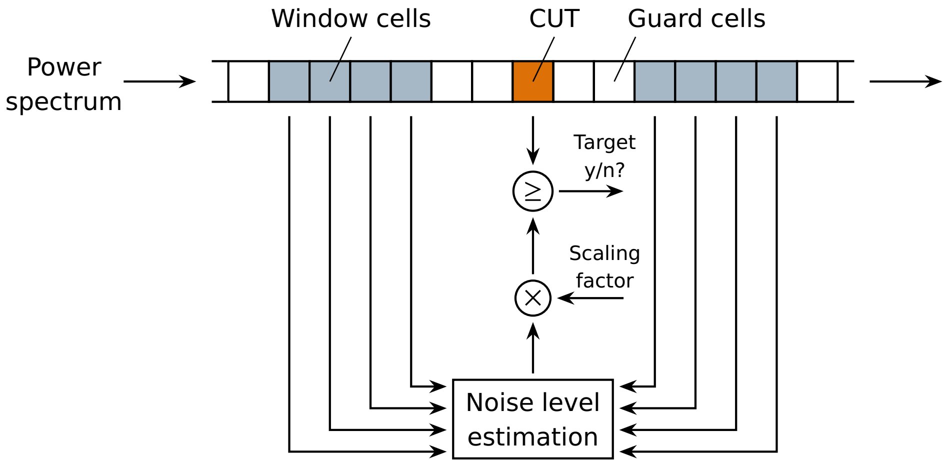


Figure 2.4: Generic CFAR processing scheme operating on an one-dimensional window. The values of multiple window cells around the CUT are used to obtain a noise level estimate which in turn can be used for a target detection decision. For a chirp sequence modulation, one cell usually corresponds to one entry in the range-Doppler matrix in Fig. 2.3.

CA-CFAR

A simple yet powerful approach is to use the mean value of a number of window cells as an estimate for the local noise level. This estimate can subsequently be used as a target decision threshold, after an appropriate scaling value has been applied to it. The procedure is a very common approach and is known as cell averaging CFAR, or CA-CFAR [24, p. 296].

This procedure relies on the assumption that all window cells contain only noise components and hence the mean value is a good estimate of the noise variance. In the case of white Gaussian noise, the estimated value is actually corresponding to the maximum likelihood estimator. However, for many radar systems the assumption of normal distributed noise turns out to be inaccurate [37] and more sophisticated CFAR methods are required.

Once an estimate for the local noise level could be obtained by the CA-CFAR, it has to be scaled appropriately before it can be used as a threshold for the target decision. This procedure assures that a certain interspace to the noise floor can be maintained, i.e. the power level of the CUT has to lie significantly above the estimated noise level.

A direct comparison without scaling would result in an extremely high false alarm rate P_{fa} which is demonstrated in the following derivation, according to [38, p. 347]. For the calculation of this exemplary false alarm rate, a complex Gaussian noise distribution is assumed. Thus, the resulting probability distribution of the absolute square values in the power spectrum is an *exponential distribution*.

If the noise distribution is known, the rate of false alarms can be obtained by simply integrating the right tail of the probability density function, i.e. the portion above the chosen threshold Th. The integral expresses the rate of false alarms, because it describes the frequency of occurrence of noise components greater than Th. Hence, for a certain noise variance σ^2 and threshold Th, the false alarm rate P_{fa} corresponds to the complementary value of the *cumulative distribution function*[3]. For an exponential distribution, this leads to:

$$\mathrm{P}_{fa}(Th) = \mathrm{P}(X > Th) = 1 - \mathrm{P}(X \leq Th) = \exp\left(-\frac{Th}{\sigma^2}\right) \tag{2.14}$$

Setting $Th = \sigma^2$, i.e. using the (estimated) noise power level as detection threshold, results in $\mathrm{P}_{fa} = e^{-1} = 0.37$. This means that in average every third cell would provoke a false alarm.

For most applications, such a high rate of false alarms would stress the following signal processing steps with too many noise components. Consequently, a scaling factor is used to control the false alarm rate according to the application's requirements. Generally, for many applications the false alarm rate is chosen to be very low, e.g. in the order of 10^{-6}.

OS-CFAR

In the case of white Gaussian noise, the CA-CFAR performs very well in single target scenarios. However, in a multi-target environment, the estimated noise level will deviate due to interfering targets inside the window cells. Robust statistics can be used in order to suppress outliers arising from other targets inside the window. A commonly used variant is the ordered-statistic (OS-CFAR) which relies on a sorting of the values inside the window, similar to a median filter [39].

In the same manner as for CA-CFAR, the window cells around the CUT, excluding the guard cells, are used to form the noise estimate. But instead of

---

[3] The cumulative distribution function is commonly defined as $F_X(x) = \mathrm{P}(X \leq x)$ and corresponds to the integral of its probability density function: $F_X(x) = \int_{-\infty}^{x} f_X(t)dt$

calculating the mean value, all cells inside the window are sorted as a first step. Then, the k-th value of the sorted list serves as an estimate of the local noise level. The remaining portion of the algorithm is identical to the CA-CFAR. A scaling value which controls the false alarm rate is applied to the noise estimate and subsequently a target decision is made.

The robust statistics approach of the OS-CFAR is considered as superior CFAR procedure for automotive radar applications because it performs significantly better than the CA-CFAR in scenarios with interfering targets [40,41]. However, the computational effort is considerably higher due to the complexity of the sorting, especially at larger window sizes. An optimized evaluation scheme of the OS-CFAR which avoids a full sorting will therefore be investigated in this thesis (cf. section 3.2.1).

2.2.2 Non-coherent integration

Even though the detection takes place before the angular processing, the data of multiple receiving channels can be used to further improve detection performance. An integration or averaging of all channels prior to the detection step turns out to be beneficial [28, p. 328], assuming that the noise components are independent and identically distributed (i.i.d.).

Special attention should be paid to the phase relationship of the signals between adjacent channels. In general, the relation is not known prior to the angle estimation and can take any value. When summing up the complex values of all channels, the signals can interfere either constructively or destructively. In order to avoid a cancellation of the signal power, the integration takes place in the power spectra, which is also known as non-coherent integration (NCI). Other integration methods like a coherent or binary integration exist, however they have not been used in this work.

In the following derivation, the noise components are modeled as additive-white Gaussian noise which means that a zero-mean normal distributed signal $n[t]$ is added to the reflected signal echo $s[t]$. Finally, the resulting signal $\hat{s}[t]$ at the receiver consists of the desired signal superimposed by an arbitrary noise component (cf. Fig. 2.5). As mentioned above, the noise components $n[t]$ are assumed i.i.d. for all channels.

Usually, the detection process takes place in the frequency domain, which is why the spectral characteristics of the additive noise components are likewise

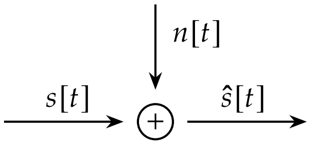


Figure 2.5: Additive white Gaussian noise model.

important. It can be shown that both, the real and imaginary parts of the noise components $N[k]$, follow a zero-mean normal distribution after transformation into the frequency domain [38].

$$\hat{s}[t] = s[t] + n[t] \tag{2.15}$$

$$\begin{array}{c}\circ\\ |\\ \bullet\end{array}$$

$$\hat{S}[k] = S[k] + N[k] \tag{2.16}$$

The variance of $N[k]$ depends on the input noise variance ${\sigma_n}^2$ as well as on the length of the input signal, i.e. the length of the discrete Fourier transform (DFT). It must be equal for all receiving channels due to the assumption of i.i.d. random variables.

The actual detection process takes place in the power spectrum, which can be obtained by summing up the squared values of the real and imaginary parts of $\hat{S}[k]$. For cells containing only noise components, the actual signal term $S[k]$ is zero and only $N[k]$ contributes to the power spectrum. It can be seen that in this case, the sum of two squared, i.i.d. Gaussian variables is formed:

$$\left|N[k]\right|^2 = \mathrm{Re}\left(N[k]\right)^2 + \mathrm{Im}\left(N[k]\right)^2 \tag{2.17}$$

$$\left|N[k]\right|^2 \sim \chi^2(2) \tag{2.18}$$

The result for the squared noise magnitude $|N[k]|^2$ is a random variable following a chi-squared distribution $\chi^2(d)$ with $d = 2$ degrees of freedom. The expected value of this $\chi^2(2)$ distribution determines the height of the noise floor in the power spectrum. Furthermore, the variance of the distribution causes a random deviation from the nominal mean value so that the noise floor in the spectrum will never appear as a perfectly flat surface.

When summing up a number of M receiving channels, i.e. M i.i.d. random variables, the result will again be chi-squared distributed but with a higher degree of freedom.

$$\left|N_{NCI}[k]\right|^2 = \sum_{i=1}^{M} \left|N_i[k]\right|^2 \sim \chi^2(2M) \tag{2.19}$$

According to this chi-squared distribution, the expected value of the noise power scales linearly with the number of channels M. Furthermore, in the case of an NCI, the signal power is raised by the same amount, i.e. linearly with M. Hence, the resulting SNR, defined as the ratio between the mean values of signal and noise power, is not directly enhanced. In other words, the distance to the noise floor of cells which contain a target echo is not increased. This result is in contrast to the DFT, where a coherent integration takes place and the SNR can be improved solely by taking longer input sequences.

However, the NCI decreases the variance of the noise power in relation to the signal power which has an effect on the possibility of false alarm. The noise floor in the spectrum appears to be more flat after a certain number of channels has been averaged. Consequently, a real signal echo can be distinguished more easily and with a higher confidence from random noise components. The effect is also referred to as non-coherent integration gain [42].

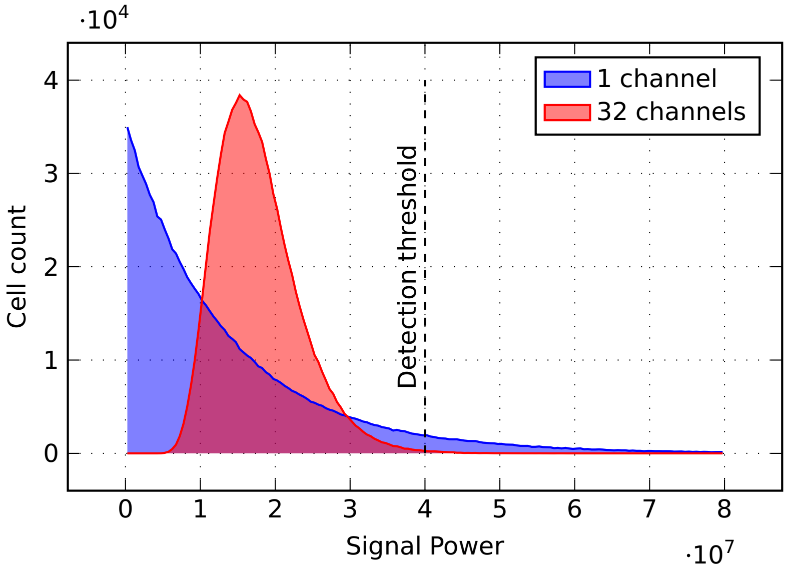


Figure 2.6: Distribution of the signal power in a measurement comprising only noise components. The shape of the distribution is changed after the non-coherent integration of 32 channels.

An example measurement is depicted in Fig. 2.6, comparing the noise distribution of one channel and the distribution after the integration of 32 channels. The mean value of the two distributions is identical, while the variance of the red histogram is much smaller. It can be observed that for the same threshold level, a lower probability of false alarm can be achieved after the integration of 32 channels because the area below the red curve exceeding the threshold is considerably smaller than the area below the blue curve. The other way round, if the same probability of false alarm is desired, then a lower threshold level could be used in the case of 32 integrated channels. This would increase the detection rate accordingly while maintaining an identical false alarm rate.

2.3 Direction of arrival estimation

In the previous sections, different radar waveforms and their mode of operation were described which enable the measurement of the distance r toward an object and its relative velocity v_r in radial direction. Without any further information, the object position in space is still not uniquely determined and can be anywhere on a sphere with the radius r. The exact location in a three-dimensional space can be described by providing the angle in azimuth and elevation, which express the angular offset between the direction toward the target and a reference direction.

For the sake of simplicity, the rest of this thesis will only focus on the azimuthal angle θ which is more useful in the context of automotive radar. However, the following approaches can be easily adopted to the elevation angle which simply lies in a different plane. State of the art sensors like the Bosch MRR are evaluating both angles in order to obtain the unique three-dimensional coordinates in space of each reflection [3, p. 382].

The problem of estimating the angle θ is often referred to as direction of arrival estimation (DOA) and is encountered frequently in many radar and sonar applications. A variety of approaches exist and the most frequently used are described briefly in the following paragraph.

Steerable antenna A commonly known method and often the first choice for airspace monitoring, for instance at commercial airports. These systems use a mechanically rotatable antenna with a high directivity, such as a parabolic antenna which has a very high gain into a certain direction.

The detection space is then scanned step-by-step for each possible angle θ. Once a target echo is received, its direction of arrival is immediately known from the current orientation of the antenna.

Antenna array Multiple antennas which are separated spatially can provide information about the direction of arrival of a signal. Similar to the range measurement, an additional propagation delay at the different receiver elements is observable which depends on the direction of arrival. Even though very small and often in the order of a single wavelength, this delay can be measured as a phase offset. With the help of appropriate array processing algorithms, the corresponding DOA can be found. Other phased array radar techniques like electronic beamsteering on the transmitter side can also be employed with the help of antenna arrays. However, their functional principle is slightly different [43].

Synthetic-aperture radar (SAR) A radar sensor which is mounted on a moving platform can acquire multiple measurements from different positions. By evaluating the gathered data altogether a synthetic aperture is created, similar to an antenna array looking to the side of the moving direction. The technique is often used for airborne earth observation and can provide a very fine resolution. [44]

Most of the current automotive radar sensors use an antenna array in order to provide a possibility for measuring the angle of incidence. With the help of two-dimensional arrays, both, the direction in azimuth and in elevation can be estimated. Furthermore, no mechanical parts are required at all, which helps to make the sensor reliable and maintenance-free over its lifetime. Beside the class of array-based sensors, mechanically scanning sensors have also been used with success for automotive applications. For example, the ARS300 radar sensor from Continental contains a steerable antenna implemented as a rotating drum [3, p. 386]. Finally, the SAR approach could be employed for vehicles, too, because they form a moving platform [45]. Though, the resolution can mainly be increased at the left and right side of the car, i.e. perpendicular to the driving direction.

In summary, radar sensors incorporating antenna arrays are considered as the prevailing class of automotive radar sensors in the near future. The underlying array processing concepts are explained in the next section, because they will play a crucial role when investigating the corresponding hardware architectures.

2.3.1 Antenna arrays

An antenna array is formed by multiple individual antennas which are distributed in space, so that the DOA with the angle θ of a signal can be estimated. This is possible because the different amplitude and phase relationships at the antenna elements provide information about the origin of the reflected signal. As mentioned previously, the time shift due to an additional propagation delay can be recognized, or alternatively a different amplitude level that results from a directional characteristic of the antenna can be evaluated.

Preconditions

In the case of automotive radar sensors, the distance to a possible reflector is usually large compared to the array size. Hence, the incident signal can be modeled as a *plane wave* impinging on the sensor which simplifies all following derivations. Most importantly, the occurring time delays at the antennas can be decoupled from the target range so that only a dependence on the DOA remains. For the sake of simplicity, only the azimuthal DOA is considered in the following derivation. Though, the elevation angle estimation works in the same way and all described measures can likewise be applied to it.

Furthermore, a second simplification can be used, the so-called *narrowband assumption*. As the name suggests, the requirement for this assumption is that the occupied bandwidth B is small. Many authors [46–48] establish the following condition for the narrowband case:

$$B\,\Delta\tau_{\mathrm{max}} \ll 1 \tag{2.20}$$

where B is the used bandwidth and $\Delta\tau_{\mathrm{max}}$ is the maximum time delay occurring between two elements of the array. In other words, the travel time across the array has to be significantly smaller than the inverse bandwidth of the signal.

Then, the additional time delay due to the extended path length at the individual receiver elements can be expressed as a phase shift. This is very convenient if the DOA estimation is executed in the frequency domain, because the estimation algorithms can be directly applied to the FFT results of the range-Doppler processing step (cf. also section 2.1.4).

More specific conditions for a narrowband signal exist [49], however the simple relationship (2.20) is sufficient for all further derivations in this section. Furthermore, even though the methods and algorithms presented below rely on the narrowband assumption to be true, they can be extended to the wideband case [50]. This will be of importance when increased signal bandwidths in the order of 4 GHz in combination with large MIMO arrays are employed.

Uniform linear array

The geometry of the used antenna array plays a crucial role because it directly influences the resulting phase shift. An often encountered array geometry is the uniform linear array (ULA) where all elements are arranged on a straight line with identical spacing between them. Depending on the DOA θ, a relative phase shift ϕ_m can be recognized at each individual antenna element Rx_m (cf. Fig. 2.7). Considering a plane wave impinging on a ULA, the relative phase shift at the m-th element amounts to:

$$\phi_m = 2\pi \frac{(m-1)\,d}{\lambda} \sin(\theta) \tag{2.21}$$

where d is the distance between two antennas and λ is the signal's wavelength.

In Fig. 2.8, a picture of a receiving antenna array of a 77 GHz sensor is shown. It is a ULA with 12 antennas which are spaced in the horizontal direction, allowing for a DOA estimation in azimuth. The antennas are directly integrated onto a printed circuit board, which also holds the electronic circuits. It can be seen that multiple patch elements[4] are connected together in the vertical direction which results in a narrower and more focused beam in elevation. The received signals are guided on the circuit board with the help of a microstrip transmission line toward the bottom border of the picture.

In practice, each antenna of the array receives its distinct signal which differs depending on the antenna's location in space. The problem to solve is to estimate the angle of arrival θ based on the measured array response. Below, some useful concepts assisting in the angle estimation process will be described.

---

[4] A patch antenna consists of a rectangular metal surface which can be directly fabricated on a printed circuit board. Its geometry is closely associated to its frequency characteristics [51].

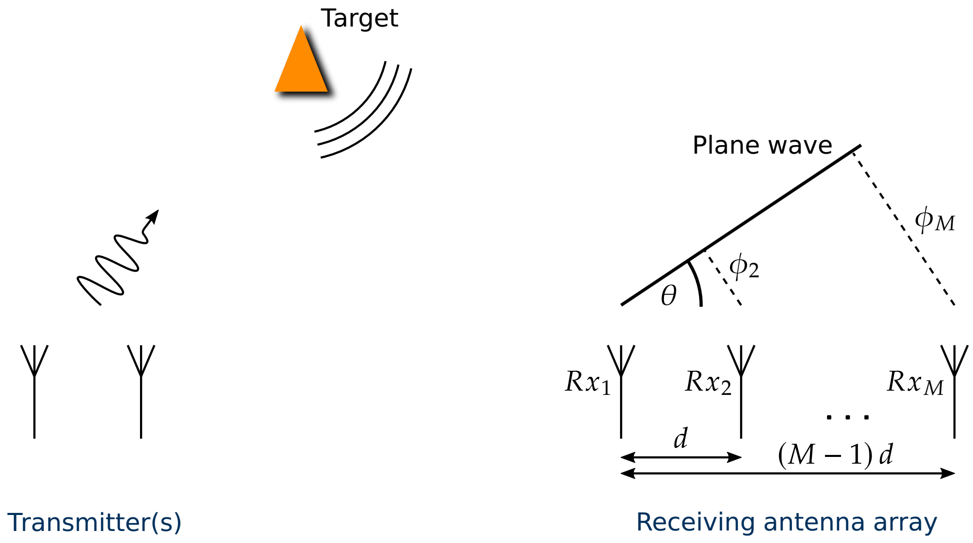


Figure 2.7: Direction of arrival (DOA) estimation with the help of a uniform linear array (ULA) at the receiver side. The impinging signal can be modeled as a plane wave simplifying further derivations.

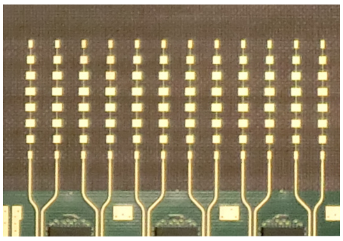


Figure 2.8: A picture of a ULA with 12 elements on a printed circuit board. The distinct patch antennas are spaced with approximately 4 mm in the horizontal direction.

Steering vector

A lot of research has been carried out to estimate the DOA as good as possible and many different algorithms are established in this field which is also known as array processing [46, 47]. An important modeling technique in this context is the so-called steering vector $\boldsymbol{a}(\theta)$. It describes the resulting array output for a single signal source residing at the angle θ. It contains one entry for each antenna element of the array and carries the magnitude and phase response of each channel depending on the DOA θ. Very importantly, the array response can be decoupled from other signal parameters like distance or reflectivity of the target. As a consequence, the DOA estimation process can be executed independently from the targets' characteristics and the same approach can be used for all possible detections irrespective of their positions, velocities and other parameters.

For a ULA, the steering vector can be derived theoretically by the geometrical relation from (2.21), as long as the directive gain $D_m(\theta)$ of the antenna elements is known. The antenna gain is reflected in the magnitude response $|a_m(\theta)|$ of the steering vector and can be assumed to be equal across the array, if identical antennas are used. Then, the elements of the steering vector differ only in phase and are mainly determined by the antenna position.

$$\begin{aligned} \boldsymbol{a}(\theta) &= \begin{bmatrix} a_1(\theta) & a_2(\theta) & \cdots & a_M(\theta) \end{bmatrix}^T && (2.22) \\ &= D(\theta) \begin{bmatrix} 1 & e^{j2\pi \frac{d}{\lambda} \sin(\theta)} & \dots & e^{j2\pi \frac{(M-1)d}{\lambda} \sin(\theta)} \end{bmatrix}^T && (2.23) \end{aligned}$$

Interestingly, a relative phase shift in the positive direction can be observed for such antenna elements which encounter a longer time of flight. It is a direct cause of the FMCW beat signal equation (2.8), where the phase is positive linear dependent on the distance r. This is in contrast to many other definitions of the steering vector for a ULA, where the phase shift is usually negative for a larger distance to the target [46, 47]. The discrepancy arises, because most references in literature use the phase of the electrical field at the array elements as a basis for the steering vector. However, an FMCW sensor cannot measure the field of the impinging wave directly. Instead it measures the phase difference of the local oscillator and a time delayed version of it, which is finally the cause of the positive sign of the phase term in (2.23).

Array calibration

Rather than using theoretical derived values, the steering vectors can also be measured with a special sensor calibration setup. This has the advantage that any deviation from the ideal array response can be taken into account for the angle estimation process. In practice, the array response can be distorted, for instance by small tolerances in the manufacturing process, by a radome in front of the antennas, by signal coupling between adjacent antenna elements or by component variations over temperature, just to mention a few.

A sensor calibration is usually conducted with the help of a fixed target at a known position. The sensor scans and records the array output for each possible target angle θ. Ideally, the target distance is constant for all measurements so that the free space loss remains constant and has no influence on the result. Furthermore, the calibration target should behave very close to an ideal point target with no directional properties. Often a trihedral corner reflector or a metal sphere are used for this kind of measurement.

In order to illustrate a practical array response, the magnitude and phase values of the steering vectors for a real radar sensor featuring a ULA with 8 elements are shown in Fig. 2.9. The used representation is also referred to as *two-way antenna diagram* and contains meaningful information about the sensor's directional characteristics. The used prototype radar sensor for these measurements has been developed during this thesis' time span and is presented more in detail in chapter 5.

As the name suggests, the two-way antenna diagram includes both, the characteristics of the sending and receiving antennas. The advantage of this calibration method is that it closely resembles the real measurement conditions. The output is the appropriate steering vector which can be directly applied to many angle estimation methods.

The magnitude curves $D_m(\theta)$ in Fig. 2.9a provide information about the directivity of the used antennas. The antenna gain as a function of the target angle can be immediately read out of the diagram. Two important parameters can be identified, the location of the maximum gain which defines the *antenna boresight* and the width of the *main lobe* which basically determines the sensor's *field of view (FoV)*.

Remarkably, the influence of the sine term on the phase of the steering vectors can be observed in the phase curves in Fig. 2.9b, where a smaller gradient for

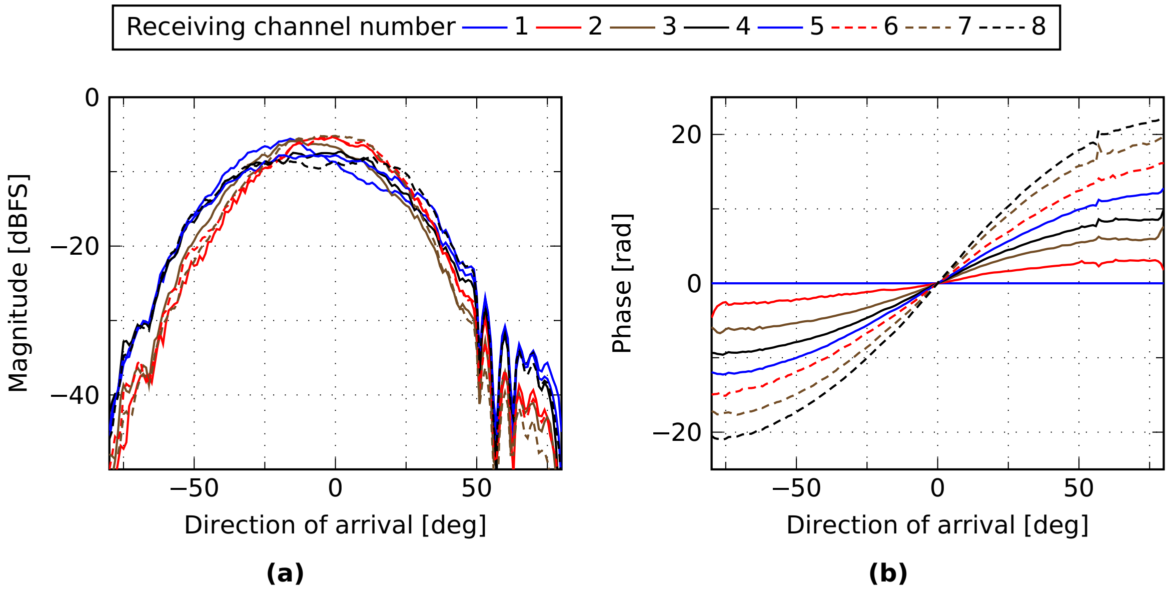


Figure 2.9: Measured two-way antenna diagram of the experimental sensor which was developed along with this thesis (cf. chapter 5). The antenna array has 8 receiving elements spaced with 0.5λ and the measurement was made without a radome.

larger angles θ, i.e. toward ±90 degrees, can be seen. The decreased slope causes steering vectors of a similar angle θ to resemble each other which makes the angle estimation process more difficult. As a consequence, the angular resolution of an array based radar sensor typically deteriorates toward the sides [46, p. 949].

For a more detailed analysis of the phase values, the deviation of the real steering vectors from the theoretically derived can be seen in Fig. 2.10 where only the differences between them are shown. As it can be seen from the plot, the variations in phase are rather large and amount up to 1 rad, or equivalently about 60 degrees. Consequently, the DOA estimation quality will suffer if no calibration or equalization method is applied [52].

2.3.2 Classical beamforming

A very basic approach to estimate the DOA is to directly compare the received signals of each antenna. An inverse time delay based on a hypothetical DOA can be applied to each channel. If this delay matches the actual offset, then the signals will all be in phase again and a subsequent summation would result in

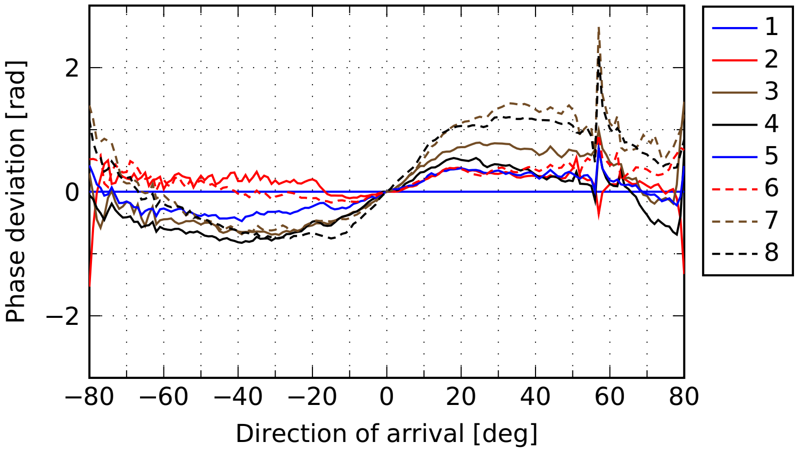


Figure 2.10: Measured phase errors of the real antenna array from Fig. 2.9. The shown numbers are the differences between the actual and theoretical values.

a constructive interference. With the help of an exhaustive search, all possible DOAs can be tested in this manner and the direction θ for which the sum is maximized can be used as a DOA estimate.

The described approach can also be conducted in the frequency domain, after the range and velocity estimation took place. The time delay at the different receiver elements appears as a phase offset[5] which can be easily compensated by a rotation in the complex plane. Besides, the SNR gain resulting from the spectral evaluation improves the estimation quality and beyond that, the probability of a superposition of different signals is also reduced. Objects with a different distance to the sensor will be separated prior to the angle estimation by the two-dimensional FFT (cf. also section 2.1.4). These reasons suggest a DOA estimation after the frequency transformation which is also the choice for all following investigations in this work.

Starting from a detected object in the r-D space, a signal vector $\boldsymbol{x}$ consisting of complex values is extracted. It carries all necessary information and forms the data input for the following DOA estimation along with the steering vector. The number of elements of the signal vector $\boldsymbol{x}$ corresponds to the number of antennas, just as it is the case for the steering vector.

The individual phase values of the signal vector $\boldsymbol{x}$ can now be modified according to a possible DOA. This operation corresponds to an element-wise multiplication with the complex conjugated steering vector $\boldsymbol{a}(\theta)$ for the spe-

---

[5] Narrow band assumption (2.20)

cific direction θ. The complex conjugation assures that the inverse phase delay according to the assumed DOA is applied. Finally, the norm of the resulting vector gives an indication about the degree of correlation. A large value indicates that the individual elements are in phase and sum up constructively. This procedure can be repeated for other possible DOAs and the same signal vector. If an exhaustive search for the entire FoV is performed, the problem to solve can be denoted as:

$$\arg\max_{\theta} \sum_{m=1}^{M} |x_m a_m^*(\theta)|^2 \tag{2.24}$$

Or, in vector notation:

$$\arg\max_{\theta} |\boldsymbol{a}^H(\theta)\boldsymbol{x}|^2 \tag{2.25}$$

This method is often referred to as conventional beamforming or Bartlett beamformer [46,47]. Even though the principle is very simple, the performance is satisfactory for many applications. It can be shown that the conventional beamformer corresponds to the maximum likelihood estimator in the case where only one single signal source exists (cf. (2.34) in the following section). In many situations however, especially in the case of automotive radar scenarios, multiple reflections from different directions will superimpose which makes it more complicated to detect and separate all the different signal sources.

For a better understanding, the angular spectrum of a real measurement is shown in Fig. 2.11. It consists of a scenario with only one object at about -5 degrees, whose reflection is relatively strong so that a high SNR value is achieved. The result shown corresponds to the evaluated term in (2.25), except that the full range for all angles θ is shown and not only the maximum value. Similar to a frequency spectrum, the occurrence of a maximum does not appear as a sharp line, but as a rather wide peak which is called *main lobe*. The width of the main lobe is connected with the size of the used antenna array. Larger aperture sizes decrease the width of the main lobe, which consequently increases angular resolution. A nearby second object can be separated more easily if the main lobe is more narrow.

The remaining local maxima in Fig. 2.11 do not originate from real targets. They are called *side lobes* and belong to the same signal source which induces

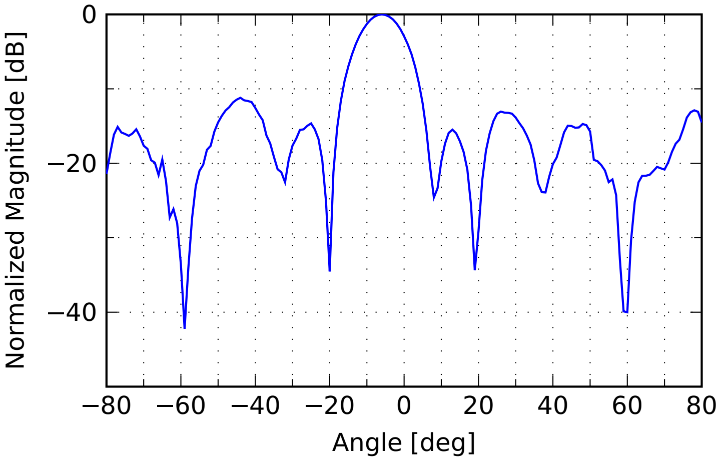


Figure 2.11: Computed output of the conventional beamformer for a single target at about -5 degrees. The residual side lobes are induced by the same object and do not originate from other targets.

the main lobe. Their relative signal power is quite high and lies only about 10dB below the level of the main lobe. A target with a smaller signal echo could get masked easily by the side lobes of a strong reflection. This makes it very difficult for array based sensors to detect multiple signals in the angular domain.

2.3.3 Maximum likelihood (ML) estimator

Maximum likelihood (ML) estimators belong to a special class of DOA algorithms with some important statistical properties. They are efficient in the sense that they achieve the Cramer-Rao lower bound for large sample sizes. In addition, it has been shown that the performance for small sample sizes or even in the single snapshot case is superior to other estimators [53,54]. Nevertheless, the immediate use for real applications has been unacceptable for a long time due to their high computational cost. In most cases, ML estimators need to find the global maximum of a non-linear, multidimensional function. In general, global search procedures are required, because no closed form solution for this optimization problem exists. Iterative search strategies have been reported, however their convergence is highly dependent on the initial value [55].

Motivation for ML estimators

A major drawback for high-resolution DOA estimation in the case of automotive radars is the number of snapshots, i.e. the number of independent measurement cycles, which is generally small due to the non-stationarity of the environment. Especially when the data is subject to a range-Doppler preprocessing, like it is the case for many radar applications, only a single snapshot is available. This impedes the usage of many DOA methods and justifies the usage of computational demanding maximum likelihood (ML) estimators.

In general, the complexity grows with the number of dimensions, i.e. the number of signal sources, which lead to the development of approximate methods. The advantage of many algorithms is that they only require a 1D search, for instance the minimum-variance (Capon [56]) or MUSIC estimator [57]. Another class of estimators exploit the regularity of a ULA whose antenna elements are all equally spaced. ESPRIT [58] or Root-MUSIC [59] are prominent examples which exploit such properties of the array design. A lot of those algorithms have in common that they require the number of snapshots to be greater than the number of signal sources. In addition, if the incident signals are correlated, the performance degrades heavily and in the case of fully coherent signals the estimation may even completely fail. In contrast, the performance of ML methods is still satisfactory, which makes them essential for certain applications. Furthermore, they still work if only a single snapshot is available. For this reason, ML methods are very attractive for the usage in automotive radars, which made them subject to research [54, 60]. More background information about other DOA estimation methods can be found in [46, 47, 61].

In the remaining part of this section, the general ML estimator is simplified for the most frequent cases: The estimation of the DOA of a single target or the estimation of two DOAs belonging to two targets. Even though the total number of objects in the vicinity of an automotive radar sensor is high, the angle estimation is conducted separately for each detected target in the r-v-space. It is recalled, that many different echoes have already been isolated by the preceding range-Doppler preprocessing step (cf. section 2.1.4). Thus, only in very few cases more than a single target is remaining and if so, the two target estimation is usually sufficient.

One aspect when performing a parametric DOA estimation is the determination of the model order, i.e. the number of present signals whose parameters have to be estimated. In fact the problem can also be described by detecting the number of target echoes in a noise environment. This topic is out of scope of this thesis and is not specified more in detail. Nevertheless, solutions to this problem exist which are often referred to as model order selection [62, 63].

Derivation of single and two target ML estimator

The signal models often distinguish two different variants of ML estimation. The first one assumes a deterministic signal, which is often present in communication or active radar applications where the waveform is known. The second one adapts to a non-deterministic signal which has not been identified yet, e.g. when passively locating an unknown sender. This leads to two different results of which the first is known as conditional ML estimation (CMLE) or deterministic ML (DML). The second one is referred to as unconditional ML estimation (UMLE) or statistical ML (SML). The elaboration and results presented in this thesis are entirely based on DML estimation, because the received signals result from echoes of the transmitted and hence known waveform.

The following derivation is based on previous work from [46, 47, 60, 61], though it has been specifically adapted in this thesis. The used signal model assumes that K complex signals s_k impinge on the antenna array comprising M elements. The signals can be integrated into the signal vector $\boldsymbol{s} = [s_1 \quad s_2 \quad \ldots \quad s_K]^T$ and every signal s_k can arrive from a different direction θ_k. In this case, the array output at each of the M elements is a superposition of the K signals. Finally, a noise vector $\boldsymbol{n}$, whose elements are all independently complex Gaussian distributed (i.i.d.), is added to form the measured array output $\boldsymbol{x}$, which is a $M \times 1$ vector:

$$\boldsymbol{x} = \boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{s} + \boldsymbol{n} \tag{2.26}$$

The matrix $\boldsymbol{A}(\boldsymbol{\theta})$ consists of the steering vectors $\boldsymbol{a}(\theta_1)$, $\boldsymbol{a}(\theta_2)$, $\ldots$, $\boldsymbol{a}(\theta_K)$ which relate to the signal vector $\boldsymbol{s}$. The problem to solve consists of two steps: The number of sources K has to be estimated. Afterwards, an incident angle θ_k has to be found for each source s_k. As already mentioned above, the number of sources is assumed to be known for the following derivation in this work.

The DML function $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{s})$ for the signal model of (2.26) in the case of a single snapshot is given by the following expression [46, p. 1005]:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{s}) = \frac{1}{\det(\pi\sigma^2 \boldsymbol{I})} \exp(-\frac{1}{\sigma^2} \|\boldsymbol{x} - \boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{s}\|^2) \tag{2.27}$$

where σ^2 is the variance of the Gaussian noise and $\|\cdot\|$ denotes the Euclidean norm. The same noise model as in section 2.2.2 has been used (cf. also Fig. 2.5).

Applying the natural logarithm function results in the log-likelihood function, which becomes maximal for the same argument values of $\boldsymbol{\theta}$, however its computation is simplified.

$$\ln \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{s}) = -M \ln \pi - M \ln \sigma^2 - \frac{1}{\sigma^2} \|\boldsymbol{x} - \boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{s}\|^2 \tag{2.28}$$

The goal is to maximize this function in order to find the ML estimator $\hat{\boldsymbol{\theta}}$. This can be achieved by fixing all but one of the parameters and forming the partial derivation with respect to the arbitrary parameter. For the reason of brevity just the solution is given, whereas a complete derivation can be found in [46, pp. 1004-1007].

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left\| \boldsymbol{x} - \boldsymbol{A}(\boldsymbol{A}^H \boldsymbol{A})^{-1} \boldsymbol{A}^H \boldsymbol{x} \right\|^2 \tag{2.29}$$

The dependency of $\boldsymbol{A}$ on the parameter $\boldsymbol{\theta}$ has been omitted to simplify the expression above. Furthermore, the notation $\boldsymbol{A}^H$ is used for the *conjugate transpose*[6] of the matrix $\boldsymbol{A}$, and $\boldsymbol{A}^{-1}$ is used for the *inverse* matrix of $\boldsymbol{A}$.

The minimization problem from (2.29) can be rewritten as follows:

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= \arg\min_{\boldsymbol{\theta}} \left[\boldsymbol{x} - \boldsymbol{A}(\boldsymbol{A}^H \boldsymbol{A})^{-1} \boldsymbol{A}^H \boldsymbol{x} \right]^H \left[\boldsymbol{x} - \boldsymbol{A}(\boldsymbol{A}^H \boldsymbol{A})^{-1} \boldsymbol{A}^H \boldsymbol{x} \right] && (2.30) \\
&= \arg\min_{\boldsymbol{\theta}} \left[\boldsymbol{x}^H \boldsymbol{x} - \boldsymbol{x}^H \boldsymbol{A}(\boldsymbol{A}^H \boldsymbol{A})^{-1} \boldsymbol{A}^H \boldsymbol{x} \right] && (2.31) \\
&= \arg\max_{\boldsymbol{\theta}} \left[\boldsymbol{x}^H \boldsymbol{A}(\boldsymbol{A}^H \boldsymbol{A})^{-1} \boldsymbol{A}^H \boldsymbol{x} \right] && (2.32)
\end{aligned}$$

In general, no closed form solution exists for this expression, which is why

---

[6] The conjugate transpose of a matrix, also referred to as Hermitian transpose, can be obtained by a complex conjugation of every entry followed by a matrix transposition.

an exhaustive maximum search has to be performed. In [53], an approximate estimation method for two closely spaced targets was developed, which is however only valid if the targets are known to differ only in a small angle $\Delta\theta$. For this reason, this approximation cannot be used since the values of $\Delta\theta$ are not constrained and can take arbitrary values in the case of automotive radar sensors.

For a single target, the matrix $\boldsymbol{A}$ consists only of a single vector $\boldsymbol{a}(\theta)$, denoted as $\boldsymbol{a}$ for convenience. Furthermore, if the norm of the steering vector $\boldsymbol{a}$ is known to be equal to one (which is no restriction in general), the computation of the term $\boldsymbol{a}^H\boldsymbol{a}$ gets trivial and (2.32) simplifies to the following expression:

$$\hat{\theta} = \arg\max_{\theta} \left[\boldsymbol{x}^H\boldsymbol{a}\underbrace{\left(\boldsymbol{a}^H\boldsymbol{a}\right)^{-1}}_{=1}\boldsymbol{a}^H\boldsymbol{x}\right] = \arg\max_{\theta}\left[\boldsymbol{x}^H\boldsymbol{a}\boldsymbol{a}^H\boldsymbol{x}\right] = \tag{2.33}$$

$$= \arg\max_{\theta} |r_\theta|^2 \quad \text{where } \; r_\theta = \boldsymbol{x}^H\boldsymbol{a} \tag{2.34}$$

In the equation above, the scalar product between the measured vector $\boldsymbol{x}$ and the array steering vector $\boldsymbol{a}(\theta)$ is denoted as r_θ for simplicity.

For two targets, the matrix $\boldsymbol{A}$ contains two steering vectors $\boldsymbol{a}(\theta_i)$ and $\boldsymbol{a}(\theta_j)$, which will be denoted as $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$. In order to obtain the ML estimator, the inverse matrix of $\boldsymbol{A}^H\boldsymbol{A}$ has to be computed. This can be done by using the simple analytic solution for a 2x2 matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{2.35}$$

Consequently, for the two target estimator, the matrix $\boldsymbol{A}^H\boldsymbol{A}$ and its inverse can be written as follows:

$$\boldsymbol{A}^H\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_i^H \\ \boldsymbol{a}_j^H \end{bmatrix}\begin{bmatrix} \boldsymbol{a}_i & \boldsymbol{a}_j \end{bmatrix} = \begin{bmatrix} |\boldsymbol{a}_i|^2 & \boldsymbol{a}_i^H\boldsymbol{a}_j \\ \boldsymbol{a}_j^H\boldsymbol{a}_i & |\boldsymbol{a}_j|^2 \end{bmatrix} = \begin{bmatrix} 1 & \beta_{ij} \\ \beta_{ij}^* & 1 \end{bmatrix} \tag{2.36}$$

$$\left(\boldsymbol{A}^H\boldsymbol{A}\right)^{-1} = \frac{1}{1-|\beta_{ij}|^2}\begin{bmatrix} 1 & -\beta_{ij} \\ -\beta_{ij}^* & 1 \end{bmatrix} \tag{2.37}$$

$$\text{where} \quad \beta_{ij} = \boldsymbol{a}_i^H\boldsymbol{a}_j \tag{2.38}$$

Combining (2.32) and (2.37) results in a maximization problem for the two target case:

$$\begin{bmatrix}\hat{\theta}_i & \hat{\theta}_j\end{bmatrix} = \underset{\theta_i,\theta_j}{\arg\max}\,\frac{1}{1-|\beta_{ij}|^2}\begin{bmatrix}r_i & r_j\end{bmatrix}\begin{bmatrix}1 & -\beta_{ij}\\ -\beta_{ij}^* & 1\end{bmatrix}\begin{bmatrix}r_i^*\\ r_j^*\end{bmatrix} \tag{2.39}$$

$$\text{where}\quad r_i = \boldsymbol{x}^H\boldsymbol{a}_i\,,\quad r_j = \boldsymbol{x}^H\boldsymbol{a}_j\,,\quad \beta_{ij} = \boldsymbol{a}_i^H\boldsymbol{a}_j \tag{2.40}$$

With some simplifications, this leads to the following expression for the DML estimator in the case of two targets [60, p. 86]:

$$\begin{bmatrix}\hat{\theta}_i & \hat{\theta}_j\end{bmatrix} = \underset{\theta_i,\theta_j}{\arg\max}\,\frac{|r_i|^2+|r_j|^2-2\,\mathrm{Re}\left(r_i r_j^*\beta_{ij}\right)}{1-|\beta_{ij}|^2} \tag{2.41}$$

where the operator $\mathrm{Re}\,(\,\cdot\,)$ represents the real part of a complex number. Clearly, the complexity of computing one value of the ML function is more than three times as expensive as in the single target case, because three scalar products (r_i, r_j and β_{ij}) have to be computed. In addition, the number of possible angle values is squared, which is even more severe in the case of a full grid search. This problem is addressed in section 3.3.2 of this work, where an implementation of the two target DML estimator incorporating several optimizations is presented which makes a real-time computation nevertheless possible.

2.3.4 Multiple-input multiple-output (MIMO)

So far, the described array processing techniques have been applied solely on the receiving array. However, the concept of an incident plane wave which excites a phase shift at the individual receiving elements can also be applied on the transmitter side. An additional propagation delay likewise occurs if an array of multiple transmitting antennas is used. In such a setup, a mechanism to distinguish the superimposing signals at the receiving side is required which will be outlined in the following paragraphs.

This concept of utilizing multiple transmitting (Tx) and receiving (Rx) antennas at the same time is also known as multiple-input multiple-output (MIMO) technique. It is commonly used by state-of-the art wireless communication standards to increase the link capacity due to the benefit of spatial diversity. In a similar fashion, this method can also help to increase the angular resolution capabilities of a radar sensor while reducing overall hardware and component costs.

A conventional receiving array consists of multiple antennas, each connected to its own dedicated processing channel. At least, a dedicated mixer device, some kind of baseband signal amplification and filtering circuit, plus an analog-to-digital converter (ADC) is required for an FMCW radar, which adds to the overall system cost. With increasing angular resolution and separation requirements, a trend toward larger apertures and increasing channel numbers can be observed. By using multiple transmitters, some hardware effort from the receiving side can be shifted to the transmitting side and the overall efficiency can be increased. This shall be illustrated by the following example.

If an array geometry consisting of 16 channels is desired for a certain application, the conventional approach would be to employ 16 dedicated receiving channels. With the help of a MIMO system, several channels could be economized on the receiving side at the cost of additional transmitter channels. For instance a 2 Tx and 8 Rx system achieves also 16 virtual channels at the cost of only one additional transmitter while half of the receivers can be omitted. The system could be further optimized and transformed into a 4 Tx – 4 Rx system also resulting in 16 virtual channels. In any case, the cost of additional transmitters has to be compensated by the cost of the economized receiver elements. Furthermore, specific constraints can limit the number of transmitters, i.e. the possibility to achieve a separation of a large number of transmitters.

Virtual channels

In order to understand the more complex phase relationships which occur at a MIMO antenna array, the concept of introducing *virtual antennas* turns out to be helpful. Any combination and arrangement of sending and receiving antennas can be transformed into a virtual array representation, which contains all possible Tx – Rx pairs. Therefore, the number of elements in the virtual array is obtained by multiplying the number of sending antennas with the number of receiving antennas. This is the key to the hardware efficiency of a MIMO array, because a 4 Tx – 4 Rx array only consists of 8 physical antennas, while the virtual array possesses $4 \times 4 = 16$ elements.

A geometric relation likewise exists between the virtual antennas, which is

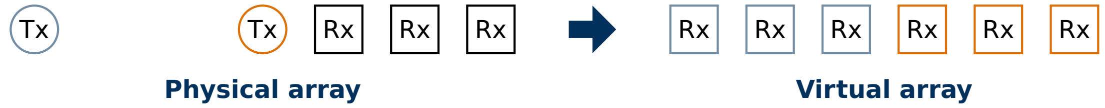


Figure 2.12: A physical MIMO array can be transformed into its virtual representation by a convolution operation. The color marks the corresponding transmit antenna for the virtual channels.

crucial for the performance of the MIMO array. It can be obtained by a binary convolution of the transmitting and the receiving array. A basic example is shown in Fig. 2.12 where 2 Tx and 3 Rx form a 6-element virtual ULA. For a detailed derivation of the theory behind virtual arrays refer to [64, 65].

The main advantage of the virtual representation is that all DOA methods which have been described so far can be applied to it, even though the physical array is built with a MIMO concept. Special attention has to be paid to the transmitter separation, because the sent signals have to be isolated precisely in order to supply the required information for the DOA algorithms. Existing techniques will be introduced in the next section.

Transmitter separation

The problem of separating the different transmitted signals at the receiver side is common for all kinds of MIMO radar systems. In fact, the phase and frequency information of the different transmitters superimposes and additional measures are required to extract the individual signals again.

In the context of digital communication systems, a similar problem is the usage of a single communication channel by many participants. This can be achieved by using an appropriate multiplexing technique which ensures a correct combination of the different signals so that an inverse demultiplexing process becomes possible at the receiving side.

For a MIMO radar system, the prevailing multiplexing techniques are more or less identical to those used in telecommunications. The most relevant methods are mentioned here and described briefly in the following list. For a better understanding, they are further illustrated in a schematic representation in Fig. 2.13.

Time-division multiplexing (TDM) A straight forward approach is to activate only one transmitter at a certain instant of time. Then, the signal at the receiving side is known to originate from a certain Tx antenna and the virtual channels can be assigned unambiguously. Even though the measurement time has to be shared by the individual Tx channels, this approach is the most commonly used because it can be realized very easily with a low hardware complexity.

Frequency-division multiplexing (FDM) Another obvious approach is the simultaneous use of multiple frequency bands for the different transmit antennas. The frequency offset between two TX can be chosen rather small compared to the overall used bandwidth B. The minimum offset is bounded by the required intermediate frequency (IF) bandwidth of a single Tx. Then, the resulting total IF bandwidth at the receiver side is the sum of multiple transmitters' IF bands and optional guard bands which imposes quite high requirements to the IF hardware. An example implementation for such an FDM system was reported in [66].

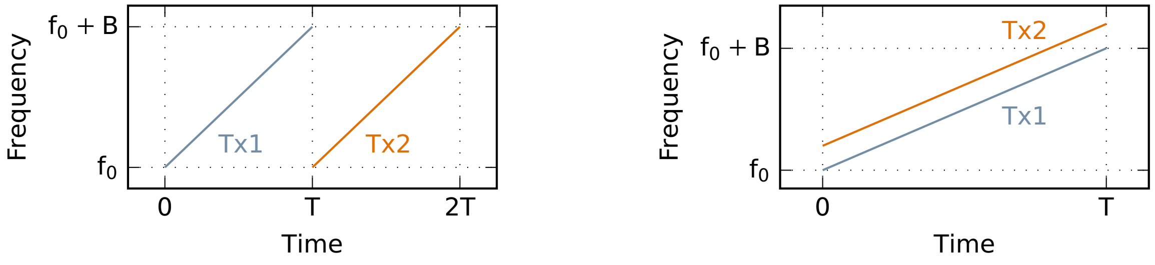
Another possibility would be the employment of multiple mixer devices along with dedicated ADCs for each virtual channel. This would help to reduce IF bandwidth at the cost of a higher component count.

Code-division multiplexing (CDM) A potential way to solve the separation problem in the digital domain is to employ a distinct code for each signal which can be recognized again in the received signals. A possibility with low hardware effort would be the usage of a binary phase shift modulation (BPSK) with optimized code sequences [67].

In the rest of this thesis, only the TDM-MIMO technique is further used and described, but nevertheless many algorithms and architectures can be used without modification by other methods like FDM. In fact, the full steps of range-Doppler processing, CFAR processing, target detection as well as DOA estimation can be performed independently from the used modulation technique. Only some kind of demultiplexer has to be used first which isolates the virtual channels present in the physical receiving channel.

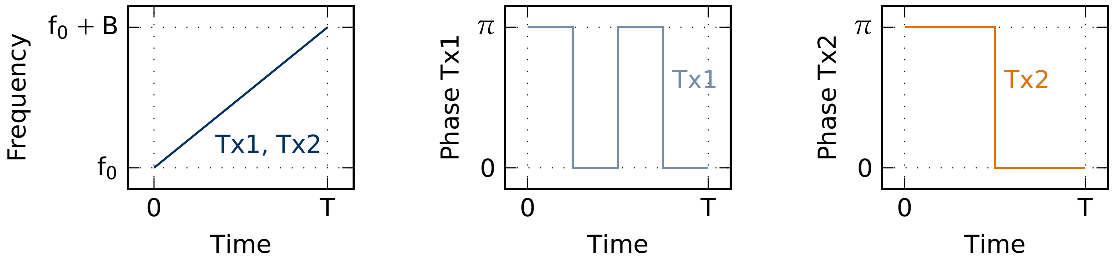
Motion compensation

A drawback of the TDM-method in a MIMO radar environment is the occurrence of the Doppler shift for moving targets. Due to the non-stationarity of



(a) Time-division multiplexing (TDM)

(b) Frequency-division multiplexing (FDM)



(c) Code-division multiplexing (CDM)

Figure 2.13: Different MIMO multiplexing techniques used in the context of an FMCW radar system with two transmitters.

certain objects, the measured phase will change depending on the instant of time. Therefore, if different transmitters are triggered sequentially, a phase shift is introduced depending on the relative velocity. This phase shift has to be compensated before any DOA estimation method is applied because the phase relationship between all virtual channels has to be solely provoked by the DOA.

Various techniques for a motion compensation exist, e.g. the employment of overlapping virtual channels [68]. Another possibility is the accurate estimation of the velocity prior to the DOA estimation and the straightforward compensation of the expected Doppler shift [69]. This method is also used in this thesis.

2.4 State of the art

For nearly all driver assistance systems, a reliable sensor system is required in order to provide the necessary information about the environment. Automotive radar sensors are one big pillar of current ADAS[7] applications, due to their

---

[7] Advanced Driver Assistance Systems

robust and accurate distance and velocity measurement capability. This section provides a brief overview of the most recent trends and developments regarding commercially available radar sensors for automotive applications.

A steady growth in the number of equipped cars and thus in the number of produced radar sensors can be observed in the last years. For instance Bosch started with a series production of automotive radar sensors back in 2000 and sold one million sensors until 2013 [70]. In the following years until 2016, in total more than 10 million sensors were sold and a further increase is still expected [71,72].

On the technology side, some major changes were conducted compared to the first generation sensors. In the beginning, the used semiconductor material for the high-frequency components was mainly based on gallium arsenide (GaAs), while it was more and more replaced by silicon-germanium (SiGe) later [73]. Current research focuses on the use of pure silicon CMOS[8] which would provide a further cost reduction and a better integration capability [74]. Beyond that, a combination of antennas, analog front-end and digital signal processing blocks onto a single chip may be within reach.

Nowadays, more and more radar sensors utilize the 76 – 77 GHz frequency band which is allocated for automotive applications and is approved for worldwide use. Besides, a second frequency range from 77– 81 GHz is available and offers the possibility for a bandwidth extension even though it is currently unused [3, p. 327]. This frequency band above 77 GHz is still not approved for vehicular radar use in some countries. However, the FCC (Federal Communications Commission) has opened up the full frequency range from 76 to 81 GHz in the US recently [75], so that a use of the full spectrum by automotive radar applications seems to move closer. Consequently, wideband applications incorporating a total bandwidth of up to 5 GHz may become possible.

Some midrange sensors also operate in the 24 GHz frequency band like for example the Continental SRR 200 or the 24 GHz Radar sensor from Hella. The advantage of the lower frequency are mainly reduced component costs at the disadvantage of a larger sensor size due to the increased wavelength [73].

From a modulation point of view, a predomination of FMCW-based sensors can be clearly observed. Especially the chirp sequence modulation which is said to “achieve the best possible utilization of the signal power, bandwidth,

---

[8] Complementary Metal-Oxide-Semiconductor

and measuring time" [3, p. 354] has been successfully brought to the market recently. This modulation technique allows a significant performance gain and is increasingly employed, despite its higher requirements on the signal processing unit. Basically, the raw data rates grow significantly due to the usage of fast frequency chirps which induce higher baseband frequencies [31]. Furthermore, current sensors only use a fraction of the available bandwidth. An increase of this bandwidth would result in even higher baseband frequencies which is a predictable scenario.

The higher data rates which are mainly a result of the faster analog-to-digital (A/D) sampling rates in the case of fast chirp modulations have to be handled under strict real-time constraints. Specialized signal processors have already been developed for this purpose and dedicated radar co-processors which have been integrated into automotive microprocessors are available. At the moment, their acceleration capability is limited to very basic tasks, however, a further increase in functionality is expected in the near future.

At this point in time, the following three products targeting the market of signal processors for radar-based ADAS can be identified. Although their additional functionality is confined on FFT accelerators and ADCs only, a trend toward integrated and more specialized radar system building blocks is recognizable.

NXP MPC577xK-MCU An automotive microcontroller offering a dedicated accelerator block which is able to compute a fixed-point FFT. An analog front-end including ADCs is also integrated which directly connects to the FFT accelerator block. [76,77]

Infineon Aurix TC297TA Similar to the NXP signal processor, this automotive microcontroller incorporates an FFT accelerator unit as well as ADCs. These blocks connect to the bus system of the processor which enables a flexible data flow. [78]

Texas Instruments AWR1642 A complete radar system on chip (SoC) which combines the analog circuits for the signal and ramp generation with the digital parts like A/D conversion and DSP. It is manufactured in a radio frequency (RF) capable CMOS process. [79,80]

A further increase in computational power as well as a larger number of units will lead to the development of more powerful application-specific integrated circuits (ASICs) for radar in the near future. The basic signal processing tasks can be progressively integrated as a hardware accelerator in order to attain a maximum of performance and efficiency. A case study was conducted during the development of this thesis which indicated that the expectable data rates might exceed 10 GBit/s and more if the resolution and accuracy of the sensors is further increased [36]. The raw data volume of a single measurement snapshot can amount to more than 100 MByte which cannot be handled reasonably by a general purpose automotive microprocessor.

The already available accelerator blocks of current radar DSPs show the beginning trend of higher integrated and more specialized radar ASICs. The subsequent signal processing tasks like noise level estimation, target detection and angle estimation also have quite high requirements in terms of computational load, so that an acceleration of these steps would lead to a more powerful and efficient processing unit. At the same time, these algorithms consist usually of a very regular schedule so that they offer a high potential for parallelization and savings of computation time.

As a practical example, a recent study [81] investigates the integration of a target detection accelerator into the specialized radar DSP *Infineon TC29X* which has also been mentioned above. The conducted experiments show an execution time speedup factor of 700 in comparison to the general purpose central processing unit (CPU) which is integrated on the same chip.

2.4.1 Related work

Several isolated investigations and implementations of hardware accelerators for automotive radar signal processing tasks have been reported in literature, but a systematic design space exploration and an integration of the different blocks has not been realized so far. Often, the intended applications are slightly different and hence the implementations are not optimized for a use within an automotive radar sensor employing a chirp sequence modulation technique. The following paragraph lists some known examples from literature along with their covered use-case. The examples are sorted by their covered algorithmic sub-step, namely FFT, CFAR and DOA.

Langemeyer, 2011 [82] An architecture optimized for the processing of large two-dimensional FFTs on an FPGA. The implementation is real-time capable and can be used on a moving platform for SAR processing. A sophisticated addressing scheme for dynamic random-access memory (DRAM) is employed in order to increase the memory throughput.

Winkler et. al., 2004 [83] An automotive radar system which uses an FPGA for signal processing. The radar system uses a pulse modulation scheme and operates at 24 GHz. A 256 point FFT which is based on a decimation-in-time (DIT) Radix-2 butterfly as well as an OS-CFAR procedure is calculated on the FPGA.

Bales et. al., 2012 [84] A comparison of OS-CFAR implementations on different architectures including CPU, GPU and FPGA. The computation is based on single-precision floating-point values and includes different architectural variants. A focus is set on real-time capability and large window sizes up to 256 cells.

Magaz et. al., 2008 [85] An optimized implementation of the OS-CFAR on an FPGA is presented. The sorting is stopped after the desired value has been found which helps to save resources. Some figures of the resource usage are presented for a rather small window size of 16 cells.

Perez-Andrade et. al., 2010 [86] An OS-CFAR processor which is based on a full sorting of all window cells is presented. The architecture is based on an insertion sort approach which exploits the sliding window nature of a CFAR processing. A list of already sorted window cells is kept in memory while only one additional cell is inserted and removed per time interval. Different window sizes up to 64 cells were evaluated.

Seguin et. al., 2011 [87] A digital beamforming system for weather tracking applications was developed and implemented on an FPGA. A conventional beamforming approach based on phase shifts is used. The system is capable to process 64 I/Q channels at a maximum sampling rate of 27 MSPS. Multiple beams can be computed in parallel so that a real-time processing of all samples is possible. No target detection prior to the beamforming is conducted so that the requirements on the beamforming data throughput are quite high.

Winterstein et. al., 2012 [88] A digital signal processor for a phased array radar demonstrator is presented employing conventional beamforming.

The intended application is a ground-based space surveillance radar, though the described system is only a functional small-scale demonstrator. It uses 16 channels operating at a sampling rate of 50 MSPS. The beamformer output consists of 256 beams which are computed for each FFT cell.

Walke et. al., 1999 [89] An architecture suitable for the calculation of adaptive weights in real-time is presented. The weights for the subsequent beamforming operation are approximated with the help of a QR decomposition. This technique is an important preprocessing step if an adaptive filtering of the input signals is required. The QR decomposition relies on rotations in the complex plane which are realized with the help of Cordic processors.

From the list above it can be observed that a variety of radar applications rely on optimized processing architectures in order to handle high data rates and a large number of parallel channels. The real-time constraints, however, are often less strict as it is the case for automotive radars. On the other side, certain applications have a higher throughput demand on the DOA estimation step which is performed on the full raw data input. The requirements can be lowered significantly if the detection takes places before this step. Such discrepancies make it difficult to directly apply and reuse previous results to automotive applications using current state-of-the-art chirp sequence modulation techniques. It is thus indispensable to review existing approaches in the context of changing conditions and also to consider the big picture of the whole system.

Different approaches and realizations are investigated in this work. The architectures which have been developed and optimized during this thesis are presented in chapter 3. They cover the most essential radar signal processing chain starting from a digitized time signal and arriving at a detection list which includes, amongst others, the positional coordinates of all measured target echoes. The underlying algorithms correspond to the ones presented in this section, namely spectrum evaluation by an FFT, target detection including CFAR and NCI as well as DOA estimation with the help of ML estimators. A strong focus is set on the efficiency when being used with a chirp sequence based FMCW modulation technique.

Besides, a comprehensive architecture for a radar pre-processing unit has been built up which incorporates all the required modules. They are integrated in a stream-based architecture and their data throughput capabilities are adopted to each other. This allows the usage of a continuous data processing pipeline which avoids unnecessary memory accesses because the data can be directly transferred to the next module. The whole architecture is further verified with the help of an experimental high resolution radar prototype (cf. chapter 5).

Finally, the developed modules are investigated with the help of a model-based design space exploration in chapter 4. Several important characteristics can be seen in the results which help designers to partition and optimize a future radar system.

3

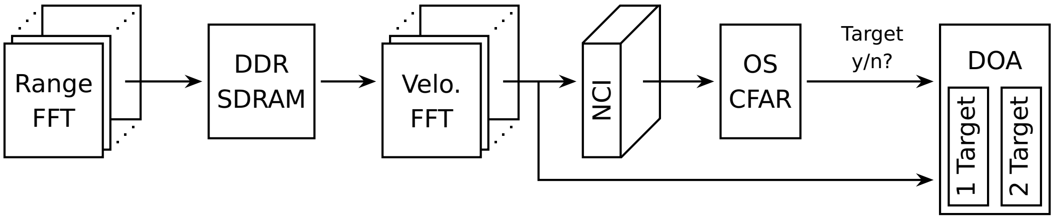
Architectures for Real-time Signal Processing

In the previous chapter, the theory behind automotive radar techniques and corresponding algorithms was derived and explained. A focus was set on the application of FMCW-based automotive radar sensors, using a MIMO antenna array and a chirp sequence modulation. The advantages of certain algorithms and their choice was motivated.

In this chapter, the efficient realization of the required signal processing steps on dedicated hardware architectures is investigated, like for instance on FPGAs or ASICs. The implementation along with architectural choices and optimizations is described, specific properties are identified, numeric effects related to fixed-point arithmetic are investigated and concepts for an efficient DRAM utilization are proposed.

The chapter is organized in accordance with the signal processing chain which is shown in Fig. 3.1. It starts with the spectrum evaluation of the raw time signals which is realized as a two-dimensional FFT and described in section 3.1. The following target detection step that incorporates a CFAR processing is then presented in section 3.2. Finally, the angle estimation leads to a detection list which contains all recognized objects along with their positional coordinates. The realization of the required DOA algorithms on dedicated hardware is explained in section 3.3.

Except the reuse and integration of existing pipelined FFT implementations, all other modules were designed and built from scratch in this work. The design was made at register-transfer level (RTL) so that a very good efficiency



Step	2D-FFT	Target detection	DOA
Algorithm	2.1.4	2.2	2.3
Implementation	3.1	3.2	3.3

Figure 3.1: Overview of the signal processing chain with references to the respective sections.

of the resulting implementations was achieved. The standard language VHDL[1] was used which makes it possible to map the individual modules to different hardware targets. For this work, the hardware descriptions were successfully mapped to a Xilinx Virtex 7 FPGA, though an adaptation to other vendors and technologies should be possible.

The systematic investigation of the resource usage, power consumption and performance characteristics is included in chapter 4. As mentioned above, all modules were implemented on an FPGA and a model-based design space exploration (DSE) was conducted. The obtained model functions along with a graphical representation of the results are shown for each algorithm. Furthermore, a basic comparison to a CPU-based realization is made.

Finally, a functional verification of the whole signal processing chain under real world conditions inside a prototype vehicle is presented in chapter 5.

Parts of this chapter were also published in [26,27].

3.1 Two-dimensional fast Fourier transform (FFT)

The very first processing step shown in the block diagram in Fig. 3.1 is the two-dimensional DFT according to (3.1). The data samples $s_{n,k}$ are indexed by n for the consecutive ADC samples and k for the frequency chirps. A common way to perform this calculation is to execute the twofold sum in two separate steps. First, the inner sum is calculated which only depends on the index n.

---

[1] Very High Speed Integrated Circuit Hardware Description Language

This step corresponds to a one-dimensional DFT and, very importantly, can be applied to each frequency chirp independently. The index k which stands for the current frequency ramp remains constant while the DFT over the N samples is computed. The results of this first stage DFT have to be stored until the last ramp has been processed. Once all results of the inner sum are available, the outer sum can be performed which is again a one-dimensional DFT, however this time operating along the index k.

$$S_{u,w} = \sum_{k=0}^{K-1} \sum_{n=0}^{N-1} s_{n,k} \cdot \mathrm{e}^{-2\pi \mathrm{i} \frac{nu}{N}} \cdot \mathrm{e}^{-2\pi \mathrm{i} \frac{kw}{K}} \tag{3.1}$$

The most obvious way to map this algorithm to an efficient implementation is to use two conventional FFT processors which use a shared memory to store the intermediate results. This memory has to be large enough to store and transpose the full data matrix. For future high resolution radar sensors, the data amount of a single measurement cycle can amount to 100 MByte and more [36]. Consequently, for a growing number of samples and ramps, only external DRAM can provide enough capacity at a reasonable cost whereas an on-chip storage is hardly feasible. The particular properties of a DRAM have to be considered in order to achieve an adequate read and write speed, which is described in section 3.1.3.

In terms of data throughput, the requirements on the 2D-FFT are the highest among all processing steps, because the full raw data of all receiving channels has to be handled by this block. In the worst case, the radar system uses a modulation with a 100 % duty cycle, i.e. there are no gaps between individual chirps. Then, the data rate can be derived from the ADC sampling rate f_s, the word length b and the number of receiving channels M.

For the experimental system presented in chapter 5, the maximum data rate amounts to $M \cdot b \cdot f_s = 16 \cdot 14\,\mathrm{bit} \cdot 250\,\mathrm{MHz} = 56\,\mathrm{GBit/s}$. In this case, a conventional bus topology where all processing elements are connected via a shared datapath to the system memory would shortly operate at its limits. Furthermore, the memory throughput would become a limiting factor if the modules always load their input data from and store their results to the memory. A direct handover of the data between the modules along the processing chain, without a transfer to the system memory, was favored instead. Hence, only fully pipelined architectures for the realization of a streaming 2D-FFT accelerator were considered in section 3.1.1. In this context, streaming means that the accelerator possesses enough built-in buffer memory to store all intermediate results, so that no further interactions with the system memory are required.

An important design parameter is the word length of the data path. It determines essentially the numeric accuracy and it has a strong influence on the resource usage. The challenge for the system designer is to find a minimal word length which fulfills all requirements of the application. In order to achieve this, the origins of quantization noise during the 2D-FFT processing should be well understood. A noise model for the used FFT implementation with fixed-point arithmetic is presented in section 3.1.2 along with specific optimizations explicitly related to the two-dimensionality of the FFT.

Lastly, an appropriate window function is applied to the input data before the actual FFT processing. Its realization is described briefly in section 3.1.4 for the sake of completeness.

3.1.1 Pipelined FFT implementations

For streaming applications, pipelined FFT architectures provide a very high throughput. They are especially useful for real-time applications with a constant workload, where a high degree of capacity utilization can be achieved.

In this work, only the choice of an appropriate architecture is motivated, however in-depth investigations and optimizations of the FFT block itself have not been performed. The theory behind streaming FFT accelerators is well-established and the most relevant topics for a practical use are just mentioned and referenced in this section.

Decimation-in-frequency (DIF) algorithm

The main principle of every FFT realization is to reuse the results of previous computations so that certain operations can be economized compared to a brute-force implementation of the DFT. The first and also the most prominent FFT algorithm is described by Cooley and Tukey [90] and uses a Radix-2 decomposition. The basic idea is to split the full DFT into two halves which can be processed independently and to put the individual results together in the end. The two parts can be further split and so on, until a trivial transform of the length two remains.

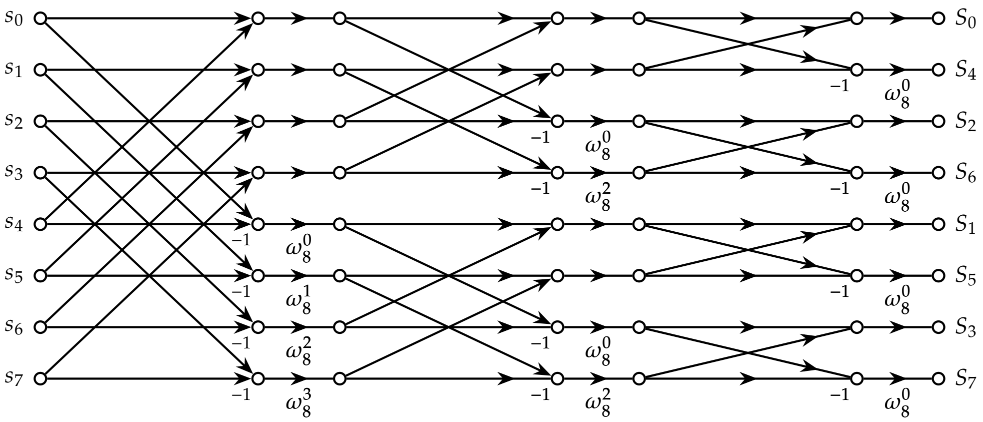


Figure 3.2: Data flow diagram of an 8-point DIF FFT. Three butterfly stages with a different degree of interleaved edges can be identified.

The described approach is also referred to as decimation-in-time (DIT), because the decomposition is done in the time domain and the shorter result vectors are put together in the end. In contrast, the decomposition can also be realized in the inverse direction which is known as decimation-in-frequency (DIF). In this case, the full length data is processed in such a way that the two halves of the intermediate result can be handled completely independent after the first decomposition. The flow graph of the DIF algorithm is shown in Fig. 3.2 for an eight point FFT.

The DIF algorithm can accept the input data in its natural order, which can be advantageous for certain applications. For this work only the DIF decomposition is further considered, simply because a reordering of the ADC input data stream can be avoided.

Radix-2 butterfly

As it was mentioned in the previous section, a longer transform length is realized by recursively splitting it into two halves. The decomposition level within the transform is often referred to as *stage*. The total number of stages N_S for a N-point FFT can be easily expressed by $N_S = \log_2 N$. At each stage, the intermediate results of the previous stage are combined in an interleaved way.

A transform of the length two is also called Radix-2 butterfly, due to the crossing edges in its graph representation. Two input samples are either added or subtracted in order to form two output samples. Furthermore, one of the

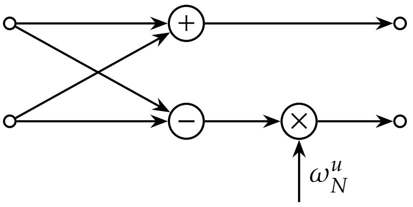


Figure 3.3: Single Radix-2 decimation-in-frequency (DIF) butterfly.

outputs gets multiplied by a constant which is called *twiddle factor* and will be denoted as ω_N^u. The subscript N is the length of the FFT while the index u changes with the location of the butterfly in the signal flow graph (cf. Fig. 3.2). The described procedure is specific to the Radix-2 DIF algorithm and depends in general on the type of decomposition. The basic Radix-2 DIF butterfly is also shown graphically in Fig. 3.3.

Except for the values of the twiddle factors and the degree of data interleaving, the operations at each stage of the FFT are identical for the whole transform. Hence, the Radix-2 butterfly can be identified as the smallest building block of a Radix-2 FFT implementation.

Depending on the data throughput requirements, dedicated FFT implementations can employ multiple Radix-2 butterflies which operate in parallel. An often encountered realization is a fully pipelined architecture which uses one dedicated butterfly for each stage. Below, the most prominent Radix-2 architectures are described briefly. They differ mainly in the data path and buffer memories between the individual butterflies.

MDC pipeline

A straight forward implementation of the Cooley and Tukey FFT algorithm is shown in Fig. 3.4. It is referred to as multipath delay commutator (MDC) architecture [91, pp. 604–609]. It is realized with Radix-2 butterflies which are combined in a DIF decomposition. This architecture can process two samples per clock cycle and needs $\log_2 N - 2$ complex multipliers. The last two multipliers in the pipeline can be replaced by trivial operations, even though the butterflies are still shown in Fig. 3.4 for a better understanding. Furthermore, several buffer memories are required which have the total size $N - 2$.

The input samples have to be provided in parallel and in the same manner, the results are also output in parallel. If a continuous data stream in natural

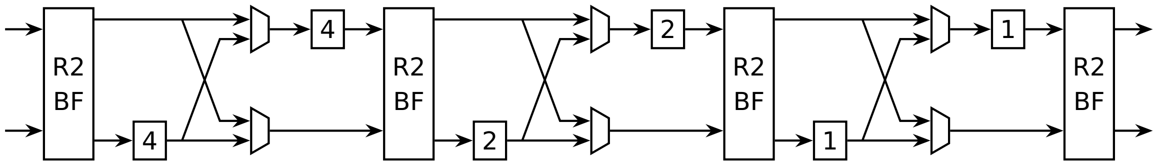


Figure 3.4: Radix-2 MDC pipeline with four stages.

order has to be processed, an additional buffer memory of size $N/2$ has to be added in front of the pipeline. The processing cannot start until the first half of the input data has been written to the input buffer. Hence, the utilization ratio of the processing elements would only amount to 50 %. Furthermore, the total required memory size in this case amounts to $3N/2 - 2$.

Several optimizations have been proposed in order to increase the utilization of the multipliers and memories in the case of a single streaming data input. For example when using feedback networks, the efficiency in terms of memory usage can be improved. This class of pipeline architectures is known as single-path delay feedback (SDF) network [92] and is presented in the subsequent section.

Besides, if multiple parallel data streams have to be processed, the utilization of the complex adders and multipliers of an MDC pipeline can be further increased to 100 %. This can be achieved by using a modified architecture with a proper scheduling of the different data streams [91, p. 607], [93]. In fact, a proper input buffer is added in front of the pipeline so that the two inputs can be fed into the pipeline without interruption. In the case of MIMO systems employing several parallel RX channels, this approach is a good alternative to an SDF pipeline because a single MDC pipeline could be shared amongst two or more channels.

SDF pipeline

A slightly different arrangement of the data buffers and the multiplexer elements inside the processing pipeline helps to reduce the overall memory size. This architecture is known as single-path delay feedback architecture. As the name suggests, only one data path connects the individual butterfly stages. As a result, this FFT realization can process one sample per clock cycle, which is only half of the throughput compared to the peak data throughput of an MDC pipeline. Though, for a single channel streaming application, the average

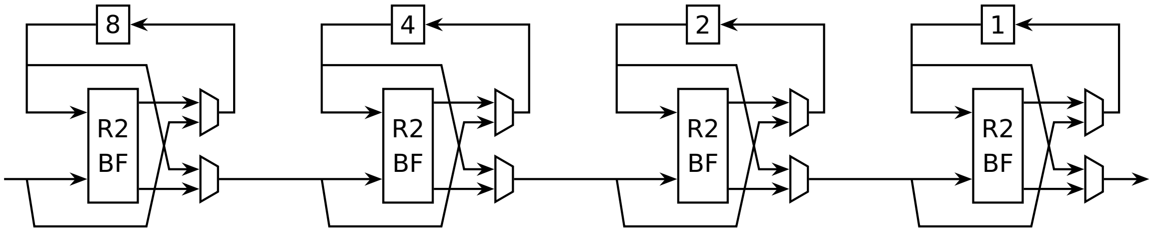


Figure 3.5: Radix-2 SDF pipeline with four stages.

throughput will be the same and this implementation requires only $N-1$ storage elements which is roughly 50 % less than the MDC architecture including the additional input buffer. The number of Radix-2 butterflies and thus the number of multipliers is the same in both cases.

Similar to the MDC pipeline, the multipliers and butterfly elements of the SDF pipeline are only used half of the time so that there is still room for improvement. For instance when using a Radix-4 implementation, the number of multipliers can be reduced at the cost of more complicated butterflies which require more dedicated adders.

Another FFT algorithm for pipelined implementations was proposed by He and Torkelson [94] and is known as Radix-2^2 algorithm. This optimization simplifies the traditional Radix-2 FFT decomposition by considering two butterfly stages at once. When modifying some of the twiddle factors, the multiplications at every second stage can be omitted or rather transformed into a trivial multiplication by $\pm j$. Adopting this modification to the presented Radix-2 SDF architecture, half of the multipliers can be saved.

Even though not optimal in terms of butterfly utilization, a separate Radix-2 based processing pipeline provided by the Xilinx IP Core [95] is used for each channel of the presented MIMO radar system. The principal reason is the faster implementation and integration time. The efficiency in terms of resource usage could be improved in future work, for instance by using a multi-channel MDC architecture.

3.1.2 Fixed-point arithmetic

In digital signal processing systems, all computations are carried out by discrete values, predominantly binary numbers. A discrete value is just an approximation of a real number with a certain accuracy, because only a bounded set of numbers can be represented by a binary word. Basically two different concepts are used within digital hardware, namely fixed-point and floating-point values. Their major difference is the way how the available bit-length is allocated, which results in diverse strengths and weaknesses.

When using fixed-point arithmetic, a continuous range of values is mapped in a linear way to the discrete representation, i.e. the distance between two adjacent numbers is always the same. Each digit in the digital representation has a fixed weight which simplifies further processing. In certain architectures like for instance specific hardware coprocessors, the word length can be chosen arbitrarily which enables an efficient implementation while the application's requirements are fully met. For this reason, many FFT accelerators are based on integer numbers and various models for the induced quantization noise have been developed [96–98].

In contrast, a floating-point value uses some of its digits to store an exponent which can scale the actual value by a power of two. The concept is similar to scientific or exponential notation which can be used to cover a large range of values without losing too much precision for tiny numbers. In fact, the continuous range of values is not mapped in a linear way anymore, because the distance between adjacent numbers is not constant and thus the absolute accuracy changes for each value. This drastically maximizes the dynamic range compared to a fixed-point representation [99, p. 399]. Hence, a system which uses floating-point arithmetic offers an improved flexibility and can be adapted to many different applications.

The benefit comes at the cost of increased computational complexity, because floating-point values require more complicated arithmetic units for basic operations like addition or multiplication. Furthermore, the amount of quantization noise is larger compared to a fixed-point number with the same word length. Therefore, an FFT accelerator built with floating-point arithmetic is not further pursued and a fixed-point realization was considered as a better choice. Given that the two-dimensional FFT will always process the same kind of input data, the dynamic range of the system is bounded and can be specified in advance.

For most signal processing algorithms, a rescaling of intermediate results is necessary in order to prevent growing word lengths. For instance, a multiplication of two fixed-point numbers doubles the word length which is not acceptable in most cases. The solution is to keep only a fraction of the intermediate result for the subsequent processing steps. Though, any truncation or rounding operation in between the actual calculations will result in additional quantization noise. Especially when using fixed-point arithmetic, the impact of the additional rounding errors should be analyzed carefully in the context of the expected signal characteristics.

Bit-accurate model function of an FFT

In the following subsections, several characteristics related to quantization noise are investigated for the two-dimensional FFT implementation. For this purpose, a bit-accurate function of a pipelined Radix-2 FFT accelerator, supporting arbitrary word lengths and rounding modes, was used. The function has been specifically implemented for this work, because existing realizations which were available as Python or Matlab programs, did not prove satisfactory due to their relatively long runtime. Hence, the function was developed in C which enabled a sufficient large number of simulations with many different parametrizations.

The FFT implementation is using a Radix-2 DIF decomposition and the values are rounded and scaled after each stage, in order to emulate the behavior in hardware. The width of the datapath is hence constant throughout the whole FFT pipeline. Furthermore, the width of the twiddle factors can be adapted independently and the rounding mode can be configured.

All studies presented within section 3.1.2 are own work and the results were mainly obtained with the help of this model function. Either random values or actual radar measurements were chosen as input. The real world radar data, used for the analyses in this section, was recorded with the prototype setup presented in chapter 5.

Selection of an ideal word length

In order to specify and compare the numeric performance of a fixed-point implementation, a measurement unit similar to the SNR is used frequently: The signal-to-quantization-noise ratio (SQNR) indicates the ratio between the signal power and the quantization noise power. In general, the higher the SQNR of a system is, the lower the influence of the quantization noise will be. Likewise to the SNR, it is often denoted in logarithmic scale.

A detailed quantization noise survey of the used Radix-2 FFT implementation was carried out during the development of this thesis and was previously presented in [27]. The derivation is based on previous work [100, 101] and similar results were also published in [96]. A simplified noise model was used which assumes a uniformly, independent and identically distributed (i.i.d.) white noise source at each butterfly output. Quantization errors of the twiddle factors were not taken into account, because their word length was chosen large enough so that they had only a minor effect on the resulting SQNR [100]. The assumption of a uniform distribution of the roundoff error seems appropriate, as long as the input signal occupies a wide bandwidth and has a sufficiently large amplitude [102].

It turns out that for increasing FFT lengths, a more accurate arithmetic unit is required in order to maintain a certain level of SQNR. In general two parameters, the word length b and the transform length of the FFT N, have to be considered during the design process in order to meet the SQNR requirements of the system. The relation between the parameters is

$$SQNR = \frac{3\sigma_s^2}{2Nq^2} = \frac{3\sigma_s^2}{2N\,2^{-2b}} \tag{3.2}$$

where $q = 2^{-b}$ denotes the weight of the least significant bit (LSB) and the parameter σ_s^2 describes the variance of the input signal. This equation is only valid for a white noise input signal, whose power σ_s^2 is equally distributed across all frequency bins of the FFT. In such a scenario, if the FFT length N is doubled, the word length b has to be increased by half a bit in order to maintain a constant SQNR.

To illustrate the influence of the word length on real input signals, an exemplary radar measurement was processed with a fixed-point FFT function. For details about the used FFT implementation refer to the previous subsection on page 60. The result is shown in Fig. 3.6 where the blue curve, which was processed with double-precision floating-point arithmetic, acts as a reference.

It can be observed that the power spectra of the fixed-point versions lie all above the reference. The reason is that always a certain amount of the quantization noise power is added to the signal power. As expected, the noise floor is the lowest for the floating-point version. Furthermore, the largest visible deviations from the double-precision reference occur especially in regions with a lower signal power. The difference for 1 bit word length is at most 6 dB, which

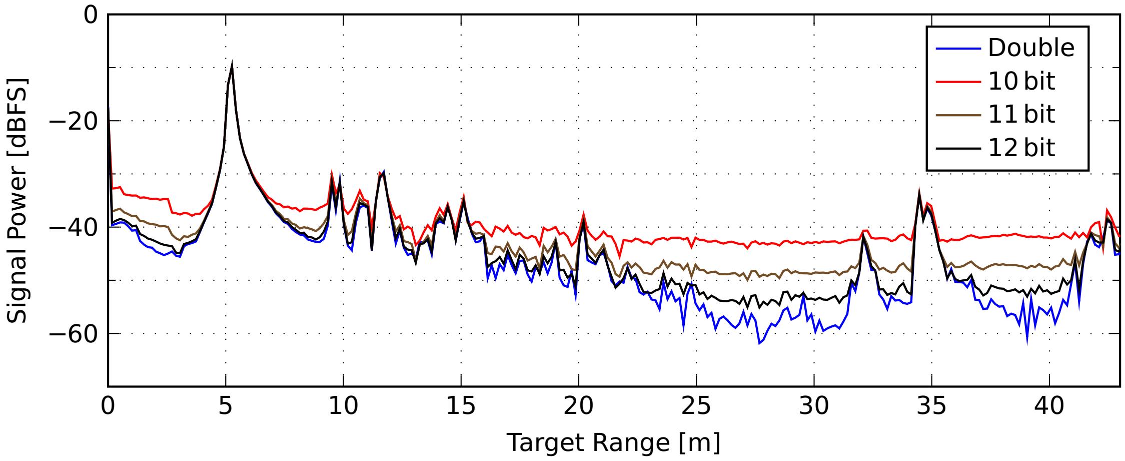


Figure 3.6: Power spectrum of an FMCW measurement consisting of a single chirp. Only the word length of the FFT is varied between the individual curves. The FFT has 512 points and the frequency axis corresponds to the target range.

correlates with the stated noise model in (3.2). In regions with a higher signal power, for instance around 5 m target range, the quantization noise effect is less severe and not visible due to the negligible contribution on the overall signal power.

Impact of the FFT transform length

In a real radar system, the minimum sensitivity is determined by the height of the noise floor in the power spectrum. The SNR of a single target after FFT processing can be expressed by the distance between the signal level and the noise floor. Hence, an SNR of 0 dB means that the signal power is at the same level as the prevailing system noise and the corresponding target cannot be detected without further measures. A common method to lower the noise floor is to take longer signal sequences in combination with a longer FFT transform length, which results in a so called processing gain (cf. section 2.1.4). In contrast, the amount of quantization noise increases for longer transform lengths which can reduce the overall gain.

Fig. 3.7 shows the resulting SNR of a single target from a real measurement for different transform lengths N and word lengths b. As mentioned before, the evaluation was carried out with a fixed-point FFT implementation (cf. page 60), specifically developed in this thesis. The same measurement data is used for all transform lengths, i.e. only a subset of the data is used for the shorter sequences.

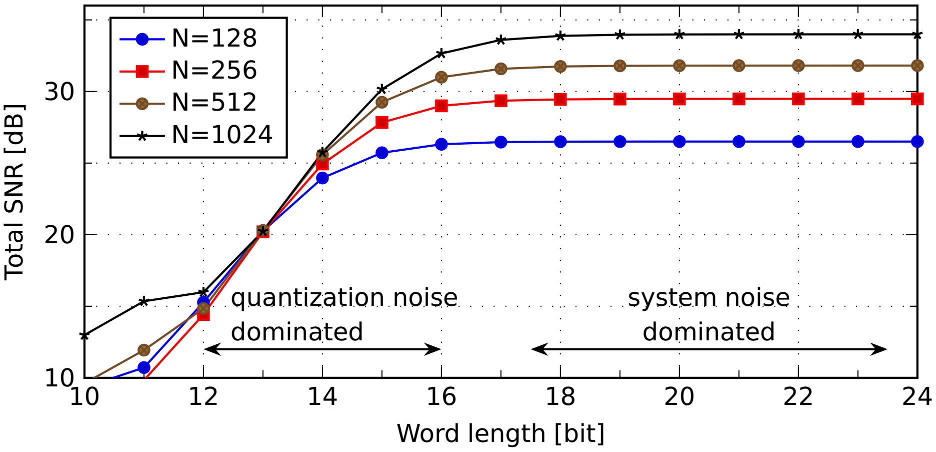


Figure 3.7: Overall SNR comparison of different FFT transform lengths. For very short word lengths below 12 bit, serious artifacts appear in the spectrum which are responsible for the visible outliers.

The shown values are denoted as total SNR, because they originate from the superposition of several different noise components. As long as the individual noise components do not have the same order of magnitude, the largest noise source will dominate the total SNR while all the smaller noise sources do not contribute significantly and can be neglected.

In order to obtain and measure the SNR, the signal level was extracted from a fixed target and the noise level was estimated from several FFT cells which do not contain any target. The absolute SNR level is tightly coupled with the size and the distance of the used reference target, which means that the presented absolute values vary heavily depending on the observed scenario.

In this diagram, mainly two different areas can be identified, the quantization noise dominated section at the left and the thermal noise dominated section on the right. For a sufficient word length of $b > 18\,\mathrm{bit}$, the total SNR gets saturated and the expected processing gain for the different transform lengths can be observed clearly. Each time the transform length N is doubled, the SNR increases by about 3 dB. When looking at the transition phase between the two domains, i.e. around 16 bit word length, it can be seen that for a longer transform length also a bigger word length is required in order to reach the saturation region. This property coincides with (3.2) where the amount of quantization noise is determined by the word length as well as the transform length. However, in the quantization noise dominated section at the left, no difference in SNR between the individual transform lengths can be observed. The reason for this remarkable effect is a slightly different definition of the total SNR and the SQNR which will be explained in the following paragraph.

The SQNR relation in (3.2) considers the input signal to be white noise which will equally spread across all frequency bins of the FFT. However, for a deterministic input signal with a single frequency component, the whole signal energy will concentrate in a small frequency segment or even in a single bin. Hence, if only the relation between the signal energy of a discrete target and the quantization noise is used to extract the SQNR, then (3.2) is not valid anymore. This is the reason why the total SNR in the left section of Fig. 3.7 is independent from the transform length, even though (3.2) suggest otherwise.

For a general radar system application it should be ensured that the added quantization noise does not deteriorate the total signal-to-noise ratio. The SNR is a key parameter for reliable target detection, because smaller targets with a weak echo signal can be masked by the noise floor which makes them undetectable. Hence, noise components arising from fixed-point computations should be clearly below the system noise components (e.g. thermal noise) in any case, i.e. the system should always be situated in the thermal noise dominated domain.

Rounding to integer

Special attention has to be given to the used rounding mode when computing the FFT. As mentioned in the previous section, the intermediate results have to be rescaled after each stage in order to prevent the word lengths from growing too fast. The simplest way is to truncate the values because no extra logic is required for this operation.

The quantization error in the case of a truncation, also known as rounding toward minus infinity, varies between 0 and -1 and can be modeled by a random variable following a uniform distribution [102, 103]. This form of rounding involves a deterministic error contribution due to its non-zero mean component which can be problematic for certain applications. The expected value for the noise components is $-0.5\,q$ for each truncation operation. As long as a single dimensional Fourier transform is considered, a constant bias toward minus infinity will not have an immediate effect on the subsequent target detection operations because the error and thus the noise energy is still equally distributed across the whole spectrum.

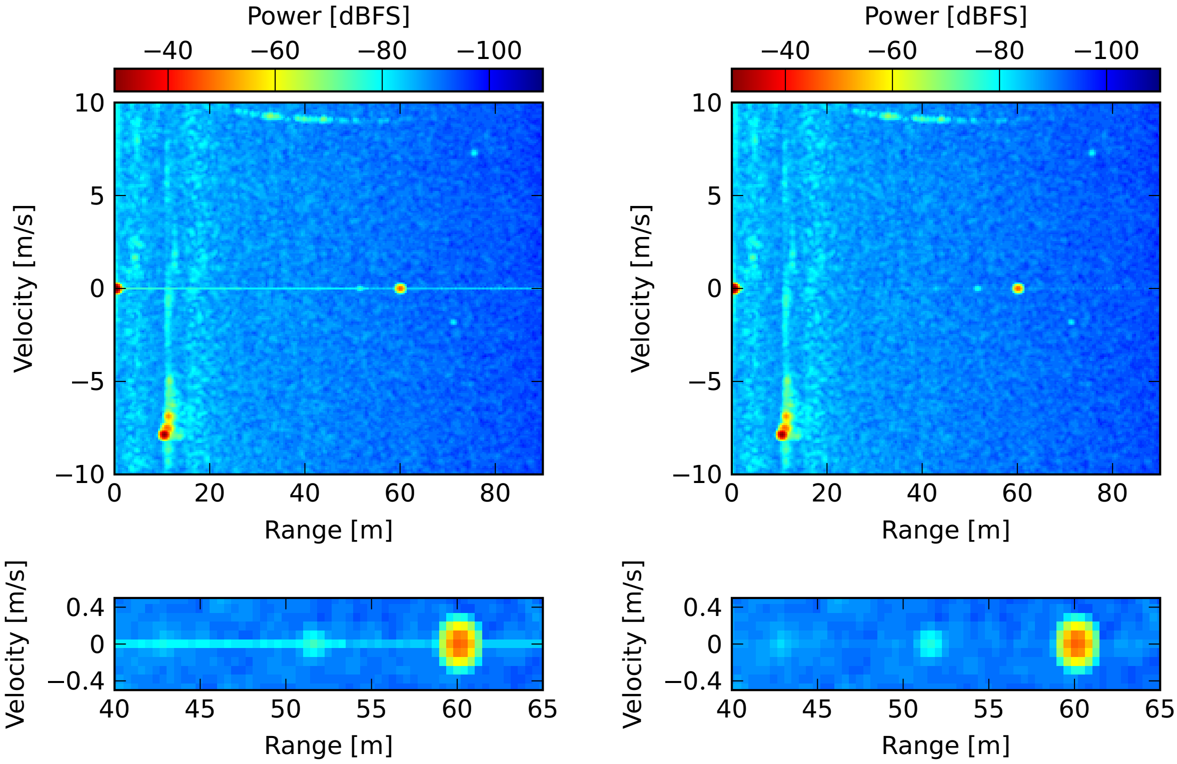
However in the case of a two-dimensional FFT, a non-zero mean quantization noise component has an annoying effect on the resulting power spectrum. The bias introduced by the first-dimension FFT gets subsequently concentrated in the DC bin of the second-dimension FFT, because it adds up constructively due to its non-zero mean noise component.

This effect can be observed in Fig. 3.8 where an exemplary measurement illustrates the influence of the rounding mode of the FFT on the resulting range-Doppler power spectrum. The two small additional figures on the bottom provide just an enlarged view of this effect. The input data consists in both cases of 512 samples times 512 chirps and is processed with a two-dimensional FFT. The word lengths of the FFTs are identical: 17 bits are used for the first FFT, 24 bits are used for the second FFT and the twiddle factors are quantized with 24 bits word length for both dimensions. The figures were obtained with the help of a custom FFT implementation, which has been introduced on page 60.

For the simple truncation-based rounding implementation, a deterministic non-zero mean component manifests itself as a horizontal line in the center of the left spectrum of Fig 3.8. This local deviation of the noise power is certainly cumbersome for the following target detection process, because it resembles a real target. An additional check would have to be performed in order to avoid a false detection near the line where $v = 0$. Furthermore, a real target could be simply masked, and thus the sensitivity for these bins is considerably worse. This situation can be observed for a small target at a distance of approximately 43 m in Fig. 3.8a.

An effective countermeasure is the usage of a more advanced rounding scheme which only induces a zero mean noise component. Even though some extra logic is required for the realization, it can still be more efficient than using a wider word length for the whole implementation.

An actual method is known as *round to nearest* and in a special form *convergent rounding*, which minimizes the absolute error during a single rounding operation (refer to Fig. 3.8b and Fig. 3.9). Special attention has to be paid when the value lies exactly half between two integers. For instance consider a number like 13.5 which could be rounded either to 13 or 14, introducing an error of +0.5 or -0.5, respectively. Such a situation must not always be treated in the same manner, e.g. always rounded down, in order to avoid a small but deterministic bias in the overall error distribution. An established solution is to round to the



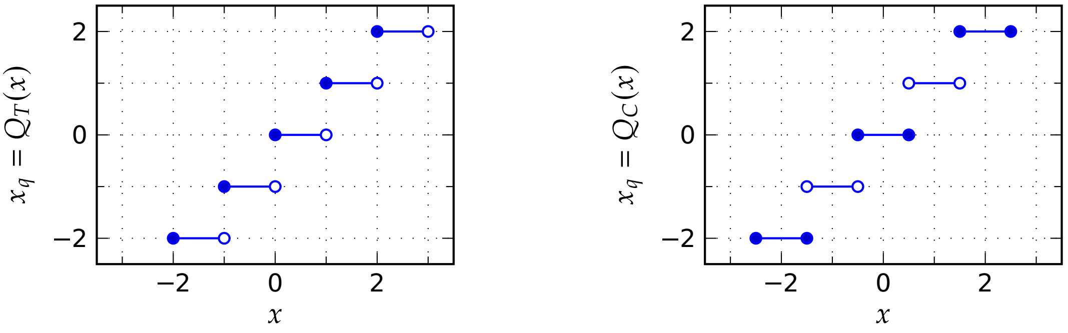
(a) Truncation **(b)** Convergent rounding

Figure 3.8: Influence of the FFT's rounding mode on the resulting power spectrum.

nearest even number if the value lies exactly half in between. Then, no directional tendency is introduced because the value is rounded up and down with the same incident rate, depending on the digit before the point. This rounding mode is referred to with various names, e.g. *convergent rounding* [101], *round ties to even* [99, 104] or *mathematical rounding* [105, p. 391].

Word length optimization for the 2D-FFT

For a quantitative comparison of different word lengths, a proper measuring unit is used which is based on the mean squared error (MSE) of the FFT result. The peak signal-to-noise ratio (PSNR) expresses the MSE in relation to the maximum possible value S_{max} which is also referred to as peak signal. It is



(a) Truncation **(b)** Convergent rounding

Figure 3.9: Comparison of two rounding modes. The input variables x are rounded according to the respective quantization function $Q(x)$ to the result x_q.

usually indicated in decibels and can be calculated as follows:

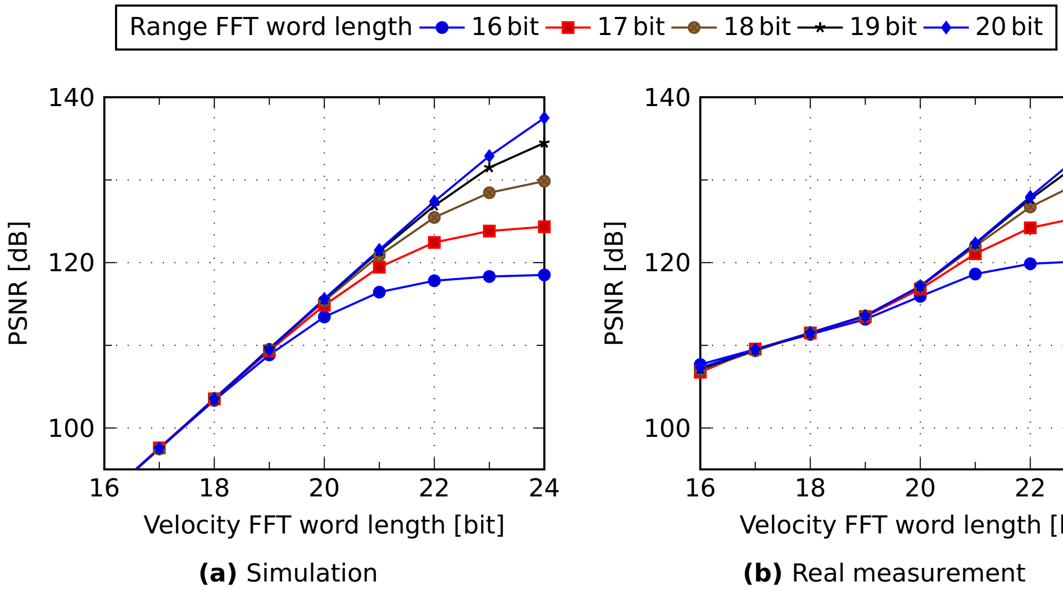
$$PSNR = 10 \log \frac{S_{max}^2}{\frac{1}{NK} \sum_{u=0}^{N-1} \sum_{v=0}^{K-1} \left| \hat{S}_{u,v} - S_{u,v} \right|^2} \tag{3.3}$$

where u, v are the indices of the two FFTs, N, K are the number of FFT samples, $\hat{S}_{u,v}$ is the fixed-point result and $S_{u,v}$ is the noise-free result, mostly computed with the help of double-precision floating-point values.

As mentioned at the beginning of this chapter (cf. page 53), the two-dimensional FFT is computed in two steps with a matrix transposition in between. Intuitively, it seems reasonable to use identical word lengths for the two FFT transforms, so that all rounding operations during the computation cause an error within the same order of magnitude. However, it turns out that if different word lengths are used for the two transforms, nearly the same level of PSNR can be reached but with a significant lower resource usage.

Furthermore, an asymmetric word length implementation can be appropriate when considering additional constraints like the available memory bandwidth. For instance a second optimization goal might be to limit the word length of the intermediate results in between the two transforms while maximizing the PSNR at the output. If the word length of the second FFT is larger than the result of the first FFT, an improvement in terms of quantization noise contribution can be attained to a certain extent.

Fig. 3.10 illustrates this use case for different combinations of the word



(a) Simulation **(b)** Real measurement

Figure 3.10: PSNR level as a function of different FFT word lengths.

lengths. On the x-axis, the word length of the second stage FFT (i.e. the Velocity FFT) is shown, while on the y-axis the corresponding PSNR value is shown. Distinct curves are used for different word lengths of the first stage FFT (i.e. the Range FFT). The twiddle factors were quantized with 24 bits word length for both dimensions and the transform length of the FFTs is 512 times 512 points.

The presented evaluation was performed as part of this thesis' work and makes use of a custom, fixed-point FFT function (cf. page 60). The PSNR value was first calculated based on simulated input values following a zero-mean Gaussian distribution (Fig. 3.10a) and subsequently based on real world data from a moving vehicle on public roads (Fig. 3.10b). Multiple measurements were investigated and the results were virtually similar so that the presented characteristics should hold for a wide range of situations.

The simulated data was obtained by using random input values following a normal distribution. It can be observed that the PSNR gain for the additional bits in the second stage FFT is very close to the ideal behavior of +6 dB PSNR per bit, as long as the word length of the first stage FFT is not too short. Once the word length of the Velocity FFT is more than 4 bits longer, the PSNR does not increase anymore and saturates. The point of the transition into saturation is probably interesting for the system designer, because it can be considered as an ideal word length combination with the highest PSNR to resource usage ratio. The difference of 4 bits has to be regarded as specific value for this configuration and may not be generalized, because it depends on the lengths of the FFT transforms.

The PSNR relation for real measurement data differs clearly from the simulated data. If the word length of the second stage FFT is below 20 bit, the slope of the curves is considerably smaller than 6 dB and the overall PSNR value is larger compared to the simulated input data. It seems that the input signals cannot be modeled appropriately by Gaussian noise in this case. Once the word length is long enough, however, the slope of the curves increases and converges to the simulated ones. Furthermore, the absolute values are quite the same for word lengths of 20 bit and above.

Dynamic scaling between the two FFTs

Additional improvements regarding the utilization of the available memory bandwidth can incorporate block-floating-point strategies. A full-scale amplitude after the first FFT transform is rarely attained, which offers the possibility to further optimize the dynamic range at this point. For instance a radar sensor without any range compensation method will provide much lower signal amplitudes for targets at larger distances. The theoretical free-space path loss for a point target is proportional to r^{-4} and amounts to 40 dB per decade [28, pp. 54 – 57]. This means that more than 6 bits less would be utilized each time the distance increases by a factor of ten (1 bit ~ 6 dB).

The free-space path loss effect is illustrated in Fig. 3.11 where the relative power level of a single target is shown. Multiple measurements from different ranges of a single, static corner reflector with a fixed direction of arrival were conducted, so that the power level as a function of distance was obtained. The visible attenuation effect arises from the range dependent free-space path loss effect as well as from the frequency response of the used radar system.

The measurement was carried out on an empty test track, so that no interfering targets were present. However, multi-path effects with the ground surface could not be avoided which manifest themselves in positive and negative interference. The consequence is a relatively strong variation of the power level at certain distances. Though, the overall trend of the signal power is observable and a power loss of more than 40 dB per decade is noticeable. As a result, the red line indicating the theoretical free-space path loss can be considered as

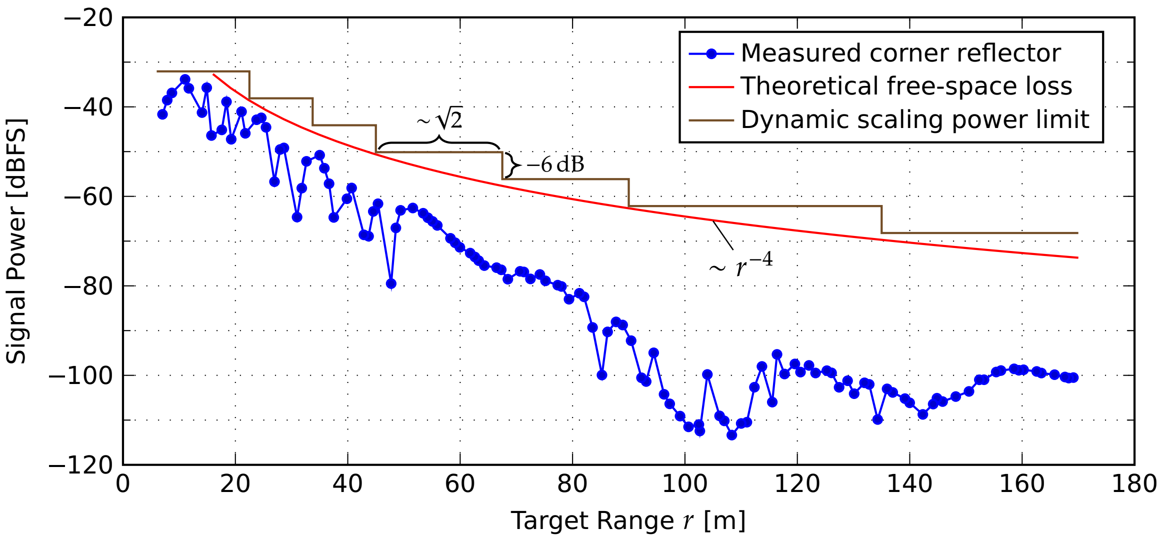


Figure 3.11: Resulting power level of a single target at various ranges.

valid model assumption for the proposed optimization technique.

Actually, a total power loss beyond the theoretical free-space attenuation is visible, i.e. the blue curve decreases faster than the red one and a gap of approximately 25 dB becomes apparent for larger distances. It is supposed that the additional attenuation arises from the frequency response of the used electronic circuits. However, they are only specific to the used prototype radar sensor and the additional power loss should not be generalized.

In this work, a basic range compensation method has been developed and implemented. It shall account for the lower signal amplitudes in farther distances and thus improve the dynamic range. The method has been designed on the basis of own observations and will be presented in the following paragraphs.

According to the expectable attenuation due to the free-space path loss, the frequency spectrum is divided into multiple domains, each having a different power-of-two scaling factor. Hence, the compensation can be realized by a simple bit shift operation. The actual number of bits by which the digits are shifted to the left depends on the target range and is illustrated in Fig. 3.12. The boundaries between the different scaling domains were selected to match the theoretical free-space path loss effect. For an ideal compensation, a scaling of the signal power proportional to r^4 is necessary which corresponds to a scaling of the amplitude proportional to r^2. Thus, the signal amplitude should be doubled each time the target distance increases by a factor of $\sqrt{2}$. However,

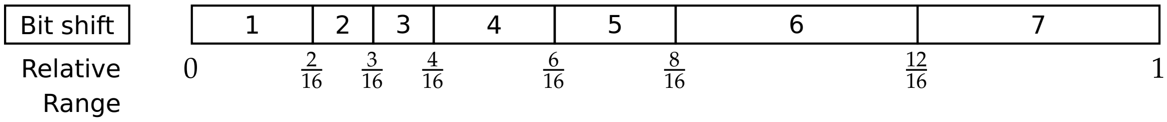


Figure 3.12: Power of two scaling factors which are applied after the first FFT. The numbers inside the rectangles correspond to the bit shift applied to a delimited target range domain.

some of the sections were slightly extended so that the edges fall on a power-of-two fraction which can be calculated more easily. Seven different domains have been chosen in this particular example even though this number can be adapted to the application's needs.

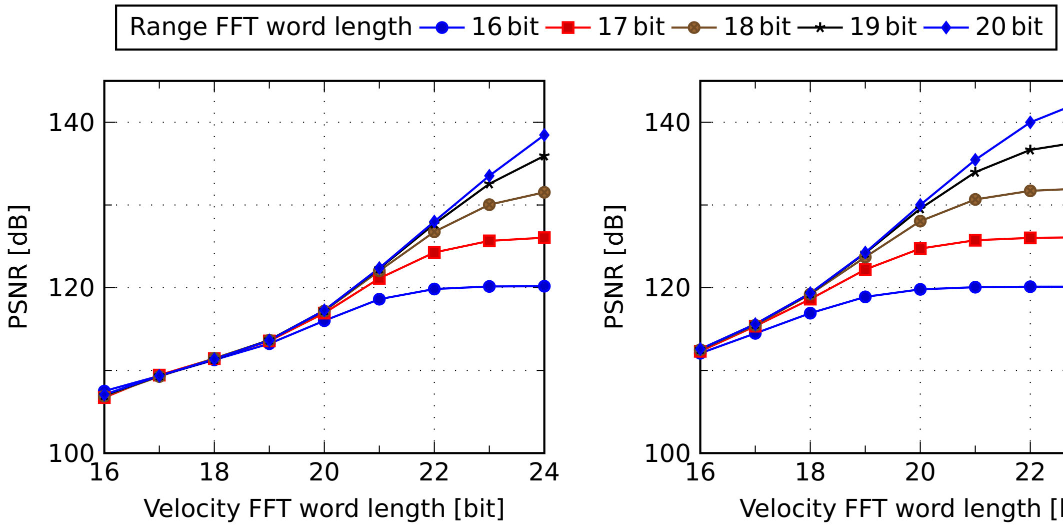
For a better understanding, the scaling schedule has been integrated into Fig. 3.11 by indicating the maximum signal power which may occur at a certain range. Obviously, the curve for this power limit (brown color) approximates the theoretical free-space loss, indicated by the red curve. Note that the absolute value of the power limit is adapted to the reference target (blue curve) for a more convenient representation. In reality, the interspace to the actual power limit of the sensor is more than 20 dB for the used reference target.

When computing the Range FFT, the signal energy is evenly distributed between the positive and negative frequency bins due to the real-valued FFT input. Consequently, the maximum amplitude after FFT processing can only amount to 0.5, given that the FFT is already scaled by $\frac{1}{N}$ during processing and that the maximum input amplitude is 1. Hence, the values in the first domain in Fig. 3.12 are multiplied by a factor of two even though the signal attenuation is minimal for these frequency bins.

During processing, the actual compensation is applied in between the two FFTs. The results of the first FFT are shifted left while their void MSBs[2] are discarded. No overflow should occur, because the MSBs equal zero due to the small amplitudes at larger target ranges. A simple check during runtime can monitor any overflow event, even though no further measure is taken to avoid such a case. The bit shift is predetermined for each frequency bin and no additional information incorporating the current exponent is stored in order to save memory bandwidth. The actual bit shift can be obtained from the current range index information. It is finally reversed after the second FFT and the

---

[2] Most significant bit



(a) No scaling **(b)** With dynamic scaling

Figure 3.13: Comparison of the PSNR level with and without dynamic scaling. Values are based on measurement data from public roads.

CFAR processing, allowing these signal processing steps to benefit from the improved dynamic range.

When using the proposed scaling scheme from Fig. 3.12 in practice, not a single overflow condition was observed during several hours of test drives, corresponding to more than 10^6 measurement cycles, so that the free-space path loss model assumption appears to be valid. The additional gain in PSNR compared to an implementation without range dependent scaling is shown in Fig 3.13.

The PSNR level after a two-dimensional FFT processing was calculated with the help of real measurement data and is shown for different word lengths. The twiddle factors are quantized with 24 bits for each dimension, and the size of the FFT is 512 times 512 points. The left figure does not use any scaling between the two transforms, while the right figure shows the result incorporating the proposed dynamic scaling scheme. The computations have been performed with a fixed-point FFT implementation, which was specifically developed for this thesis (cf. page 60).

It can be observed that a considerable amount of quantization noise can be avoided, if the signals are scaled in between the two FFT transforms. The required logic for this operation can be minimal, e.g. a barrel shifter which is controlled by a simple counter is sufficient for the proposed scaling procedure.

The advantage is a considerably higher PSNR level while the resource usage stays more or less constant.

3.1.3 High-throughput matrix transformation with SDRAM

After the FFT of the first dimension has been computed, i.e. the Range FFT, the intermediate results have to be stored in a buffer. The computation of the second dimension FFT, i.e. the Velocity FFT, cannot start before the Range FFT of the last frequency chirp has been written to this buffer. Hence, a quite large memory is required which has to cope with a high data rate at the same time. In the following sections, a solution based on SDRAM[3] is presented, which will be used throughout this work.

Requirements to the memory

The matrix transformation in between the two dimensions of the FFT has a rather high requirement in terms of data throughput, because the full data amount of the chirp sequence has to be written to memory and read again during each processing cycle. Unfortunately, the large data amounts impede the usage of fast memory cells realized as an on-chip buffer. Even with shrinking semiconductor technology, the storage volumes which exceed easily 100 MByte and more [36], cannot be handled efficiently by on-chip RAM. It is thus obvious to employ an external DRAM memory for this processing step which offers a larger and more cost-efficient storage capacity.

If an FFT transform of real input values is computed, like it is the case for the Range FFT, only half of the results have to be transferred to RAM due to the symmetry of the transform. At the same time, these results now consist of complex numbers so that the overall data amount is still not reduced after the first FFT. Furthermore, the word length of the transform is often chosen larger than the word length of the input data to avoid quantization noise effects and to accommodate the processing gain of the FFT (cf. also section 3.1.2).

For the radar system used in this work, the maximum data rate at the DRAM interface amounts to 64 GBit/s if all 16 channels with a 16 bit word length are used at their maximum sample rate of 250 MHz. Such a high data rate can

---

[3] Synchronous dynamic random-access memory

already exhaust current state of the art DDR4 interfaces so that special attention has to be paid to the addressing scheme.

The maximum data throughput of a DRAM interface is given by its data width and the operating clock frequency. A state-of-the-art DDR4-2400 module with a 64 bit interface would accordingly achieve a peak data transfer rate of $2400 \cdot 64\,\text{MBit/s} \approx 150\,\text{GBit/s}$. However, this value is only a theoretical boundary which cannot be reached in most applications due to certain limiting factors.

Refresh cycles Periodic refresh cycles are required to avoid data loss due to leakage currents. These memory refresh operations occur in between the actual read and write operations and do not contribute to the achievable data transfer rate.

Read/Write turnaround The data bus is shared between read and write operations which imposes a certain turnaround penalty. This means that an additional delay has to be introduced each time the data transfer direction is about to change. For faster data bus speeds, the relative delay measured in number of clock cycles increases and becomes more significant [106].

Access time An important constraint of DRAMs are the intrinsic memory timings which specify a minimum latency for each data access. Depending on the type of the operation, a certain delay is added which is typically measured in number of clock cycles of the interface speed. There are always different kinds of delays specified, because certain commands can be executed more quicker than others.

Special attention has to be paid to the employed addressing scheme, because each memory location is determined by multiple address parts. Conventionally, DRAM is divided into *rows* and *columns*, which can be implemented several times in parallel in multiple memory *banks*. Since the introduction of the DDR4 standard, multiple banks are further organized in *bank groups* which can also be selected by an extra address input [107].

To access a single memory location, the corresponding row of the memory, also known as *page*, has to be activated first and then the desired column can be accessed in a subsequent command. Once a single row has been opened, the following column operations can be executed more quickly. Hence, a strategy which minimizes the number of row activations will increase the efficiency of the memory system. For CPU-based systems, often the whole page is transferred into the processor's cache even though only a single data word out of it is required.

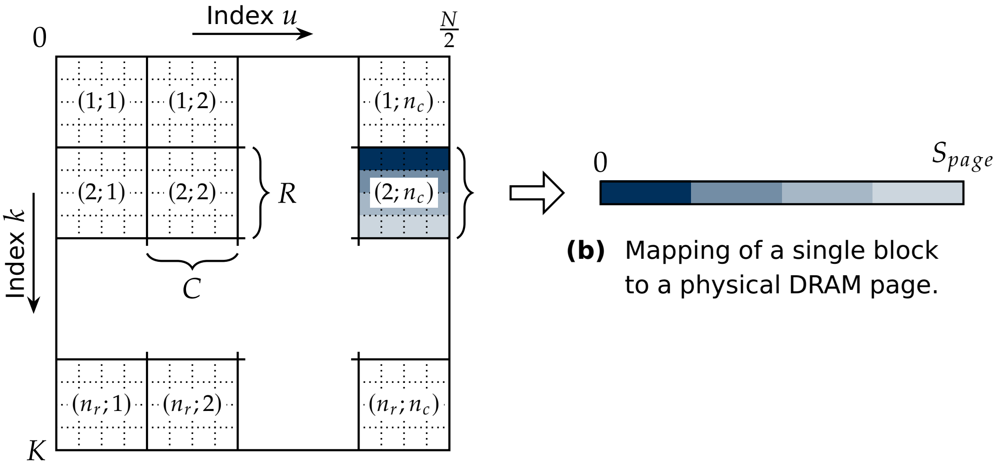
A typical delay for a column access amounts to 16 clock cycles for a DDR4 module [108], but the actual data is transferred in only four clock cycles. In other words, the memory interface operates much faster than the DRAM itself. Fortunately, multiple commands can be interleaved because the internal banks of a single memory circuit can operate in parallel. A proper command scheduling is thus likewise essential if a high utilization rate of the data bus is desired. A suitable strategy maximizing this efficiency is described in the following section.

Optimized addressing scheme for SDRAM

The most obvious way to map a two-dimensional array to a linear memory address is to use either a row-major or column-major order. In the first case, the subsequent elements inside a single row are located next to each other. In the latter case, the data is stored column by column, i.e. the consecutive elements of one column are located nearby.

If used in combination with an SDRAM, such an addressing scheme will only perform well when accessing contiguous data in one certain dimension. For instance, if a row of the data matrix is mapped to a single row of the SDRAM, then a row-wise access is ideal and the number of required row activations is minimal. However, a column-wise data transfer will result in a very poor performance. Then, for every element of a single column, a new row has to be activated which introduces a large delay. Hence, additional measures should be taken when a matrix transformation at high data rates is desired.

A common solution used by many implementations is to employ a window based address mapping scheme [109–111]. The idea is to fill each memory row evenly with consecutive data words from both dimensions. Instead of filling the whole row by a continuous data fragment of the first dimension, only a short block is written to the beginning of the row, so that a bigger part of the row remains unused. Some of this free space is then later filled by a second block when the index of the second dimension increases, and so on. The result is that each memory row contains consecutive data words in both dimensions.



(a) Range-Doppler matrix, divided into blocks of the size $R \cdot C$.

(b) Mapping of a single block to a physical DRAM page.

Figure 3.14: Window-based address mapping.

The overall row activation count is minimized by this approach which helps to improve transfer speed [109] and energy efficiency [111].

This approach is illustrated in Fig. 3.14, where the full data matrix holding $\frac{N}{2} \cdot K$ samples is decomposed into several rectangular blocks. Only half of the N samples of the Range FFT have to be stored due to the symmetrical spectrum. Each block of the size $R \cdot C$ contains several data samples of both dimensions with the word length B. The block size is chosen in a way so that the data amount matches the capacity S_{page} of a single row of the employed SDRAM.

$$R \cdot C \cdot B = S_{page} \tag{3.4}$$

In addition to the window address mapping scheme, a bank interleaving scheme similar to the approaches described in [112, 113] was used to further optimize the memory transfer rate.

The assignment of a data block to a physical memory row is depicted in Fig. 3.15. Each row of memory holds one data block, which further holds $R \cdot C$ samples. The data blocks are indexed by their position inside the two dimensional matrix, as it is shown in Fig. 3.14.

Adjacent blocks are then written in an interleaved manner to different memory banks. This happens likewise for both dimensions, so that in total four memory banks are required. With this approach, a transition from one data block to the next block will always result in the usage of a different memory bank. Accordingly, the row activation delay can be successfully hidden, because an alternation of the currently used memory bank will occur each time a new data block has to be addressed. This is likewise true for a row-wise and a column-wise data access.

When observing the data block to memory bank assignment in Fig. 3.15, it can be seen that Bank 1 holds the blocks with an odd column index, while Bank 2 holds the blocks with an even column index. Both, Bank 1 and 2 store only blocks of an odd row index. Accordingly, Bank 3 and 4 store data blocks with an even row index.

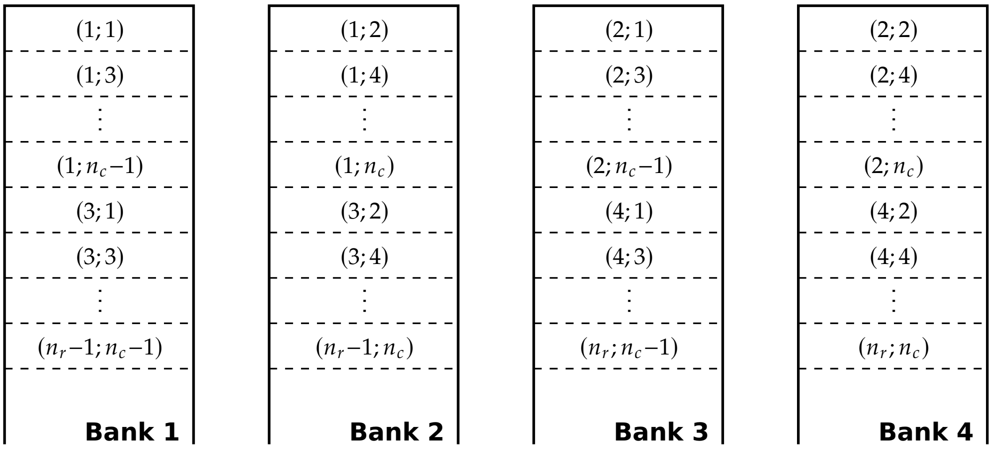


Figure 3.15: Optimized bank interleaving scheme with four memory banks.

3.1.4 Window function

It is known from theory, that the Fourier transform of a signal which contains only a single frequency component will produce a sharp peak in the spectrum. For instance, the transform of a real-valued sinusoid with frequency ω will be zero, except at the positive and negative frequencies $\pm\omega$. Such a behavior would be ideal for a radar system, because superimposing signals with different frequencies originating from closely spaced targets could be separated with infinite accuracy.

Though, in practice it is not possible to obtain a signal of infinite duration. Rather, only a finite excerpt or time frame of the actual wave can be measured and processed. Hence, the spectral transformation of a time-windowed sinusoid will suffer from spectral leakage effects. That is, a multiplication with a

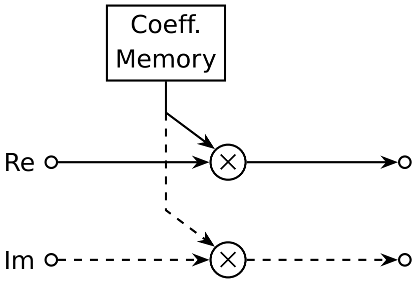


Figure 3.16: Application of an arbitrary window function prior to the actual FFT computation. The coefficients are stored in an on-chip SRAM. The imaginary data path only exists for the Velocity FFT.

certain window function $w[t]$ in the time domain will produce a convolution of the transform of the actual signal $S[k]$ with the transform of the window function $W[k]$. For instance a rectangular window in the time domain, resulting from a finite number of input samples for the FFT, will result in a sinc function in the frequency domain. Especially the sidelobes of the sinc function are problematic because they could mask real targets with a lower signal amplitude. Hence, the application of appropriate window functions prior to the actual FFT computation is an established measure to lower the amount of spectral leakage which finally improves the detection performance.

Many different functions have been reported and investigated in the past [114, 115]. With parametric window functions, a large variety of different functions each with its proper characteristics exist. A trade-off between sidelobe suppression and target separability has to be made, so that it is difficult to chose one window function which fits for all possible applications.

In order to maintain a certain degree of flexibility, the employed implementation of the window function uses a run-time configurable memory which stores the function coefficients. During processing, the values are read from the memory and applied to the actual input data stream in linear order. A simple counter is used as address generator which counts upwards until the $\frac{N}{2}$-th sample is reached. Then the direction is changed and the same values are read in inverse order. This is possible because all window functions are symmetric so that half of the storage space can be economized.

The realization of the module is shown in Fig. 3.16. The second, dashed datapath on the bottom is only required for the Velocity FFT which is computed on complex-valued input data.

3.2 Target detection

The main objective of the target detection step is to distinguish target echoes from interfering noise and clutter components, as it was described in section 2.2. The target detection block operates on the evaluated frequency spectrum values and provides a binary decision signal for each frequency cell. Only those cells which are considered as a valid detection will be further processed by the subsequent algorithms. Therefore, this step can help to reduce the data rate which lowers the requirements for all downstream modules.

The target detection process used in this work is mainly based on two sub-steps, namely constant false alarm rate (CFAR) processing and non-coherent integration (NCI). These steps aim to mitigate varying operational conditions as well as to improve the detection performance. First, the realization of the CFAR processing module will be presented in section 3.2.1. Afterwards, the implementation of the NCI will be described along with the architecture of the overall detection module in section 3.2.2. For the underlying theory of the algorithms, refer to section 2.2 in the previous chapter.

3.2.1 Rank-only OS-CFAR

It was explained in section 2.2.1 that the OS-CFAR processing performs very well in multi-target scenarios. Thus, it is considered as a suitable candidate for automotive radar sensors and will be further investigated in this section. Primarily, an efficient implementation of the OS-CFAR processor is presented and analyzed.

The main task of an OS-CFAR realization is to find the k-th largest value inside a window consisting of N values. The k-th value serves as a noise estimate by means of which an adequate decision can be made, whether the cell under test (CUT) should be considered as target or not. In general, a straightforward solution to this problem is to sort all cells inside the window which can be a quite expensive operation. Hence, a direct implementation of the algorithm can be cumbersome if N becomes too large. Unfortunately, with increasing resolution capabilities of the sensors, that is exactly the case: The CFAR window needs to grow which is explained in the following paragraph.

The basic assumption in the case of a CFAR processing is that the bigger part of the window cells only consist of noise components. Hence, the minimum window size of a CFAR detector should be designed so that even large and widespread targets will never occupy major parts of the window. Consequently, if the range and velocity cells are becoming smaller and more fine-grained, then a higher number of cells is required to maintain a certain spatial expansion of the window. For this work, window sizes of 64 cells and larger were used, which cause a high sorting complexity. It should be kept in mind, that the OS-CFAR procedure has to be applied to each cell of the range-Doppler spectrum which can easily comprise a million bins. So even if a list of 64 values can be easily sorted, this procedure would have to be repeated a million times during one measurement cycle.

Interestingly, the complete sorting of the whole window is not necessary to obtain a decision result according to the OS-CFAR algorithm. Hence, a brute-force implementation which relies on a full sorting would not be a very efficient solution. Most importantly, two aspects of the OS-CFAR sorting characteristics have to be considered:

- Only a single value of the sorted list, the future noise estimate, is of interest, while all other entries are discarded. In other words, the algorithm is not dependent on a fully sorted list, it only requires to know the k-th largest value in a set of items.

- When evaluating a continuous spectrum, the CFAR processing can be applied as sliding window, i.e. neighboring cells are evaluated one after another. In this case, the previously sorted list can be kept in memory and used as a starting point.

Several optimized implementations of the algorithm aim at these specific sorting characteristics. For instance, a "k-th maximum search" can be performed which finds the greatest value and removes it from the set. This step is repeated until the k-th value has been found [85]. Another efficient realization uses a sliding window approach which keeps a sorted list in memory [86]. Now, when moving the window one step further, the insertion of a single value requires at most N comparisons.

Besides, if one is only interested in the decision result, the complete sorting of the list can be bypassed and the detection step can be performed in a "rank-only" manner [84]. According to this method, the inverse threshold is applied to the CUT and the result is compared to each cell inside the window. The

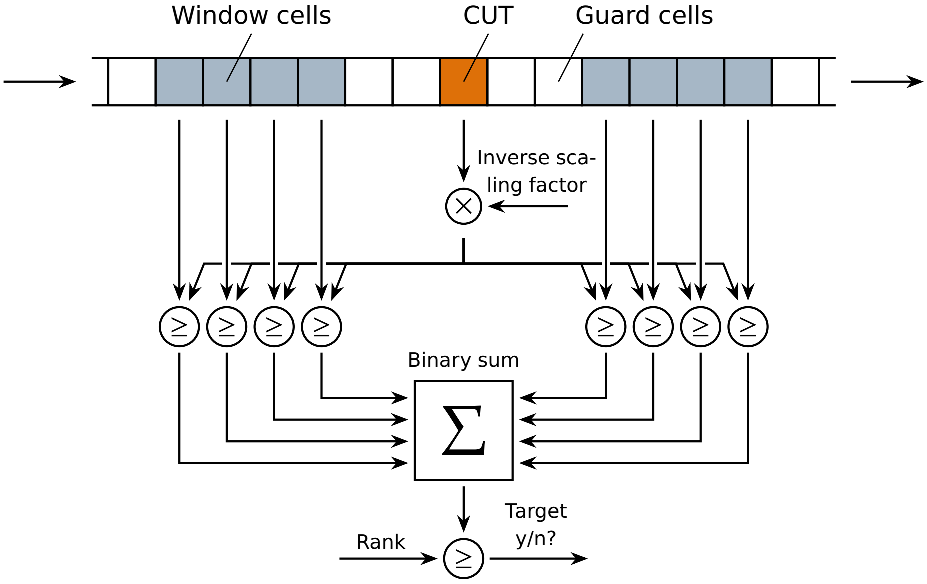


Figure 3.17: Realization of the CFAR module as rank-only variant.

binary comparison results, i.e. 1 if the value is bigger – 0 if not, can be summed up to get a rank. Only if the actual rank is higher than the desired one, the CUT is considered as valid detection. This approach is depicted in Figure 3.17.

In contrast to a complete sorting, the rank-only algorithm depends only on N comparisons. The complexity is thus linear for growing window sizes. The target decision result is exactly the same, i.e. there is no performance loss. The only disadvantage is the lack of the k-th value, which is not determined by the rank-only algorithm. This value could serve as an estimate for the local noise level and may be required by subsequent signal processing blocks. A supplementary estimation of this value could be envisaged in this case, e.g. the computation of the mean value of all neighboring cells which have already been classified as noise.

FPGA implementation

The rank-only implementation form has several advantages, especially due to its low computational complexity even for large window sizes. Thus, it was further investigated in this work by a novel FPGA implementation. The architecture of the module is shown in Fig. 3.17 and more implementation specific details are provided below. In contrast to the work presented in [84], fixed-point arithmetic was used in this case. Consequently, a significant better area efficiency is achieved so that the observed resource usage is lower by approximately a factor of three for a word size of 16 bit (cf. also Tab. 4.1 on page 117).

The sliding window of the CFAR processor is implemented with the help of a shift register. Current FPGA devices offer several different building blocks which can be used as on-chip memory, namely Block RAMs, lookup tables (LUTs) and ordinary flip-flops. For the presented OS-CFAR architecture all signal values inside the window need to be accessed and read at once. Hence, a data tap is required at each position of the shift register and solely flip-flops can be used for its realization.

Each register of the sliding window is routed to a dedicated comparator, whose second input is fed by the CUT with the inverse scaling factor applied. The comparison result is routed to a binary adder with N inputs. Several LUTs are cascaded for this step, which could impose an upper limit to the clock frequency. In order to maximize performance, the sum is implemented in two steps, i.e. the lower and the upper half of the window are added up separately before the final rank is computed. That is, an extra register stage is inserted and the adder is thus pipelined. If a further improvement would become necessary, the binary sum could be split into smaller chunks which are added together in even more stages.

3.2.2 Target detection accelerator

The rank-only OS-CFAR module which was described in the previous section can be further combined with an NCI module. This has several advantages in terms of detection performance (cf. section 2.2.2), but also in terms of resource usage as it will be shown in this section. The processing pipeline can be designed so that a continuous data stream can be handled by the sliding window of the CFAR implementation. The delay which is introduced by the NCI and CFAR processing can be compensated by an adequate buffer. Hence, an on-the-fly target detection is possible, before the raw FFT results from the spectrum evaluation are transferred to the next module, interface or memory. This helps to save valuable interface bandwidth because all noise components can be eliminated immediately.

The overall architecture of the target detection accelerator is shown in Fig. 3.18 and remarkably, it operates like a filter. The data itself is not changed inside

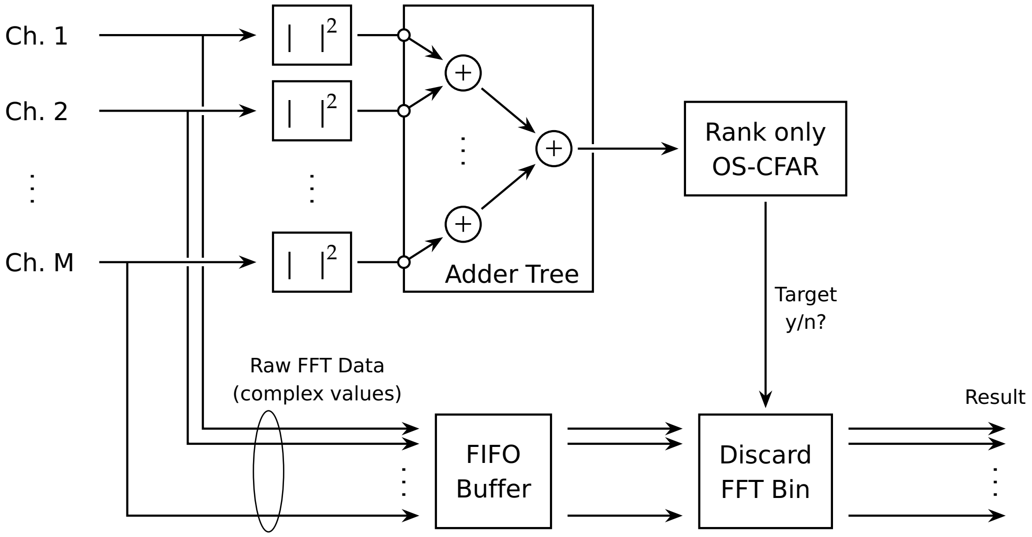


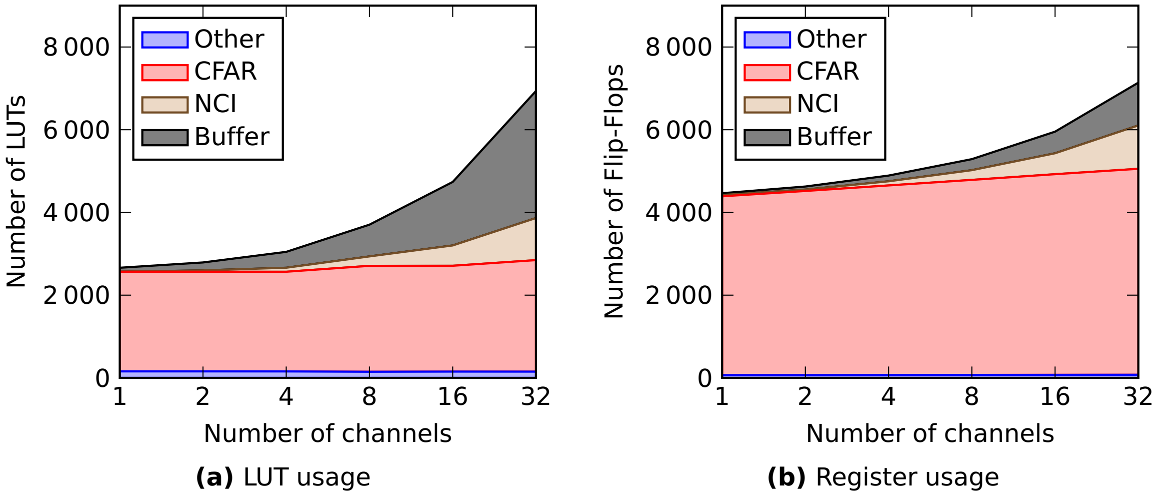
Figure 3.18: Block diagram of the comprehensive target acceleration module, including NCI and OS-CFAR.

the module, however certain data cells are dropped and discarded according to the CFAR result.

The whole module can process one sample per clock cycle without interruption. Only at the boundaries of the data matrix, the CFAR window has to be filled anew. This is accomplished by supplying the data from the opposite side of the matrix as a prefix. Hence, the data is circularly repeated which conforms to the cyclic properties of the complex-valued Velocity FFT. Accordingly, a bin at one end of the spectrum can be considered as immediate neighbor to a cell at the other end of the spectrum.

The NCI, which is basically a summation of the squared amplitudes of all receiving channels, is implemented as a binary adder tree. This permits the insertion of multiple pipeline registers which helps to assure a certain performance level for growing channel numbers. At the same time, the additional delay is small compared to the CFAR latency. At each stage of the tree, the values grow by one bit because a rounding during the integration has been avoided.

The raw data buffer which is drawn in the bottom of Fig. 3.18 operates as first in, first out (FIFO) memory and its size has to be large enough to store a data sequence with at least half the length of the CFAR window. The reason for this is the minimum latency until a decision result for a supplied FFT cell is available, due to the required filling of the CFAR shift register with successive



(a) LUT usage **(b)** Register usage

Figure 3.19: Submodule resource usage against number of channels for a constant window size (128 cells).

values. The raw data buffer can be realized efficiently on an FPGA using LUTs or Block RAMs. An intermediate data tap is not required, as it was the case for the CFAR window shift register.

Regarding the resource usage, the CFAR-processing part (red color in Fig. 3.19) is practically independent from the number of channels, because the NCI step is performed in advance. The NCI step by itself scales approximately with linear complexity, which is a result of the used tree structure. It can be concluded that the usage of NCI before the actual CFAR processing is beneficial in two ways: It improves the detection performance and reduces resource requirements at the same time.

For a number of channels above 32, the raw data buffer which compensates for the pipeline delay consumes more LUTs than the CFAR processing part. It grows also linearly with the number of channels and is the dominating part for large channel numbers. The usage of a dedicated Block RAMs may be considered in this case if the number of LUTs is scarce.

In relation to the overall resource usage of the whole signal processing chain, the requirements of the CFAR block are relatively small. Nevertheless, the integration of a CFAR hardware accelerator along with the other algorithms is crucial because it avoids a data transfer of the full frequency spectrum to the downstream modules or interfaces. Experiments have shown, that in certain environments more than 99 % of the data cells can be considered as noise (cf.

section 5.3.2). Hence, in this case, the CFAR processing could lower the data throughput for the following block by two orders of magnitude.

3.3 Angle estimation

In chapter 2.3, several different radar technologies were introduced which allow to estimate the direction of arrival (DOA) of an incident signal. The functional principle of antenna arrays was explained and the approach of using a steering vector which describes the angle dependent array response was introduced briefly. Many different algorithmic solutions to estimate the DOA can be identified and they are generally examined within the research field of array processing [46,47].

A contribution of this work is to investigate efficient implementations of computational demanding DOA algorithms on dedicated hardware architectures. Due to the huge algorithmic diversity, only the most promising approaches for DOA estimation in the context of automotive radar have been chosen, namely maximum likelihood (ML) estimators. Previous work already compared the adoption of several DOA algorithms to automotive applications based on their estimation performance. The findings are summarized in section 2.3.3 and the choice of the ML estimators is motivated.

Below, various hardware accelerator architectures of different ML estimators that were designed and implemented on an FPGA are presented. They are introduced in section 3.3.1 and 3.3.2, respectively. These realizations are further used in chapter 4 to explore the design space and to form models of their resource and power requirements. Finally the performance is benchmarked and compared to a highly optimized CPU-based implementation.

3.3.1 Single target ML estimator

Recalling the expression for the maximization problem of the ML estimator $\hat{\theta}$ from (2.34), the following equation has to be solved if the number of incident signals is fixed to one. The notation $\boldsymbol{x}^H$ is used for the *conjugate transpose* of $\boldsymbol{x}$, which is further explained in the footnote on page 38.

$$\hat{\theta} = \arg\max_{\theta} |r_\theta|^2 \quad \text{where} \;\; r_\theta = \boldsymbol{x}^H \boldsymbol{a}(\theta) \tag{3.5}$$

Hence, the computation of the ML angle estimation in the single target case consists mainly of a scalar product between two complex-valued vectors. First, the measured array output $\boldsymbol{x}$ has to be multiplied by all steering vectors $\boldsymbol{a}(\theta)$ in the search space of θ. Usually, the search space is defined by the sensor's field of view, i.e. the opening angle of the antennas. Afterwards, the absolute square value of the scalar product r_θ has to be calculated and a subsequent maximum search is required. The final result of the computation is the index of the maximum which can be translated into the estimated DOA $\hat{\theta}$, and the value of the maximum which gives an indication of how well the signal model matches the data. A low maximum value suggest the presence of multiple superimposing signal sources which cannot be handled by the single target estimator.

The three major steps of the algorithm are listed in their order of execution, each with their computational load in terms of required additions and multiplications. M denotes the number of (virtual) channels and N denotes the number of steering vectors. The values hold for the processing of a single detection, even though numerous targets will have to be handled during each measurement cycle.

1. Elementwise multiplication and summation $-$ $\boldsymbol{x}^H \boldsymbol{a}(\theta)$

   $M \cdot N$ complex multiplications; $(M-1) \cdot N$ additions

2. Absolute square value $-$ $|\quad|^2$

   $2 \cdot N$ real multiplications; N additions

3. Maximum Search $-$ $\arg\max_\theta$

   $N-1$ comparisons

Obviously, the scalar product between the two vectors is the prevailing part and thus the most promising to accelerate. Fortunately, the element-wise multiplication can be executed independently and provides a first possibility for parallelization. Furthermore, the scalar products of different radar detections are also independent so that the DOA for multiple targets could be estimated at the same time. In the end, these two different means of parallelization can be combined in order to meet the application's requirements.

Only the first step of the algorithm, the computation of the scalar product, is dependent on the number of channels M. Such a dependency is in general not desired because the system should be capable to fulfill a given real-time constraint in terms of number of detections, irrespective of the number of channels. Hence, a major simplification for the system designer is to decouple the runtime of the angle estimation module from the number of channels M, by implementing a full parallelization of the element wise multiplication and addition. Such a parallelization scheme is employed in the proposed architecture and consequently, the throughput in terms of radar detections per time is made completely independent from M. Solely the resource usage will vary if more channels are present.

Multiplier-based implementation variant

A straight forward implementation of the single target estimator employs complex multipliers for the first step of the algorithm. The complex multiplication is realized in two variants by several real operations according to the following two equations.

$$(a+ib)(c+id) = (ac-bd) + i(ad+bc) \tag{3.6}$$

$$(a+ib)(c+id) = c\,(a+b) - b\,(c+d) + i\,[c\,(a+b) - a\,(c-d)] \tag{3.7}$$

The realization according to (3.6) employs four real multipliers along with two adders. They can be implemented in a regular structure with two stages where the first one consists of the four multipliers and the second one of the two adders. A more resource efficient implementation of a complex multiplication is described by (3.7). Only three multipliers are required while the number of adders increases to five. A minor disadvantage is the slightly higher delay due to the extra stage required for the final addition [116, p. 65].

The subsequent step after the element-wise rotation is a complex addition of M elements which should be executed in parallel as well, so that the data throughput matches the previous step. A tree-based implementation scheme with 2-input adders is used in order to simplify the pipelining of this step. At each node in the tree, two values are summed up until the final sum is obtained. The required resources are $M-1$ adders and the delay of this operation increases with $\log_2 M$. Depending on the target architecture, other realizations may be advantageous, such as the usage of tenary adders (3-input adders) for certain FPGA types [117]. This specific optimization aims to fully utilize all six inputs

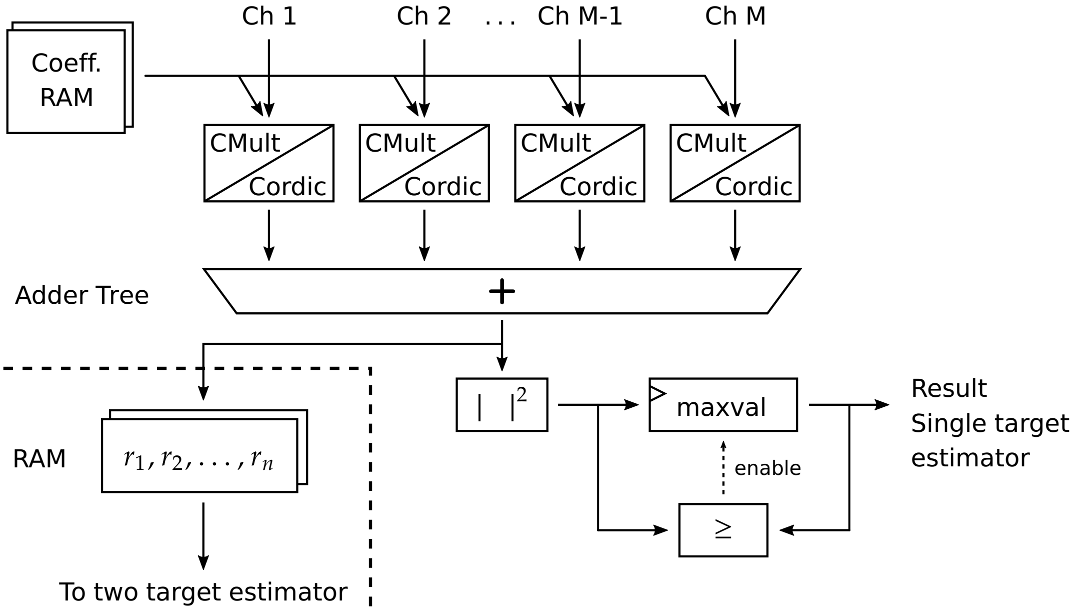


Figure 3.20: Block diagram of the single target estimator module.

of the available lookup tables (LUTs) in order to save resources.

The final maximum search consists of a simple comparator with an upstream module computing the absolute square value. Two multipliers and one adder are required to calculate the absolute square value and one comparator along with a register are used for the maximum search. Each time a larger value is found, it is stored in the register "maxval" along with the current index which carries the information about the DOA. The value of the maximum $|r_{\theta,max}|^2$ gives an indication about the estimation quality.

The realization of the described architecture is also shown in Fig. 3.20 where the elementwise multiplications are denoted by "CMult". However, these blocks can be replaced by another implementation variant referred to as "Cordic" which is further explained in the following section.

Cordic-based implementation variant

In many radar sensor realizations, the used antenna arrays are homogenous in the sense that all individual antenna elements have the same directivity and gain, which results in an equal amplitude response of all elements [3, p. 365]. An exemplary antenna diagram of a prototype sensor with such a homogenous antenna array is shown in Fig. 2.9a. A more or less identical amplitude curve for each channel can be observed which reaches its maximum at approximately zero degrees i.e. straight ahead. The remaining mismatches and deviations between the channel gains are tolerably small and can be further reduced by a constant scaling factor for each channel, if necessary.

Accordingly, if the array is made up of homogenous antennas, it can be supposed that all elements of one steering vector have the same amplitude relative to each other. Furthermore, the ML-based angle estimation can be conducted solely with normalized steering vectors, i.e. it is ensured that $\|\boldsymbol{a}(\theta)\|^2 = 1$ for any angle θ. In this case the complex multiplications in (3.5) can be substituted by vector rotations in the complex plane. The elements of $\boldsymbol{x}$ are simply rotated according to the values of the steering vector before they are summed up. This particular property can be exploited efficiently by the underlying processing architecture which is lined out in the following paragraph.

Being less demanding than an arbitrary multiplication, the rotation of a complex value can be implemented efficiently by the well-known Cordic algorithm, which was already introduced back in 1959 [118]. It belongs to the class of shift and add algorithms which converges iteratively to the desired value. Various transcendental numbers, like trigonometric, hyperbolic or logarithmic functions can be approximated by the Cordic algorithm. The accuracy is mainly determined by the number of iterations and can be adapted to the application's requirements. Furthermore, the computation can be easily pipelined, which is advantageous if high data rates are desired.

The working principle of the Cordic algorithm is to efficiently compute an arbitrary vector rotation. Therefore, the desired rotation angle Γ is not attained in a single step, but it gets approximated by multiple smaller and consecutive steps or so called micro-rotations by the angle γ_b. The step sizes of the smaller rotations are chosen so that they can be realized solely by simple additions and bit shifts, which results in a very resource efficient implementation. With each iteration, the step size is reduced and the deviation from the final target angle decreases. This concept is illustrated with the help of the following equation, which expresses the rotation of a value $a + ib$ by the angle γ_b in the complex plane.

$$\begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} \cos \gamma_b & -\sin \gamma_b \\ \sin \gamma_b & \cos \gamma_b \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \cos \gamma_b \begin{bmatrix} 1 & -\tan \gamma_b \\ \tan \gamma_b & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \tag{3.8}$$

The matrix containing the sine and cosine terms is often referred to as rotation matrix. If the rotation angle γ_b can be chosen freely so that $\tan \gamma_b = 2^{-b}$ is always true, then all multiplications with the rotation matrix are reduced to a bit shift by b and a consecutive addition. The absolute value of the rotation steps are thus predetermined and only the sign, i.e. the rotation direction can be changed. This variant is also referred to as Radix-2 Cordic and is used throughout the rest of this work.

Different sequences of the directions of the individual micro-rotations produce different final rotation angles. A corresponding coefficient $\sigma_b \in \{-1, 1\}$ is used to express these rotation directions in order to match the desired overall rotation angle Γ. Finally, the approximation of Γ with multiple smaller rotations can be described by the following equation:

$$\Gamma = \sum_{b=1}^{B} \sigma_b \gamma_b \tag{3.9}$$

In general, the determination of the coefficients σ_b as well as the compensation of the constant "$\cos \gamma_b$" scaling factor, which can be seen in (3.8), is not trivial and requires additional calculations. However, in this particular application several simplifications to the original Cordic algorithm can be applied which all together help to save a considerable amount of resources:

- Each element of the measured vector $\boldsymbol{x}$ has to be rotated by a constant angle which is given by the elements of the steering vectors $\boldsymbol{a}(\theta)$. Hence, the Cordic coefficients σ_b can be precomputed offline, and the usually required module which determines the direction of each micro-rotation on-line can be omitted.

- The final maximum search can operate on scaled values without any difficulty. As long as the scaling factor remains the same for all values, which is the case for the Radix-2 Cordic variant, the location of the maximum also stays the same. Thus, the commonly used "$\cos \gamma_b$" amplitude com-

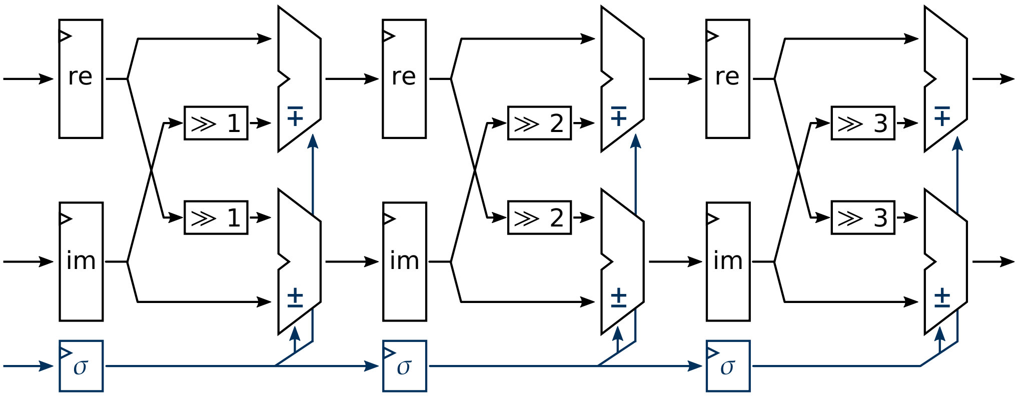


Figure 3.21: Fully pipelined Cordic implementation with three stages. The coefficients σ which control the rotation direction are marked in blue.

pensation at the end of the Cordic algorithm is not necessary and can be skipped.

- Due to the high throughput requirements, a fully pipelined Cordic implementation is appropriate for this application. This allows to implement the normally required barrel shifters as hardwired bit shifts, which reduces the logic resource usage further.

The realized Cordic implementation for this work is illustrated in Fig. 3.21 where the shown pipeline consists of three stages. The accuracy and thus the number of stages, which also corresponds to the word length B, can be configured individually. The σ_b values are supplied from a small RAM which stores the antenna diagram in the form of precomputed Cordic coefficients. Interestingly, this different representation of the antenna pattern in the form of Cordic coefficients only requires half of the memory space compared to the complex values which are required for the multiplier-based version. The reason for the savings is that the Cordic coefficients do not carry any needless amplitude information.

Each stage of the Cordic pipeline contains two B-bit adders which are controlled by the coefficient σ_b to either add or subtract the supplied values from the previous stage. The bit shifts applied to the intermediate values increase during the flow through the pipeline, however they are fixed for each stage so that this operation can be hardwired. They are illustrated as boxes in Fig. 3.21

for a better understanding even though they do not consume any additional logic.

For a quantitative comparison of the resource usage between this optimized Cordic variant and the multiplier-based implementation variants, refer to Fig. 4.6 on page 129 in the next chapter. Furthermore, a detailed benchmark of the different variants has been published in [26].

3.3.2 Two target ML estimator

The computational complexity of the two target ML estimator is considerably higher than in the single target case. At first, the evaluation of the ML function for a given parameter set (θ_i, θ_j) requires three scalar products to be computed instead of one. Moreover, the search space is now two-dimensional which is even more severe and mainly dominates the required computational resources. However, several optimizations can help to compute also the two target estimator in real-time.

The following expression is recalled from (2.41) in chapter 2 and has to be solved in the two target case:

$$\begin{bmatrix}\hat{\theta}_i & \hat{\theta}_j\end{bmatrix} = \underset{\theta_i,\theta_j}{\arg\max} \frac{|r_i|^2 + |r_j|^2 - 2\,\mathrm{Re}\left(r_i {r_j}^* \beta_{ij}\right)}{1 - |\beta_{ij}|^2} \tag{3.10}$$

$$\text{where} \quad r_i = \boldsymbol{x}^H \boldsymbol{a}(\theta_i)\,, \quad r_j = \boldsymbol{x}^H \boldsymbol{a}(\theta_j)\,, \quad \beta_{ij} = \boldsymbol{a}(\theta_i)^H \boldsymbol{a}(\theta_j) \tag{3.11}$$

The variables r_i and r_j are the scalar products between the measured signal vector $\boldsymbol{x}$ and the steering vectors $\boldsymbol{a}(\theta_i)$ and $\boldsymbol{a}(\theta_j)$. The complex number β_{ij} is the result of the scalar product between the two steering vectors and is a constant parameter for a given sensor.

If a single target estimation has been executed prior to the two target estimation, then all possible r_i and r_j values have already been computed and can be reused. Furthermore, the constant coefficients β_{ij} can be precomputed and stored in a memory so that no scalar product has to be evaluated at all during the computation. Besides, the coefficient can additionally be stored in the form $\left(1-|\beta_{ij}|^2\right)^{-1}$ so that the rather expensive division operation can be transformed into a multiplication.

Even more importantly, the two-dimensional search space defined by θ_i and

θ_j is fully symmetric. Hence, only half of all possible combinations of i and j have to be evaluated in order to solve (3.10). The symmetry can be easily proved by inspecting the argument of the "arg max" function while the two parameters θ_i and θ_j are swapped.

As a first consequence, the two scalar products r_i and r_j are also swapped whereas β_{ij} becomes complex conjugated, i.e. $\beta_{ji} = \beta_{ij}^*$. Hence, the sum $|r_i|^2 + |r_j|^2$, as well as the difference $1 - |\beta_{ij}|^2$ remains the same and only the term $2\,\mathrm{Re}\,(\,\cdot\,)$ must be further examined. For its argument, the following relation holds: $r_j r_i^* \beta_{ji} = (r_i r_j^* \beta_{ij})^*$. Given that just the real part of the argument matters, the term finally remains unchanged so that the symmetry is established. Consequently, the search area exhibits a triangular shape which complicates an efficient parallelization. This aspect is investigated more in detail later in this section (refer to page 96).

Acceleration of the two target ML function evaluation

The main task during the two target ML estimation consists of evaluating (3.10), where the three scalar products r_i, r_j and β_{ij} are already available, because they have been evaluated prior to this step. The computation of the three summands in the numerator can be parallelized and a total number of nine multipliers is required for the module: Two for the computation of $|r_i|^2$, six for $2\,\mathrm{Re}\,(\,\cdot\,)$ and one for the final multiplication with the reciprocal of $1 - |\beta_{ij}|^2$. The resulting implementation is able to produce one value of the ML function per clock cycle as long as the required r_j values can be loaded and supplied from memory without interruption.

A block diagram of the realization is illustrated in Fig. 3.22. The module is optimized to process a complete row with the index i of the data matrix. At the beginning, the value $|r_i|^2$ is computed and loaded into the respective register. Then, the r_j and β_{ij} values are loaded continuously from the RAMs and are processed by the module. The ML result is afterward used by a further unit which performs the maximum search.

A main concern during the system design process is an adequate balancing of the single target and two target estimation modules, because they need to interact tightly. The intermediate result of the single target estimator needs to be kept available until the two target estimation process has been finished. Due to the two-dimensional search space which is considerably larger than in

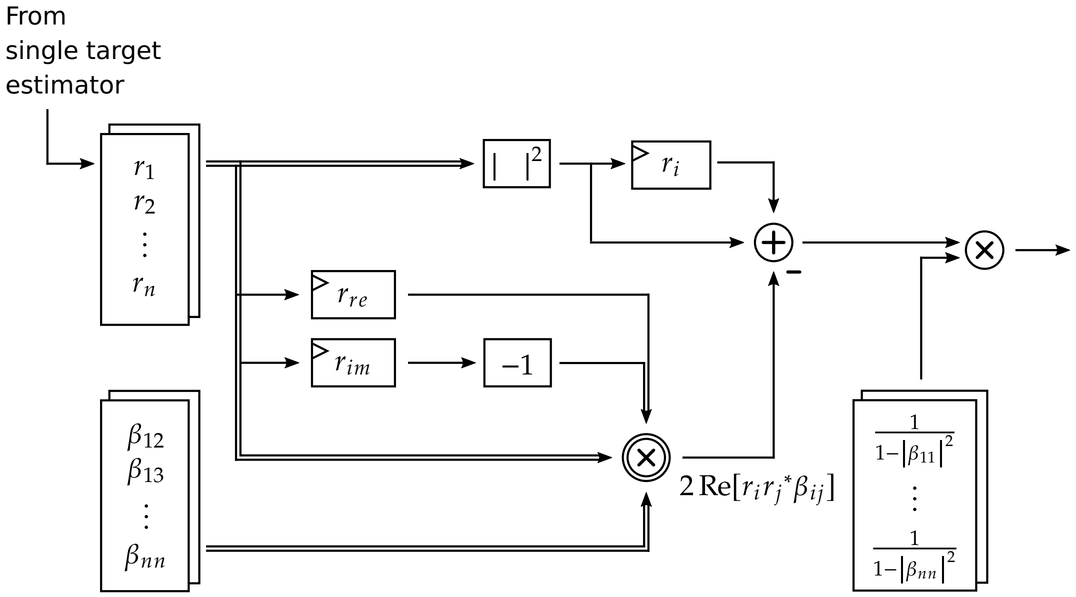

From single target estimator
r_1
r_2
$\vdots$
r_n
$| \; |^2$
r_i
r_{re}
r_{im}
-1
β_{12}
β_{13}
$\vdots$
β_{nn}
$-$
$2\,\mathrm{Re}[r_i r_j^* \beta_{ij}]$
$\frac{1}{1-|\beta_{11}|^2}$
$\vdots$
$\frac{1}{1-|\beta_{nn}|^2}$

Figure 3.22: Computation of the ML function inside the two target estimator module.

the single target case, multiple processing elements (PEs) according to Fig. 3.22 which evaluate the two target ML function in parallel are required. Below, the parameter K is used to hold the number of parallel PEs.

A first possibility for a simple parallelization scheme is depicted in Fig. 3.23 for $K = 3$. In this processing pattern, all lanes share the same value of r_j in order to minimize the RAM load operations. However, due to the triangular shape of the data structure, the lanes have to wait several cycles at the start of each row which decreases the overall efficiency. The waiting cycles are marked in olive green and several PEs are idle and do not contribute to the result at this time. Taking the first cycle, only PE_1 is active and the remaining PEs have to wait until they get a suitable value for r_j. Only from the beginning of cycle 3, all PEs are busy and the accelerator runs at maximum efficiency.

Even though the overhead in terms of waiting cycles was small and seemed neglectable in the previous example, a significant decrease in efficiency can be observed for certain scheduling configurations. Fig. 3.24 illustrates the utilization ratio as a function of the number of steering vectors and for different parallelization degrees K. It can be observed that the more modules are employed in parallel, the lower the overall efficiency of the accelerator will be. This problem can be tackled with an advanced scheduling scheme which is presented in the next subsection.

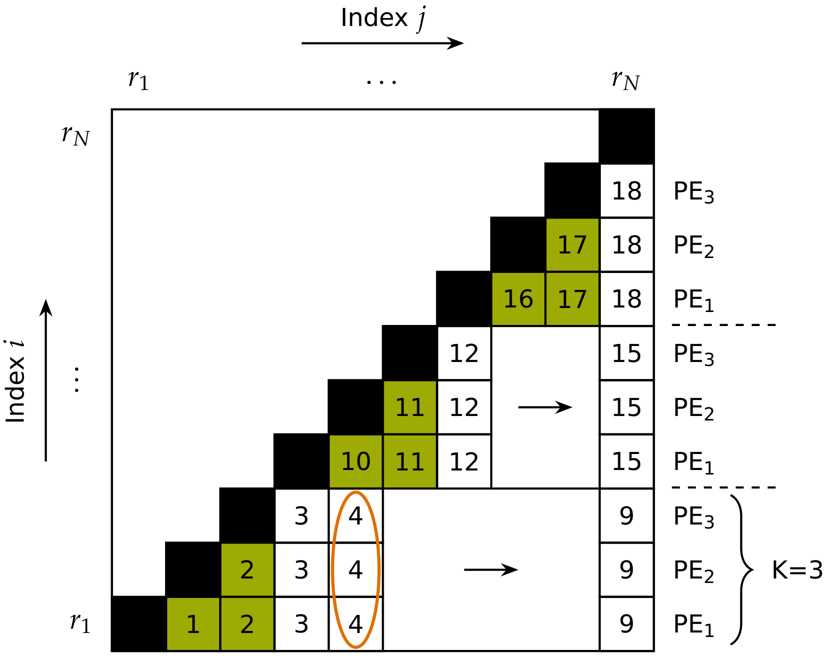


Figure 3.23: Basic parallelization scheme for triangular data structures. In this example, three processing elements (PE 1–3) are used in parallel, i.e. $K = 3$, which is also indicated by the orange oval. The numbers inside the cells designate the processing cycle and the green cells mark certain cycles, in which less than K processing elements are active.

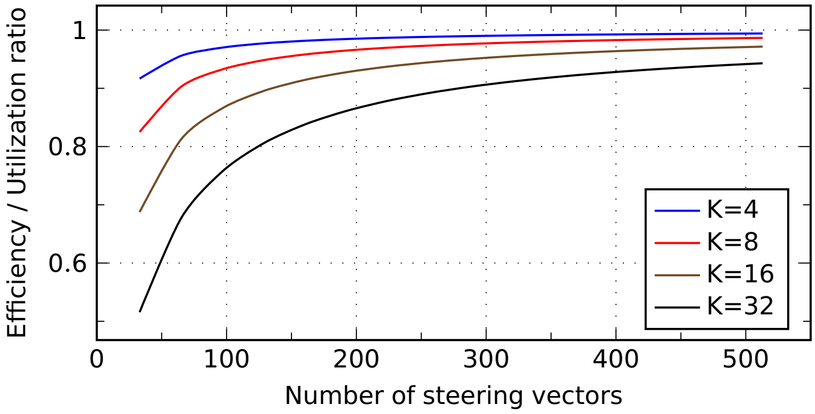


Figure 3.24: Utilization ratio of the accelerator for several degrees of parallelism (indicated by K) when using the basic parallelization scheme from Fig. 3.23.

Efficient scheduling scheme for symmetrical data structures

A more efficient scheduling scheme has been developed in this thesis, which solves the problem of waiting cycles at the beginning of a new row. It is applicable not only to the evaluation of a two-dimensional ML function but rather to an evaluation of a triangular data structure in general. In many practical problems, a symmetry in the data can be observed and only half of the values is of interest, while the indices are still organized in a matrix representation. Then, the idea is to process the data in a different form, which helps to distribute the load equally across all processing elements (PEs). The PE which had to process the longest row in the current pass is scheduled to process the shortest row in the next pass. Consequently, the average load of all PEs becomes equal even though each row of the triangular matrix has a different length.

The proposed processing scheme is depicted in Fig. 3.25 and again each matrix element is marked with the corresponding processing cycle. It can be observed that all PEs operate on different values of r_j, which is remarkable because it doesn't seem to be efficient in terms of RAM load operations. However, the offset between adjacent PEs is only one element so that the required values can be buffered inside a register or a cache memory close to the PEs. The actual RAM load operations are only performed by the first PE while the remaining PEs can reuse the r_j values from the PE with a lower index.

Furthermore, the processing direction is changed after a line has been completely processed, i.e. the index j is increased up to N and then decreased again. This is necessary to maintain the direction in which the elements r_j are passed between the PEs, i.e. even after the processing direction has been changed, the load operations are still performed by PE_1. The two processing directions are referred to as *forward* if the index j is increasing and *backward* if j is decreasing. The transition from forward to backward takes K processing cycles, i.e. each cycle one PE changes its direction. In contrast, the transition from backward to forward is executed in a single cycle, i.e. all PEs change their processing direction at the same time.

In a dedicated hardware architecture, the PEs which compute the result of the ML estimation can be cascaded in a chain of PEs which operate according to the optimized scheduling scheme. The cyclic transfer of the r_j values can be realized with the help of dedicated point to point connections between two

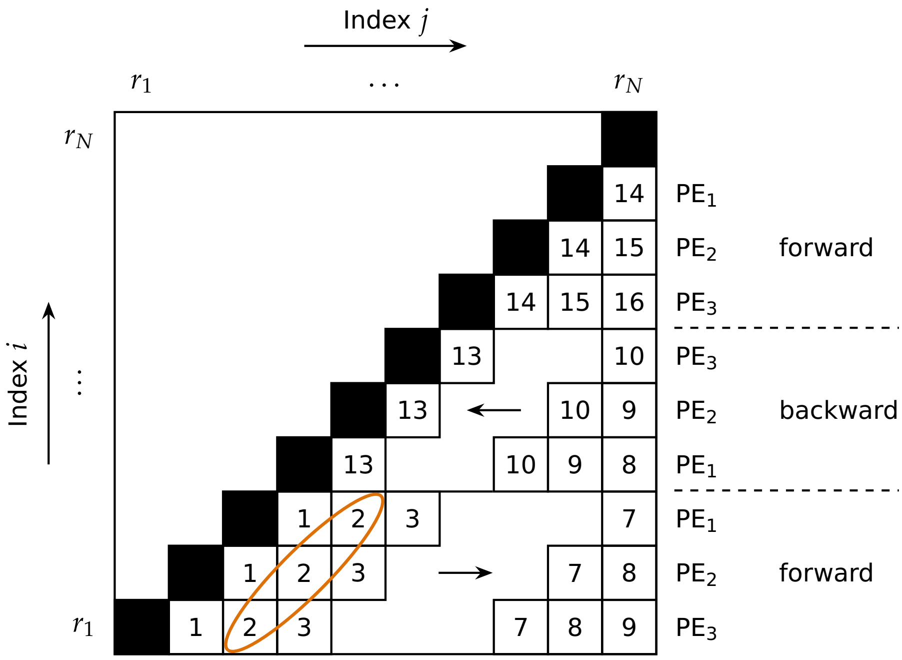


Figure 3.25: Optimized scheduling scheme for triangular data structures. The K processing elements (PEs) are always active and no waiting cycles occur. Again, the numbers inside the cells designate the processing cycle.

modules. An exemplary realization of such a PE is given in Fig. 3.26, where a first interface (*in*) is used to receive the next value r_j and a second interface (*out*) is used to connect to the next PE in the chain. In addition a third interface (*result*) is installed which provides the output of the ML function at the current index.

Two registers are implemented which are used to store the value r_i which remains constant for one row. The first register (*r-next*) directly connects to the input interface and accepts the next value r_i for the subsequent row, while the second register (*r-i*) provides the current value r_i to the arithmetic unit and protects the value against overwriting. Finally, a third register (*r-dl*) can accept and buffer the value which is destined for the downstream PE. Two multiplexers are used which facilitate the transfer of the correct r_i and r_j values at the turnaround of the processing direction.

Besides, the physical realization is simplified because large portions of the datapath only require point to point connections between two modules. A large fan-out of the data path can be avoided even if the number of PEs is high.

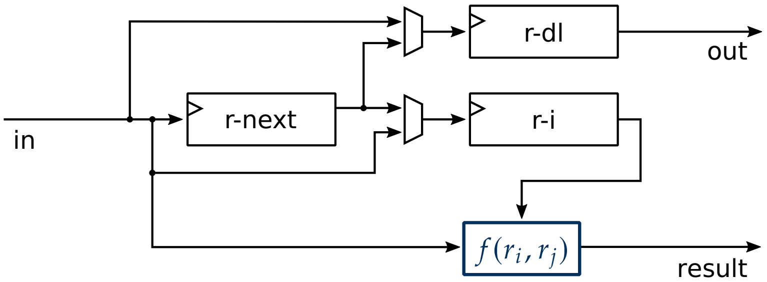


Figure 3.26: Architecture of a single processing element (PE) for the two target estimator. The actual ML function evaluation occurs in the block marked with $f(r_i, r_j)$ which corresponds to Fig. 3.22.

3.4 Summary

An efficient implementation of the processing tasks according to the general overview from Fig. 3.1 was investigated in this chapter. The final system architecture which has been elaborated in this work is shown in Fig. 3.27. All necessary signal processing steps are included in order to get a list of detections along with their coordinates in space out of the raw input data. Furthermore, the algorithms are specifically adapted and optimized for future automotive applications. Important aspects such as data throughput, numeric accuracy of fixed-point implementations, resource usage or speedup compared to other implementations were presented for each module.

In Tab. 3.1, a first estimation of the resource consumption is presented in order to give the reader an impression of the overall system complexity. For this early assessment, prior to the actual design space exploration which follows in the next chapter, a typical parametrization for a high resolution MIMO radar system has been applied.

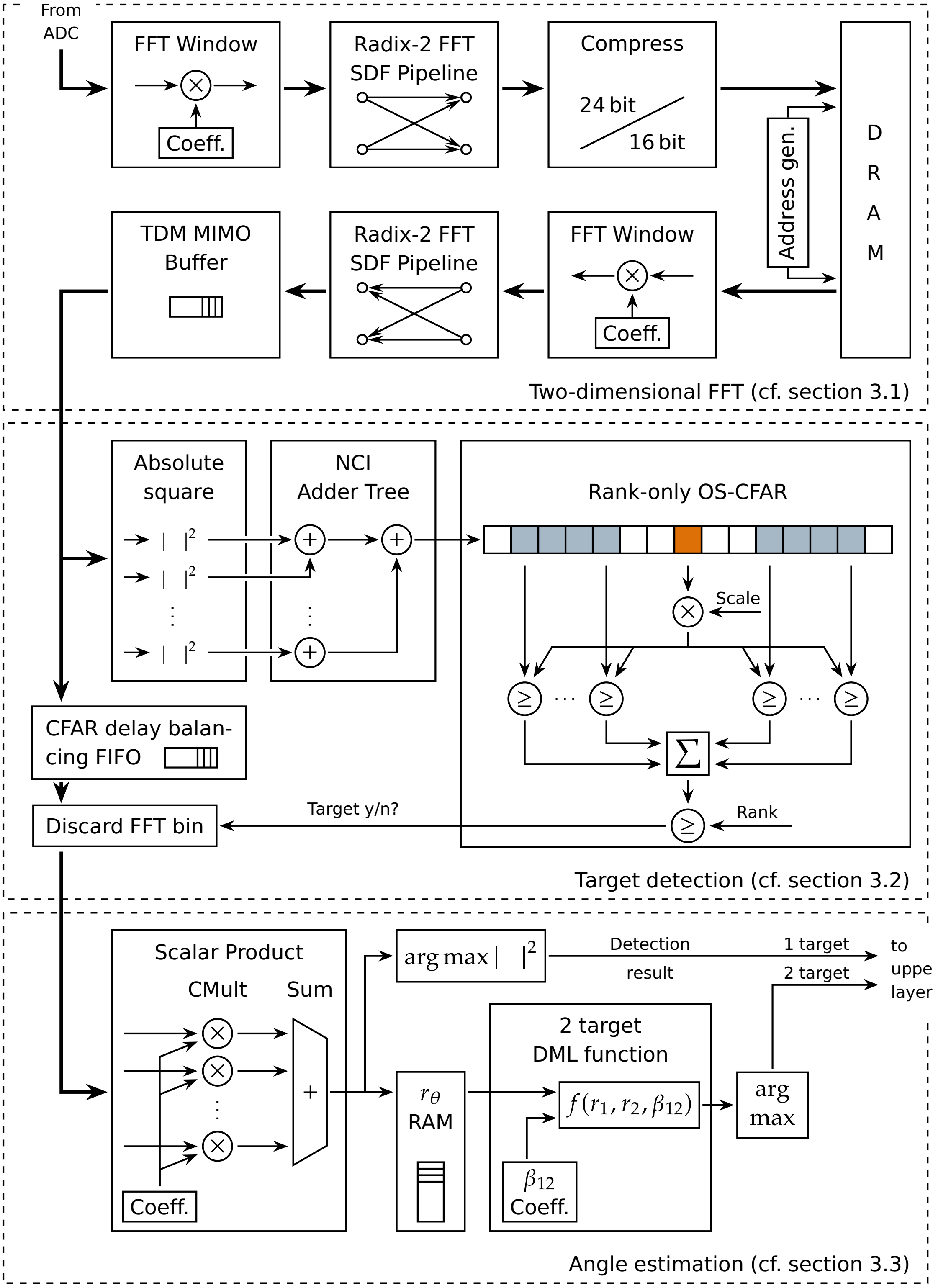


Figure 3.27: Signal processing architecture. The two-dimensional FFT evaluation operates in time-multiplex and processes one TX antenna at a time. After the TDM MIMO buffer, all virtual channels are processed in parallel.

Module	LUTs	Registers	BRAM 36kb	DSPs	Area [mm^2]
Range processing	77640	129936	209	336	53.50
FFT window	113	823	1	16	0.59
Radix-2 FFT (SDF pipeline)	76152	126723	208	320	51.91
Compress (dynamic scaling)	1375	2390	0	0	0.99
Velocity processing	79392	126410	473	352	70.97
FFT window	119	1111	1	32	0.88
Radix-2 FFT (SDF pipeline)	77553	124915	296	320	58.38
MIMO TDM Buffer	1720	384	176	0	11.69
Target detection	18111	23412	0	259	14.46
Absolute square	2816	4992	0	256	5.98
Non-coherent integration (NCI)	2829	2892	0	0	1.00
Rank-only OS-CFAR	3328	6595	0	3	2.27
FIFO buffer	9138	8933	0	0	5.19
Angle estimation	18453	28405	91	620	27.32
DML 1target (DSP-3M)	12177	17490	43	392	16.64
DML 2target (16 Lanes)	6276	10915	48	228	10.68
Ethernet interface	1824	1485	0	0	1.07
ADC interface	16023	16345	16	0	8.34
Memory controller	11261	8463	0	0	4.87
Interconnect & bus infrastructure	3476	6201	46	1	5.15
Control logic & configuration	563	619	0	2	0.50
Total	226743	341276	835	1570	186.22

Table 3.1: Exemplary system resource usage for a typical parametrization with 4 transmit and 16 receive channels.

Finally, the major findings and results of this chapter are summarized again in the following list, where a reference to the corresponding page is given for each point.

Two-dimensional FFT

- For increasing FFT lengths, also a wider word length is appropriate. Each time the transform length is doubled, the word length should grow by half a bit. (→ p. 61)
- The employed rounding scheme matters and it can introduce artifacts into the frequency spectrum. For the Range FFT, an unbiased rounding mode should be used, e.g. convergent rounding. (→ p. 64)

- It can be beneficial to use different word lengths for the Range FFT and the Velocity FFT. This may be especially considered in situations where the data amount in between the two transforms shall be minimized. (→ p. 66)
- The dynamic range of the fixed-point values is rarely fully exploited for larger target distances. A dynamic scaling can help to improve the PSNR without generating additional resource usage. (→ p. 69)
- If an SDRAM is used for the matrix transformation, an optimized addressing scheme is very important in order to fully utilize the available interface bandwidth. (→ p. 73)

Target detection

- The usage of a rank-only CFAR implementation scheme allows the employment of large window sizes with 128 cells and more. (→ p. 81)
- NCI improves the detection performance and helps to save resources at the same time. (→ p. 84)
- For growing channel numbers, i.e. 32 channels and more, the raw data buffer which compensates for the CFAR processing latency predominates the resource usage. (→ p. 84)

Angle estimation

- The usage of Cordic processors which apply the scalar product can be advantageous in terms of resource usage. Furthermore the coefficient storage requirement can be cut into half. (→ p. 88)
- The intermediate results of the single target estimator should be cached inside the angle estimation module, because they can be reused later by the two target estimator. (→ p. 92)
- An optimized scheduling scheme for symmetric data structures has been developed which allows to improve the utilization ratio of the two target estimator. (→ p. 96)

4

Design Space Exploration

One objective of this work is to investigate the available design space in a systematic manner, in order to set up a useful database of implementation results. Such a dataset can be very helpful in an early product concept phase, because it enables a quick and accurate estimation of important figures like cost, performance and efficiency. Most notably, this is possible before the actual development phase has even begun.

In the following section, the employed design space exploration method along with the used tools is introduced and explained. Subsequently, the results are presented which have been obtained for the target detection module (cf. section 4.2) as well as for the single and two target ML estimator module (cf. section 4.3 and 4.4 respectively).

4.1 Methodology

A model-based design space exploration approach similar to [5] is employed, which can be executed in two major steps. Firstly, the implementation results of different design variants, each with its own parametrization, are measured and collected. Secondly, an appropriate model function is selected which shall describe the major relations between the design parameters as accurately as possible. The selection can be based on observation of the available measurement data as well as on prior knowledge of the module's architecture. The coefficients of the model function are determined by a multiple linear regression which is explained in section 4.1.3.

Once an appropriate model function has been established, it becomes possible to estimate the outcome of any design parameter combination. Even those realizations for which a direct measurement was never performed are now covered by the model function. This is a big advantage, because usually it is not possible to explore the huge amount of all possible parameter combinations.

4.1.1 Measured quantities in the design space

For the evaluation and comparison of different realizations in the design space, numerous metrics can be measured and used for a quantitative analysis. In this work, the following variables were collected for each individual variant of the module under investigation.

Area

The consumed silicon area of the module provides a very good cost estimate and is expressed in mm^2. As the silicon area is not explicitly reported by the FPGA tools, it has to be estimated based on other values like the number of incorporated logic blocks, e.g. the number of DSP slices. The total equivalent silicon area can then be computed if the area size of a single building block is known. For this work, such area consumption factors were deduced for the used FPGA devices and for every different logic block type. The coefficients along with their derivation are included in appendix A.2. The exact relationship to the actual unit costs of the integrated circuit (IC) can be quite complex to deduce and would require more knowledge about the used technology node, wafer size and yield [119], which is out of scope for this work.

In total, three different logic elements are available within the used Virtex 7 FPGA. They are mentioned and briefly introduced in the following paragraph. For a more detailed explanation of the underlying functionality and architecture, refer to standard literature such as [120, 121] or to the respective user guides from Xilinx [122–124].

Logic slices Most universal and fine-grained building blocks, consisting mainly of multiple lookup tables (LUTs) and flip-flops. They can realize arbitrary logic functions.

DSP slices These elements are meant for implementing digital signal processing (DSP) functionality. One slice usually contains a hard-wired multiplier along with supporting logic, such as adders and flip-flops.

Block RAM (BRAM) Embedded memory blocks with a very low latency and a fixed size (e.g. 18 kbit per block for the used FPGA).

The major advantage of using the total equivalent silicon area of those building blocks is a much better comparability. Frequently, different logic block types can be used to realize the same functionality which makes it hardly feasible to compare different FPGA implementations on the basis of only a single resource type. Instead, the resource usage in its entirety, i.e. across all types of logic elements, has to be taken into account. Thus, the usage of the equivalent silicon area seems reasonable because it provides a weighted sum of all logic elements according to their actual silicon area consumption on the FPGA device.

However, it has to be considered that the estimation of a silicon area by the translation of multiple other values, like the number of used building blocks, is a complex task and that a certain risk of misinterpretation exists. It has been mentioned above, that the exact coefficients for the calculation were not available from the manufacturer. They were obtained by other means only and accordingly, they can be a potential source of error. Furthermore, the estimated area cannot account for architecture related effects and constraints like the consumption of routing resources, the availability of clocking infrastructure or the granularity of the individual logic elements on the FPGA. It may happen that the same design can be mapped more efficiently to other FPGA types of the same family which would in turn result in differing estimates of the silicon area.

Power

The power dissipation of a device is a crucial design parameter because it has an immediate influence on the required supply current and the generated heat. These are important design parameters affecting the system's power supply, housing and cooling solutions. The actual power efficiency, which is of major importance for battery powered devices, is however not only related to the power but rather on the required energy for a certain amount of computational load. The efficiency was also evaluated in this work and the unit is introduced later in this section.

The power consumption is expressed in Watts (W) and is a direct consequence of the operating voltage and current of the device. Instead of measuring the actual supply current of a physical realization, an estimated value is used for this work, which is provided by the FPGA tools. Even though the accuracy of an estimation is worse than that of a measurement, the effort to obtain the value is low enough so that a systematic exploration of many different implementations is possible. For the estimation, a constant workload near the module's maximum data throughput was assumed.

The comparison of different implementations in terms of their power consumption was always performed at the same clock frequency. This is inevitable because a change in the operating frequency will directly alter the power values so that a fair benchmark is not possible. As a compromise, a clock frequency of 200 MHz was chosen for all power estimations which will be below the maximum for certain module variants, but which is nevertheless quite fast for most FPGA implementations.

In related work, a distinction between *static* and *dynamic* power consumption is often made [125, 126]. Static power is dissipated even if the hardware is powered up but currently idle. It results mainly from transistor bias and leakage currents. In contrast, any switching activity and dynamic charging of capacitances inside the circuit is treated as dynamic power consumption. In general, the dynamic component is strongly related to the clock frequency and the computational load. The static component is rather dependent on the technology process and the operating temperature.

In this work, always the total dissipated power is stated, which is simply the sum of both components. Even though the used tools are able to report the static and the dynamic power consumption separately, only the total value is taken into account. The goal in this work is primarily to provide a realistic assessment of the expectable power dissipation for certain configurations, and not to analyze and optimize the individual components. However, the data was recorded separately throughout the whole design space and would be available for further studies.

Performance

The performance in terms of data throughput per time interval is very important for real-time systems. It is frequently measured in million instructions per second (MIPS). For a general purpose processor, this unit of measure is a meaningful metric because a single instruction is relatively clear determined by the processor's available machine instruction set. Though, for dedicated hardware accelerators, such "atomic" operations can hardly be identified because the arithmetic units, buffer memories and logic circuits are strongly adapted to a specific application. A strong level of parallelism often prevails and multiple complex operations can be performed in a single clock cycle.

Therefore, two specific metrics which are only meaningful in the context of radar signal processing will be used in this work:

Frequency cells per second This unit expresses the number of frequency bins which can be handled by the module. It is primarily useful to describe the performance of the target detection accelerator, because this blocks always operates on the raw spectrum cells. In this way, a dependency to the used modulation can be avoided, because the workload is directly measured in the input variable of the module.

Detections per second In the same manner as for the target detection accelerator above, the throughput of the angle estimation module can be measured in its input data type. Hence, the performance is denoted in detections per second which is usually one or two orders of magnitude lower than the number of frequency cells per second, due to the removed noise components (cf. also section 5.3.2).

The performance values which will be presented in the following paragraphs are all based on the maximal achievable operating clock frequency f_{max}. Of course, if a lower data throughput is desired, the clock rate can be lowered at the benefit of a reduced power consumption.

Accuracy

The computational accuracy is mainly important if fixed-point arithmetic is used and if the available word length is limited. Then, the mean error which is induced by the imperfect calculation and rounding operations can be expressed as *peak signal-to-noise ratio (PSNR)*. This unit expresses the deviation of the erroneous result from a very accurate double-precision result, measured on a logarithmic scale. For a detailed introduction of the PSNR unit, please refer to section 3.1.2.

Area efficiency

A cost-driven evaluation of different implementation variants cannot be based on its silicon area consumption alone. For instance, a common method to reduce the runtime of a certain application is to replicate certain logic blocks of a circuit, so that a parallelization can be achieved. Certainly, the silicon area of the whole module will increase, however at the additional benefit of a shorter runtime or equivalently, a higher data throughput. Hence, the ratio between area and throughput, or likewise the area time product, are good measures to express the efficiency in terms of area consumption of an implementation.

Power efficiency

In the same manner as for the silicon area consumption, the power dissipation can also be related to the performance of the module. The result is a metric which expresses the required power for a certain data throughput. Interestingly, the relation to time is canceling itself, so that the unit of measurement is equivalent to the required energy for a certain number of operations. Both versions, i.e. power per data throughput and energy per result, are an adequate and meaningful quantity to describe a module's power efficiency.

Even though a value for the power efficiency was obtained for each implementation in this work, it is not used for a further analysis. For an in-depth investigation, the design space was not traversed sufficiently because only a single operating frequency was used for each implementation. Especially when analyzing the power efficiency of certain realizations, a wide range of operating frequencies should be considered because the frequency has a strong influence on both, data throughput and power consumption.

4.1.2 Design space exploration procedure

For this work, a custom framework was developed which helps to automate the following, recurring steps:

1. Generate a list of parameter sets, which describe the discrete points in the design space to be explored. The parameters are specific for every different module.

2. Insert a single parameter set into the design under investigation.

3. Launch the corresponding toolchain which synthesizes and implements the specific design on a predefined FPGA target device.
4. Parse and collect the desired results from the log and report files. This can be data like logic utilization, maximum clock rate or power consumption, just to mention a few.
5. Repeat steps 2 – 4 until the whole list of parameters has been traversed.

Due to the fully automated exploration flow, a manual insertion of parameters as well as extraction of the results is not required by the user. This helps to avoid data errors and it speeds up the whole exploration process. The realization is based on a custom-made Python script called `do_dse.py`, which was specifically developed for this work. Basically, it executes the steps mentioned above on multiple processor cores simultaneously. A flow chart of the program is depicted in Fig. 4.1.

The starting point of every design is a comprehensive Vivado[1] project, herein named `dse_project.zip`, including all necessary modules as VHDL source files. In addition, a specific template file called `top_level.mako` is required which is thought to be updated with different parameter sets later during the exploration. Upon each iteration of the main loop, a different parameter set is inserted by the DSE framework into the template file. Only afterwards, the actual FPGA tools are started ("Run Vivado") and the design is implemented for the specific point in the design space. After the tools have completed their execution, all important result and log files are saved and parsed for later analysis.

Determination of the maximum operating frequency

In the usual design flow, the user gives the tools an indication about the intended clock frequency f of the module. Based on this information, the synthesis, placer and routing engines will chose an optimized realization which is capable to fulfill all the given timing constraints. In the final timing report, all signal paths of the completed design are listed along with the achieved setup and hold margins. In principle, the maximum operating frequency could be directly deduced from this information by simply calculating $f_{\max}$ so that the

---

[1] Vivado is the brand name for the FPGA tools from Xilinx.

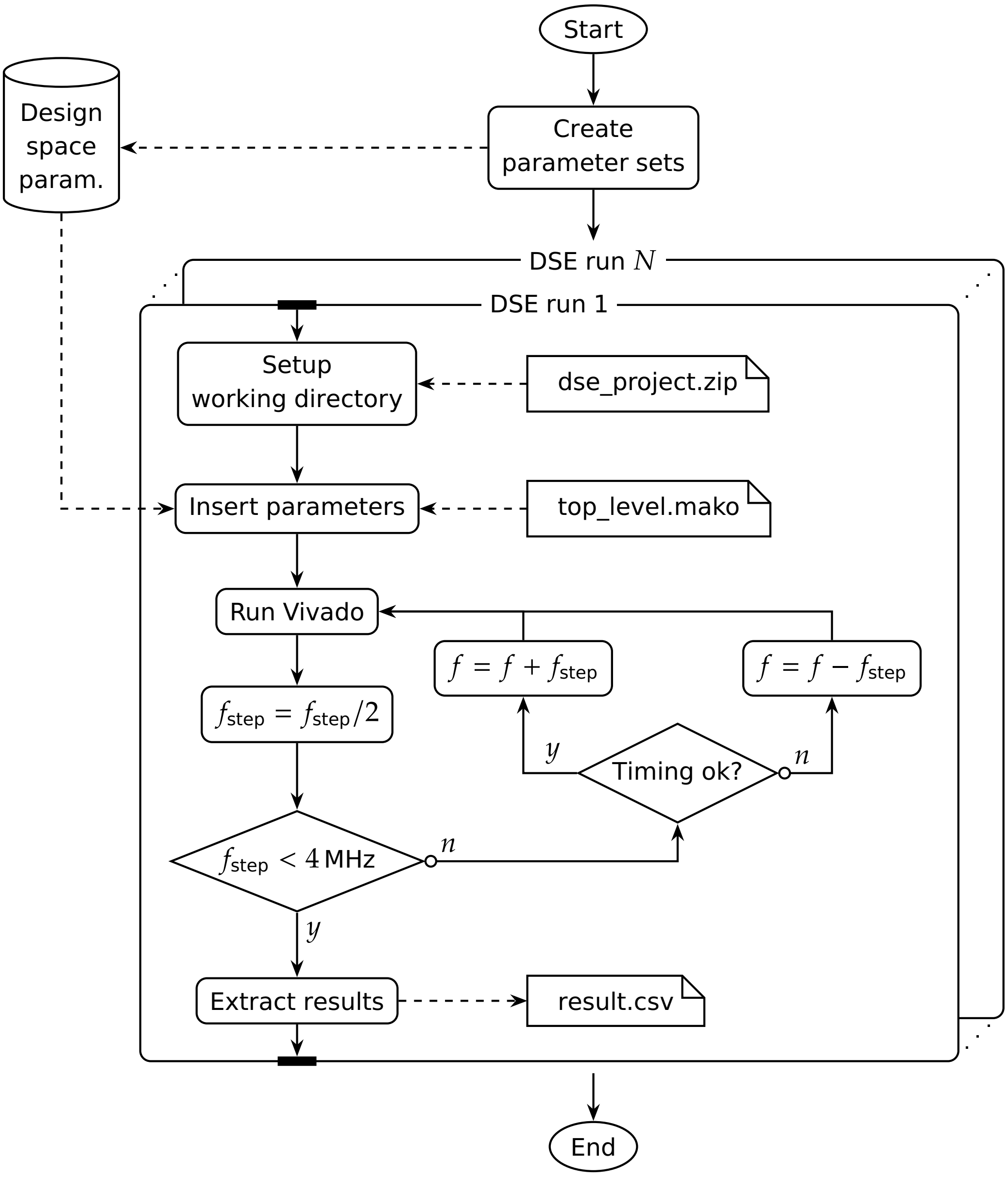


Figure 4.1: Design space exploration flow chart.

margins approach zero. However, such a value is very often far below the possible maximum which can be achieved for a certain module on a specific device. The reason is that the tools have multiple optimization targets and they will primarily try to reduce logic usage while fulfilling all other constraints. Therefore, the timing behavior of a specific realization is just optimized as much as required for a specified target frequency f.

To overcome this limitation, the implementation flow was slightly adapted.

Multiple iterative runs are executed, each with its own target frequency f instead of one unique run. After the completion of one iteration, the design status is checked whether the timing was achieved or not. On success, the frequency for the next run will be increased by a certain frequency step f_{step} whereas on failure it will be decreased. The frequency step is divided by two after each iteration until a certain accuracy has been reached. In this work, the process is stopped at an increment of 4 MHz.

FPGA related characteristics

In order to obtain meaningful results, some particular configuration settings of the Xilinx toolchain were adapted and a specific implementation flow was chosen.

As a target device, the Virtex 7 FPGA `xc7vx485t-2ffg1761c` was chosen for all runs, because the subsequent verification (cf. chapter 5) with real hardware is also based on this part. All reported results in this work were obtained after the *place and route* process of the parameterized design was completed. This means, that all values are connected to an actual realization which is ready to be used. Furthermore, all signals were checked against the predefined timing constraints, i.e. the operational capability of the implementation is always assured.

The parameter `-mode out_of_context` was given to the synthesis compiler, which instructs the tool that the design is meant to be implemented without interaction of the external world. This is required because per default, a randomly chosen device pin is used for each unconnected input or output signal. Certainly, this is not the intended behavior, because the investigated modules are designed with rather wide interfaces with a huge signal count aiming for a further integration into a system on chip (SoC). If the tools try to bring out all signals of the FPGA, the routing and timing of major parts would suffer from this behavior. Hence the results would be distorted, especially the achieved maximum operating frequency f_{max}.

Furthermore, a global clock buffer was added for every clock signal present in the design. Consequently, the dedicated clock routing resources of the FPGA were utilized and the timing of the sequential logic was improved.

4.1.3 Fitting of a model function

Based on the gathered data and the measured quantities, an appropriate model function will be selected and fitted as accurately as possible to the observed variables. Later, such a model can be used to estimate the outcome of any possible design variant even if the actual implementation result for a certain parameter combination is not on hand.

In a first step, a mathematical expression is set up which shall describe the relation between an observed, *dependent* variable y (for example the silicon area), and multiple *independent* variables x_j which could be the word length, the number of channels and so on. In combination with a set of constant parameters β_j where $j = 1 \mathinner{.\,.} p$ and where p is simply the number of independent variables, the outcome is a linear function (4.1). The additional parameter β_0 is known as the *intercept term* and models a constant offset. A random error is added by the term ϵ, which allows for a deviation of the predicted values $\hat{y}$ from the observations y.

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p \tag{4.1}$$

$$y = \hat{y} + \epsilon \tag{4.2}$$

Even non-linear relations can be modeled with the help of this simple function by using a variable substitution. For instance the quadratic term of a certain design parameter can be added as an additional independent variable x_j to the function. Hence, the approach is adequate to model an arbitrary polynomial.

Once a suitable function has been found, the constant parameters β_j have to be determined in a second step, which is often referred to as curve fitting. In the specific case of a linear function with multiple independent variables according to (4.1), the process is also known as *multiple linear regression*. Due to the linearity of the function in this case, a basic least square fit can be used which simply minimizes the sum of squared residuals, i.e. minimizes $\sum_i \left(y_i - \hat{y}_i\right)^2$ where i is the observation subscript. A necessary condition is that the number of different observations n in the form of (4.2) is larger than the number of independent variables p, so that the system of equations is overdetermined and a unique solution exists.

However, the increase of the model order p should be well considered and handled with care, even if enough observations n are available, because the risk of *overfitting* exists. The term overfitting designates a common problem in statistics [127–129], where higher-order functions are used to describe a dataset whose underlying model structure is in fact of a lower order. Especially when trying to minimize the residual error, one has to be aware that an increased model order can always adapt better to the observed data. Thus, the choice of a large number of independent variables p for the sole purpose of a small residual can be misleading and should be avoided. Typically, a polynomial of a too high degree tends to mimic the measurement noise and will consequently fail to predict new data points which were not part of the original set.

Evaluating the goodness of fit

In reality, several measures can be used to quantify the discrepancy between the model and the observed data. Such measures are often summarized and named by the term *goodness of fit*. They allow a judgment of the accuracy and the correctness of the chosen model.

A straight forward measure is certainly the residual sum of squares (RSS) which is always minimal for a least squares fit and a given model order. Taking this value and dividing it by the number of degrees of freedom ($n - p - 1$) leads to an unbiased estimate of the variance of the random errors ϵ_i, and gets denoted as s^2. Correspondingly, the value s is an estimate of the standard deviation and is often referred to as *regression standard error*:

$$s = \sqrt{\frac{\text{RSS}}{n-p-1}} = \sqrt{\frac{\sum_i \left(y_i - \hat{y}_i\right)^2}{n-p-1}} \tag{4.3}$$

Unlike the RSS, the regression standard error s does not necessarily increase if more data points are used for the regression, which makes it easier to compare the fitting result of datasets with different sample sizes. However, s is a non-normalized value and has to be set into relation with the usual range of values of the dependent variable. For instance, with the information that s is $1\,\text{mm}^2$, a statement about the goodness of fit is not possible yet. It would be a good fit if the dependent variables are all within the range of $100\,\text{mm}^2$, but it would be a rather bad fit if they amount to only $1\,\text{mm}^2$.

Another significant measure is the *coefficient of determination*, most of the times denoted as R^2. It expresses how well the variance in the measured data can be explained by the model function and ranges typically from 0 to 1. It is defined by the following expression:

$$R^2 = \frac{\sum_i \left(\hat{y}_i - \bar{y}\right)^2}{\sum_i \left(y_i - \bar{y}\right)^2} \tag{4.4}$$

where $\bar{y}$ is the mean of all observations. The closer the value of R^2 is to one, the better the predicted data fits the observations.

For a more comprehensive survey of regression analysis and least square fitting methods, including detailed derivations and more background information, the reader can refer to established standard literature, such as [130, 131]. The purpose of this section is rather to introduce the fitting approach which was used in this work and thus some previous knowledge of statistics and curve fitting methods is presupposed.

4.2 Target detection module

The first module which is to be investigated with the previously described framework, is the target detection module. The incorporated detection process comprises mainly two steps, NCI and OS-CFAR which are both explained in detail in section 2.2. The architecture and implementation of the target detection module is presented in section 3.2. The design space parameters for this module consist of the following three variables.

Word length (b) Defines the word length of the input and output spectral data. The intermediate bitwidths are derived from this parameter automatically in a manner so that the PSNR is not significantly deteriorated. For instance, no rounding is used for the NCI computation and the bit width grows with each addition.

Number of channels (M) This is the number of (virtual) receiving channels which the accelerator can handle in parallel. It should have a strong influence on the NCI step of the algorithm and nearly no influence on the CFAR processing.

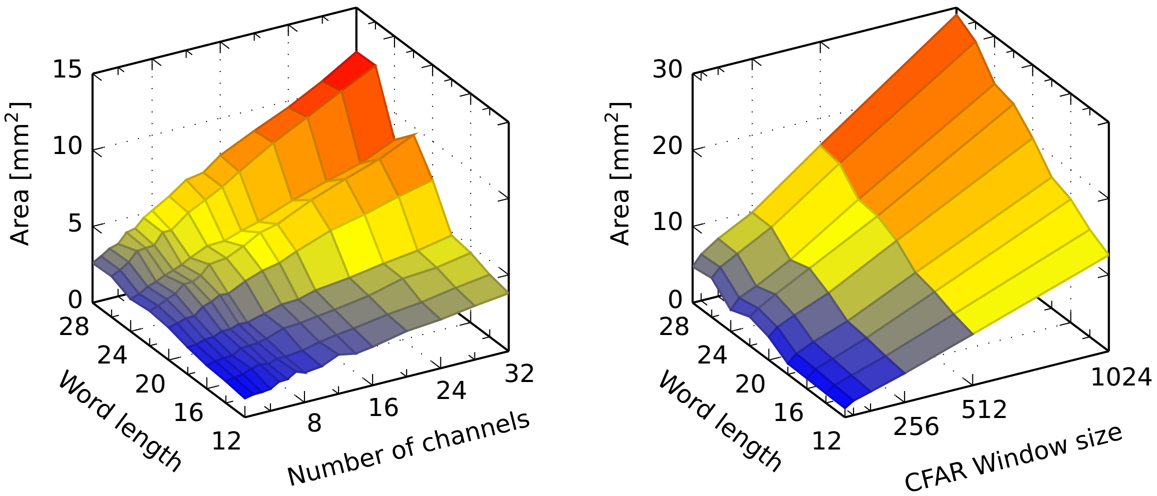
CFAR window size (N) The CFAR window size limits the number of cells which can be used for the noise level estimation. In general, a larger window size causes a larger sorting or rank determination complexity, but also a higher estimation accuracy.

4.2.1 Area

The area consumption depends on the three parameters mentioned above so that a single graphical representation including all relevant points of the design space is difficult to achieve. Thus, two three-dimensional surface plots are used where one parameter is fixed. The diagram does not show the whole design space which has been explored, however it gives the reader an impression about the relationships of the individual parameters. The two upper plots show the observed, raw measurement data. Every node of the visible mesh originates from a real implementation. For the graphical representation, all nodes are simply connected in a straight manner by the surfaces, i.e. no interpolation has been applied. In contrast, the two lower plots are based on the model function (4.5) which is introduced later in this section. The visible mesh is considerably denser for these two plots, because the model function can be evaluated in a fine-grained manner with minimal effort.

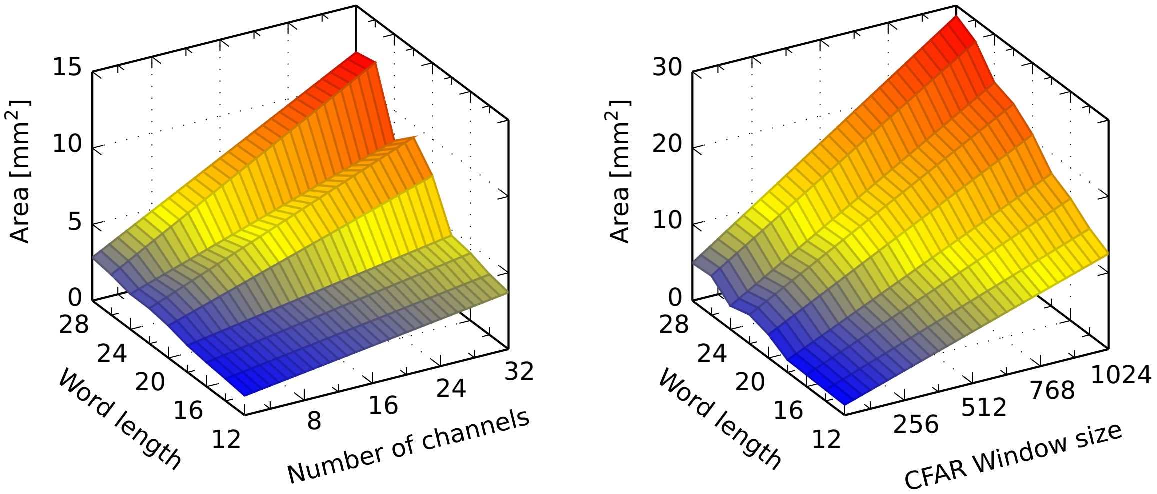
Regarding the dependency on the word length b, a slight linear increase of the area until $b = 18\,\text{bit}$ can be observed in Fig. 4.2. Afterwards, the slope rises considerably because the multipliers for the computation of the absolute square values do not fit into a single DSP slice anymore. Hence, additional logic elements are required which are finally responsible for the rapidly growing area consumption. For a small extension of the word length, for instance from 18 to 22 bit, only general purpose LUTs are added to implement the remaining bits of the multiplication. Once the word size expands and reaches 24 bit, the use of additional DSP slices instead of LUTs becomes favorable, even if their internal word size is much larger than the required difference of 6 bit. The effect can be recognized by the stepped characteristic of the area consumption, which decreases at certain points, e.g. at the transition from 22 to 24 bit or at the transition from 26 to 28 bit. The steps occur exactly at those spots where the additional LUTs are replaced by further DSP slices. The cause of the decline is ultimately a higher area efficiency of the hard-wired multipliers inside the DSP slices. The word lengths at the transitions are a reasonable choice for a practical realization, because a local minimum of the area consumption exists.

The scaling behavior of the module in relation to the window size turns out



(a) Observation
CFAR Window size (N) = 128

(b) Observation
Number of channels (M) = 16



(c) Model function, refer to (4.5)
CFAR Window size (N) = 128

(d) Model function, refer to (4.5)
Number of channels (M) = 16

Figure 4.2: Silicon area consumption of the target detection module on a Virtex 7 FPGA.

to be nearly linear (cf. Fig. 4.2b). It is mainly caused by the N comparators as well as the data buffer equalizing the pipeline delay which can be seen on the bottom of Fig. 3.18 on page 83. The number of channels has a much lower effect on the silicon area variation than the window size. For instance, the area consumption is always within the same order of magnitude when comparing one and 32 channels. The reason for this good scaling behavior is that only the NCI step depends on the number of channels and in contrast, the CFAR processing part is unaffected. Hence, the architecture can be considered as very efficient for large channel numbers and is thus suitable for MIMO systems.

A benchmark with similar FPGA implementations which have been reported in literature was made and its results are summarized in Tab. 4.1. The advantage of the rank-only implementation form is visible among all results, even though a direct comparison of the resource usage values is not possible due to slightly different design parameters. For instance the more than three times lower logic slice consumption of the module from this work in comparison to [84] is expected to originate from the shorter word length and the absence of costly floating-point arithmetic.

Publication	Resource usage	Window size (N)	Word size (b)	Comments
This work	2389 LUTs (6-input) ≅ 1025 Slices	128	16 bit	Rank-only
Bales et. al. [84]	~ 3500 Slices	128	32 bit float	Rank-only
Bales et. al. [84]	~ 6500 Slices	128	32 bit float	Insertion sort, Req. multiple cycles
Magaz et. al. [85]	1823 LUTs (4-input) ≅ 1283 Slices	16	12 bit	K-th maximum search, Req. multiple cycles
Perez-Andrade et. al. [86]	1260 LUTs (4-input) ≅ 2790 Slices	64	12 bit	Insertion sort

Table 4.1: Comparison of the resource usage of different OS-CFAR implementations on an FPGA.

Model function

The following function is used to approximate the consumed silicon area of the target detection module. The word length b is not explicitly stated in (4.5), even though a dependency exists. In fact, the influence of b is modeled by a family of curves (4.5), where the set of parameters β differs for every word length b (see also Tab. 4.2).

$$y_{Area}(M, N) = \beta_M\, M + \beta_N\, N + \beta_{MN}\, M\, N + \beta_0 \tag{4.5}$$

A linear dependency on the CFAR window size N as well as on the number of channels M is included. The origin for these two linear terms is the logic consumption of the comparators which grows with N and the binary adder tree, which grows with M. Furthermore, a combined dependency on both parameters is taken into account by the coefficient β_{MN}. It is caused by the raw data buffer whose size grows with both variables, M and N.

The third parameter, the word length b, provokes a rather irregular and stepped behavior. The root cause for this special characteristic lies in the fixed word length of the embedded multipliers inside the DSP slices, which has already been explained in detail on page 115. The effect is strongly dependent on the used FPGA architecture and the associated software tools, so that its influence on the area consumption can be hardly described by an analytical expression. Hence, the parameter b is not directly included in the function itself. Instead, the model function was fitted independently for each different word length b and all coefficients were determined individually. They are listed in Tab. 4.2 along with the goodness of fit (GOF), which has been introduced in section 4.1.3 on page 113.

From the graphical appearance (cf. Fig. 4.2c and Fig. 4.2d), as well as from the GOF measures, a very high accuracy of the model can be observed. The regression standard error s is far below $1\,\mathrm{mm}^2$, and R^2 is above 0.99 for all investigated word lengths b.

Word size	Coefficients [mm^2]				GOF	
b	β_M	β_N	β_{MN}	β_0	s [mm^2]	R^2
12	0.0551	0.0083	$0.1801 \cdot 10^{-3}$	0.1467	0.279	0.995
14	0.0594	0.0099	$0.1663 \cdot 10^{-3}$	0.1099	0.175	0.998
16	0.0635	0.0111	$0.2144 \cdot 10^{-3}$	0.0764	0.341	0.995
18	0.0670	0.0127	$0.1984 \cdot 10^{-3}$	0.0478	0.252	0.998
20	0.1442	0.0136	$0.2565 \cdot 10^{-3}$	0.2013	0.343	0.997
22	0.1894	0.0150	$0.2525 \cdot 10^{-3}$	0.1656	0.258	0.999
24	0.1479	0.0162	$0.2816 \cdot 10^{-3}$	0.0556	0.382	0.997
26	0.2705	0.0175	$0.2958 \cdot 10^{-3}$	0.0998	0.312	0.999
28	0.2524	0.0187	$0.3221 \cdot 10^{-3}$	0.1394	0.388	0.998

Table 4.2: Coefficients of the model function (4.5) describing the silicon area consumption of the target detection module

4.2.2 Power

The power consumption of the module is shown in Fig. A.1 in appendix A.3.1 with four distinct surface plots, in the same manner as previously conducted for the area consumption. The first two figures are based on the observed data, while the last two figures show the predicted data of a fitted model function (A.15). The coefficients of this function are included in Tab. A.4.

For the power evaluation, the operating frequency was fixed at 200 MHz for every different realization, in order to get rid of any dependency on the clock rate. In general, the characteristics follow the previously shown results of the silicon area, however some differences can be observed. For instance, the local minimum for a word length of 24 bit is less pronounced and the power consumption is almost monotonically increasing with b.

The absolute value of the target detection module's power consumption can be below 1 W as long as the window size doesn't become too large and exceeds 128 cells. Keeping in mind that this module is able to process 200 million frequency cells per second for the used settings, the power efficiency of the rank-only implementation form clearly stands out. This is apparent when comparing the realization with other computing platforms.

Previous work investigated the performance of a CAGO-CFAR implementation on a graphics processing unit (GPU) and compared it to a version running on a central processing unit (CPU) [132]. A window size of 32 cells and single-precision floating-point values were used so that only a rough comparison can be made. A pretty high data throughput of around 2.4 billion cells per second was achieved on the GPU platform in contrast to a throughput of only 80 million cells on the CPU. Even though the GPU implementation performs about twelve times faster than the FPGA module in this work (200 million cells per second), the maximum power consumption of the mentioned GTX 280 GPU platform is specified with 236 W [133]. The power consumption of the CPU is unknown, however it is expected to be above 10 W which is true for most general purpose processors. Both versions, the GPU and the CPU are thus considerably worse in terms of power efficiency, mainly due to the very low power consumption of 1 W of the FPGA module built in this work.

Independent, related work which investigates a rank-only OS-CFAR implementation arrives at a similar result, confirming the power-efficiency of an FPGA-based architecture [84]. With the general framework being identical to the previous example (Multi-core CPU versus GPU, single-precision floating-point, 32 window cells) a performance of 340 million cells per second was achieved on the CPU and more than 1 billion cells per second on the GPU. Furthermore, a very power-efficient FPGA implementation which consumes less than 1 W is also presented in [84]. Finally, the FPGA version is rated as the most efficient due to the distinct higher power consumption of the other architectures (max. 190 W for the CPU and max. 238 W for the GPU).

4.2.3 Performance (Data throughput)

The data throughput of the target detection module is one frequency cell per clock cycle. Furthermore, the module is able to operate without interruption once the CFAR window has been filled and as long as a continuous data stream can be supplied to the input. In this case, the only parameter affecting the performance is the operating clock frequency of the module. The maximum possible frequency f_{max} was found for each point in the design space according to the process described in section 4.1.2 so that the exact data throughput of each implementation can be stated. Amongst all investigated parameter sets, a maximum throughput rate of around 340 million cells per second was reached when using a CFAR window with less than 64 window cells and a word length below or equal to 18 bit.

Interestingly, a dependency between the area consumption and the performance is visible, which seems to be linked to the size of the CFAR window. For growing window sizes N, the maximum clock frequency and thus the data throughput decreases which can be observed in Fig. 4.3. In addition to this rule, a general upper limit of about 245 MHz exists for word sizes equal to or greater than 20 bit. Apart from that, the word size b has little or no effect on the clock frequency and for most implementations only the area consumption differs. As a result, a mainly horizontal expansion of the different clusters can be noticed. Moreover, the width of the clusters increases for larger window sizes N, which can be seen in Fig. 4.3 using the two labeled clusters with $N = 128$ and $N = 1024$ as an example.

The inverse relationship between window size and maximum operating frequency is suspected in the realization of the target detection module. The integration step of the binary comparison results, i.e. the addition of N values, is only split into two sub-steps, regardless of the window size N. As a conse-

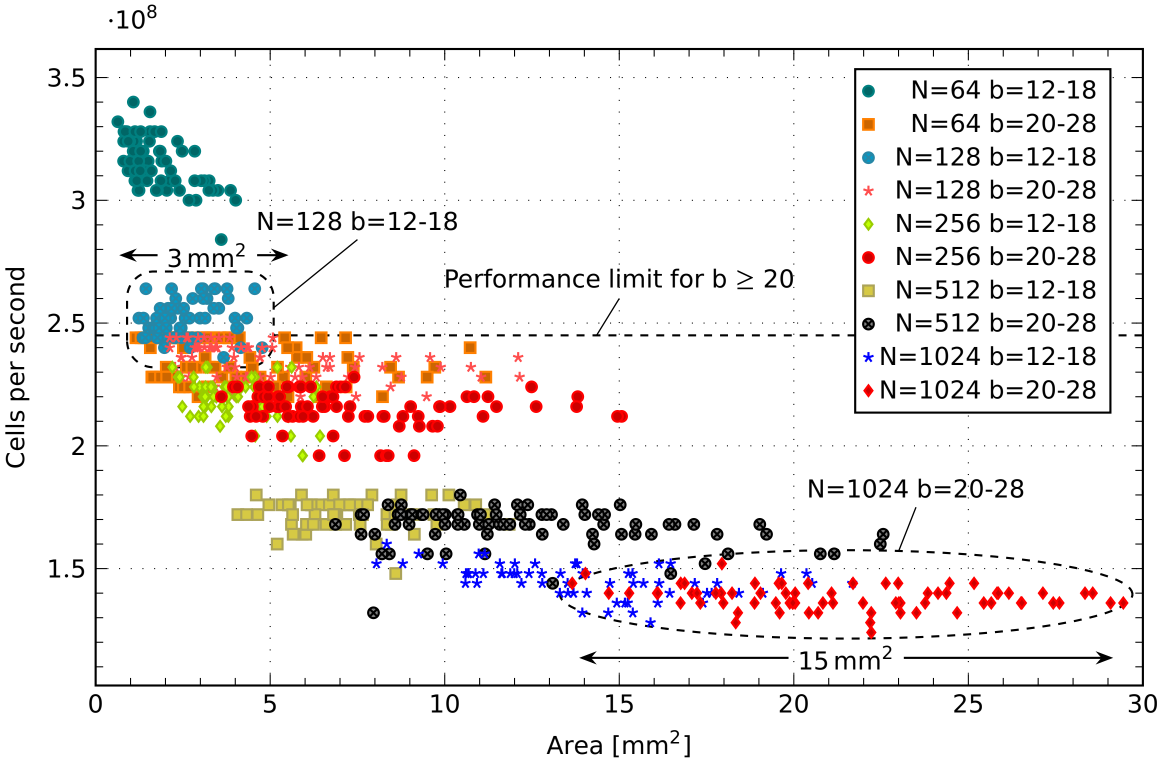


Figure 4.3: Performance and area consumption of various configurations of the target detection accelerator. In each separate data class, the channel number M is varied between 1 and 32 which has nearly no impact on the performance.

quence, more complex logic operations have to be performed within a single clock cycle if the window size keeps growing which finally results in a worse timing behavior.

4.3 Single target ML estimator

The single target ML estimator module performs an angle estimation on the basis of complex valued input data. Beside this input data, it relies on sensor specific constants which are known as steering vectors and which have to be stored in a memory inside the module. The estimation process is usually invoked after the spectrum evaluation and after the target detection process, so that a lower requirement on the data throughput exists. The angular information is provided in parallel from all receiving and transmitting channels. For a more detailed description of the module's composition refer to section 3.3.1

and for the algorithmic background to section 2.3.

The design space parameters for this module comprise mainly three variables which are briefly introduced below. Furthermore, the module can be realized in several different architectural variants. Each of these options provides the same functionality, however the performance, numerical accuracy and resource usage differ. The choice of the variant is thus marked by a fourth parameter so that the specific advantages of each implementation form can be identified in the results.

Word length (b) Defines the word length of the frequency cells at the input. One complex input value is comprised of two actual data words, the real and the imaginary part with a word size of b bits each.

Number of channels (M) This is the number of (virtual) receiving channels which the accelerator can handle in parallel. The length of each steering vector must match this value.

Number of steering vectors (N) This parameter defines the size of the angular search space and has an influence on the processing time as well as on the size of the read-only memory for the steering vectors.

Scalar product realization variant Four different variants were realized, each able to compute the scalar product. They are summarized in the following list while their functional and algorithmic background is described in-depth in chapter 3.3.1.

- *Cordic* – Complex rotation realized with pipelined Cordic processor
- *LUT-4M* – Complex multiplication using four multiply operations; realized solely with LUTs
- *DSP-4M* – Like LUT-4M, but using DSP slices instead
- *DSP-3M* – Like DSP-4M, but requires only three multiply operations

4.3.1 Area

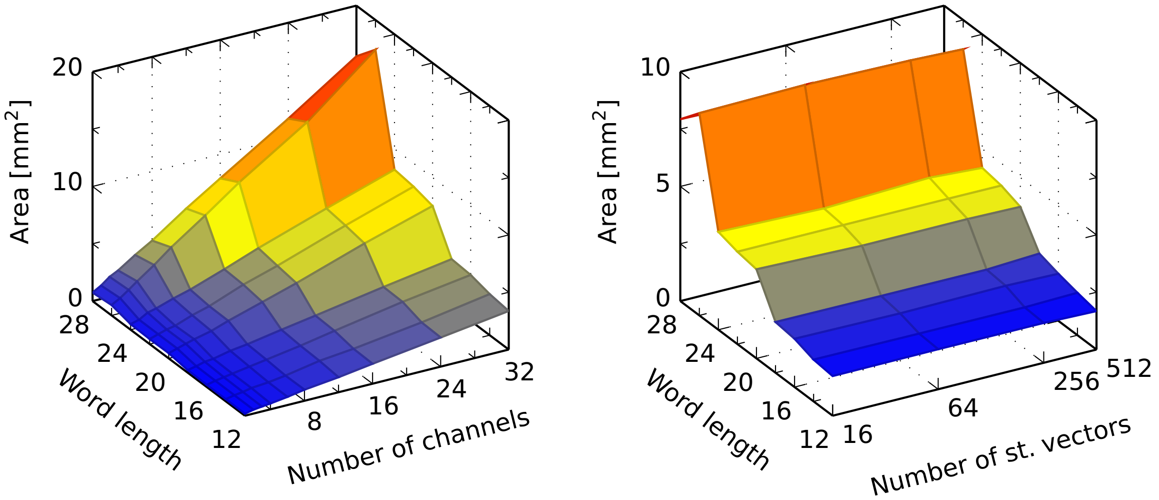
It turns out, that the DSP-based variant with three multipliers for a complex multiplication (DSP-3M) is the most efficient in terms of resource usage. Therefore, only the area consumption of this realization form is shown in the following figures. The influence of the other three design parameters are presented with the help of two separate surface plots, exactly as it was done before in section 4.2.1. Please refer there for more details about the graphical representation. Again, the two upper plots show the observed measurement data, while the two lower plots are based on the model function (4.6) which will be introduced later in this section.

In Fig. 4.4a, the figure at the left side, the number of steering vectors N is fixed to 256. A linear increase, with almost no outliers, can be observed as a function of the number of channels M. The slope of this increase is however different and changes with the word length. Concerning the second dimension, where the number of channels is fixed and the word length is varied, a more or less stepwise growth of the area can be seen, most noticeably between 18 and 20 bit, as well as between 24 and 26 bit. The root cause for the stepped characteristic is the fixed word length of the embedded multipliers (DSP slices), which has already been explained in detail on page 115.

Regarding the dependency on the number of steering vectors (cf. Fig. 4.4b), a constant scaling behavior is visible throughout the entire design space under investigation. In fact, the additional steering vectors do not provoke any additional area consumption which is not explainable without deeper analysis of the used resource blocks.

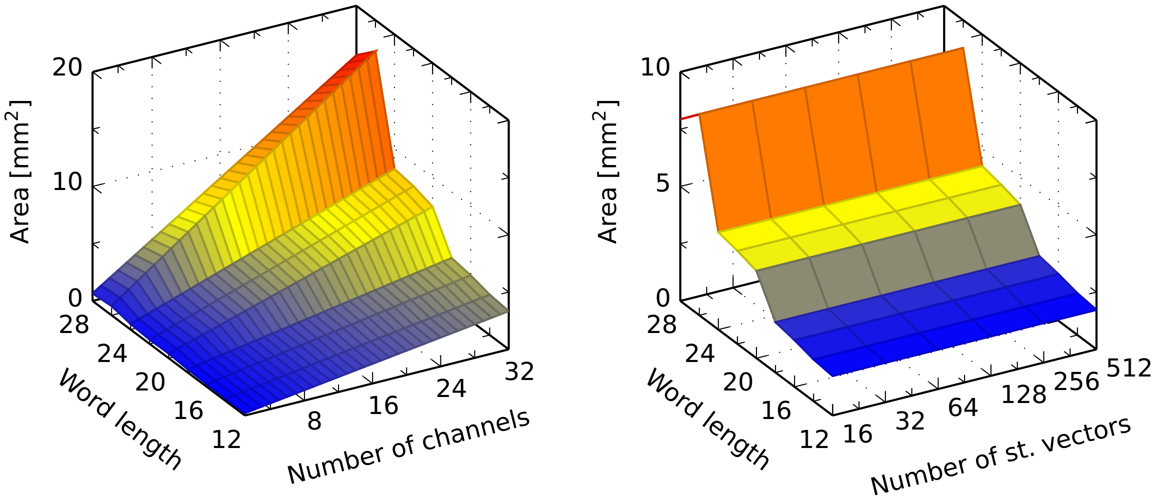
First of all, it is essential to realize that the read-only memory which stores the steering vectors is implemented by making use of the Block RAM (BRAM) resources. The minimum size of such a memory block is fixed to 18 kbit for the used FPGA device, while the width to depth ratio can be adapted to the application's needs, at least to a certain extent. When using the configuration with the widest interface, the depth of the memory is accordingly minimal, but it still exhibits 512 entries. This constraint becomes important if a massive parallel data access is required and, at the same time, the number of entries is below 512. In such a case, the remaining portion of the RAM's capacity cannot be exploited and will be wasted.

Now, in the specific case of the single target ML estimator module, a fully parallel access to one steering vector per clock cycle is required. Consequently, as long as the total number of steering vectors N is equal to or smaller than 512, the required quantity of BRAM elements is solely determined by the data width. It explains the primarily strange effect of a constant area consumption for $N \leq 512$, which is observable in Fig. 4.4b. For values of $N > 512$, additional BRAMs would become necessary and it is expected that the resource usage



(a) Observation
Number of st. vectors (N) = 256

(b) Observation
Number of channels (M) = 16



(c) Model function, refer to (4.6)
Number of st. vectors (N) = 256

(d) Model function, refer to (4.6)
Number of channels (M) = 16

Figure 4.4: Silicon area consumption of the single target ML angle estimation module on a Virtex 7 FPGA. All values are based on the DSP-3M implementation variant, which is the most efficient in terms of area consumption.

would increase with N. However, this portion of the design space was not investigated due to the lack of practical relevance.[2]

---

[2] Choosing $N = 512$ and assuming a relatively large field of view (FoV) of ±50 degrees, the resulting angular cell size would be $\Delta\theta = FoV/N = 100°/512 < 0.2°$, which is far below the cell size of current automotive applications (1° or more, cf. [3, pp. 382−401]).

A comparison to other digital beamforming accelerators from the literature, which have likewise been realized on an FPGA, is not easy to achieve. Most of the time, the modules are designed for different applications or they are integrated into a bigger framework consisting of diverse processing steps. A benchmark with two references was made and is summarized in Tab. 4.3. Further implementations of digital beamforming accelerators have been reported [134, 135], however they are not included in the table because only partial information on the design parameters were available.

Publication	Resource usage	Data throughput [Dets. / s]	Number of channels (M)	Number of st. vectors (N)	Word size (b)
This work (I)	2803 LUTs (6-input) 104 DSPs	953k	16	256	24 bit
This work (II)	12177 LUTs (6-input) 392 DSPs	953k	64	256	24 bit
Winterstein et. al. [88]	<7639 LUTs (4-input) <134 DSPs[a]	172k	16	256	>12 bit[b]
Seguin et. al. [87]	24048 LUTs (4-input) 576 DSPs	6M	64	24	10 bit

[a] The values contain a significant portion of other modules, e.g. an FIR filter. It is expected that the actual beamforming part consumes considerably less resources.

[b] The word length of the beamforming implementation is not mentioned, however the input samples from the ADC comprise 12 bit.

Table 4.3: Comparison of the resource usage of different digital beamforming implementations on an FPGA. Values from this work are based on the DSP-3M variant.

Model function

The following function is used to approximate the consumed silicon area of the single target ML angle estimation module. The influence of the word length b is modeled by a family of curves (4.6), where the parameters γ differ, depending on b (refer to Tab. 4.4). Thus, a separate model function exists for every word length b.

$$y_{Area}(M, N) = \gamma_M \, M + \gamma_0 \tag{4.6}$$

Only two terms are enough to describe the area consumption with a sufficient precision. A linear dependency on the number of channels M exists which is taken into account by the coefficient γ_M and furthermore a constant offset γ_0 is added to the equation. The coefficients were found by a least squares interpolation and are presented in Tab. 4.4. As mentioned above, the model function is fitted individually for every different word length b, because a reasonable analytical expression for the stepped and irregular behavior does not exist. It is mainly provoked by the fixed word size of the DSP slices, which has already been explained in-depth in section 4.2.1 on page 115.

A dependency on the parameter N is not included in the sum, but nevertheless the model is valid for the investigated range of steering vectors from 16 up to 512, which comprises the entire design space for practical applications at this time. The function should not be evaluated beyond $N = 512$, because a deviation from reality may happen from there on. The reason is that the minimal depth of 512 of the embedded Block RAMs will be fully utilized at this point, which has already been explained in detail in the previous subsection.

The linear dependency on the number of channels is very obvious when the graphical representation in Fig. 4.4 is studied. It originates from the parallel evaluation of the scalar product and the downstream adder tree. Due to the absence of any outliers, the quality of the fitted function is also very high which manifests itself in a low regression standard error s below 0.2 mm^2 and a R^2 value pretty close to 1.

Word size	Coefficients [mm^2]		GOF	
b	γ_M	γ_0	s [mm^2]	R^2
12	0.1009	0.1085	0.030	0.9991
14	0.1065	0.1182	0.042	0.9985
16	0.1204	0.1113	0.041	0.9988
18	0.1271	0.1640	0.049	0.9985
20	0.2274	0.1753	0.067	0.9991
22	0.2457	0.1304	0.054	0.9995
24	0.2506	0.2413	0.071	0.9992
26	0.5316	0.2756	0.105	0.9996
28	0.4847	0.1774	0.135	0.9992

Table 4.4: Coefficients of the model function (4.6) describing the silicon area consumption of the single target ML angle estimation module

4.3.2 Power

The module's power dissipation is strongly correlated with the used silicon area so that the plots in Fig. A.2 in appendix A.3.2 resemble those of the previous section. The model function for the power consumption (A.16) is also included in the appendix A.3.2 along with its coefficients which are shown in Tab. A.5. The power values given in this work comprise both, the static and the dynamic portion of the power consumption (cf. section 4.1.1, page 105 for a further explanation).

Similar to the silicon area usage, nearly no variation in the dissipated power can be observed when increasing the number of steering vectors up to 512. The maximum power dissipation does not cross the mark of 2 W even for large word lengths of 28 bits. This is remarkable, because the module has proven its effectiveness in a benchmark with a CPU implementation which will be presented in the following section 4.3.3. The CPU is specified with a maximum power consumption of 35 W even though the data throughput drops back considerably in relation to the FPGA for larger channel numbers.

Related work compared the effectiveness of several processing architectures (CPU, GPU and FPGA) for digital beamforming applications and concludes with the FPGA being the most power efficient, especially for fixed-point realizations [136].

4.3.3 Performance (Data throughput)

In this work, another variant of the single target estimator was implemented on a general purpose processor, in order to provide a basis for a performance benchmark. A state-of-the-art CPU from Intel (i5-3320M Ivy-Bridge) which runs at 2.6 GHz was used and the implementation was highly optimized in order to make use of the available vector instructions. The 256 bit AVX instruction set offers the ability to perform up to eight multiplications in parallel on a single core. This kind of SIMD[3] command can only process floating-point values which means that an exact comparison to the fixed-point implementation on the FPGA is not possible. Two different word sizes are supported by the processor, namely single-precision values (SP) and double-precision values (DP) with a word length of 32 and 64 bit, respectively. The single target estimation

---

[3] SIMD – Single instruction, Multiple data

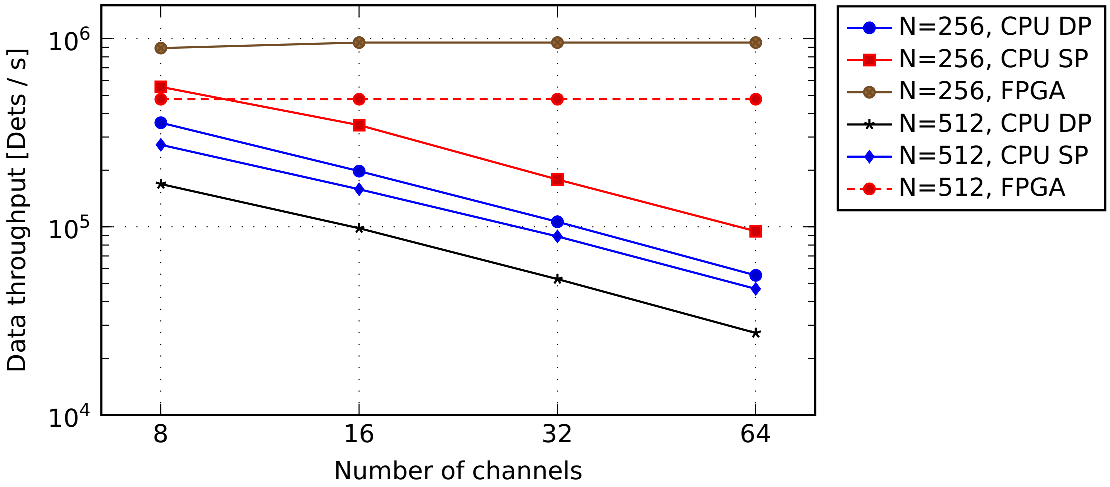


Figure 4.5: Performance in terms of data throughput for various implementations of the single target ML estimator.

routine was executed on a single core and no multi-threading was used. A further possibility of acceleration would be to exploit multiple cores, though a performance improvement might be difficult due to the already high memory bandwidth utilization.

The performance results are shown in Fig. 4.5. The FPGA implementation used for this benchmark is based on the 3-multiplier variant (DSP-3M). All complex multipliers are realized with DSP blocks and a 24 bit word length is used for the fixed-point datapath.

It should be kept in mind that the processor has to iterate over each receiving channel while the scalar product is evaluated. Hence the required processing time per detection is heavily dependent on the number of receiving channels. In contrast, the data throughput on the FPGA remains more or less constant because the scalar product is computed in parallel. Only a small deviation can be observed which originates from slightly different maximum clock frequencies. In terms of the number of steering vectors, both the CPU and the FPGA implementation achieve a lower throughput. This dependency is almost linear due to the consecutive evaluation of all steering vectors.

The speedup of the FPGA variants compared to the processor-based variants lies approximately between a factor 1.6 for 8 receiving channels and a factor 10 for 64 channels. Hence, the FPGA implementation can be considered as a powerful hardware accelerator with a significant higher power efficiency. The estimated power consumption by the used FPGA tools ranges at approximately

2.1 W for the 64 channel version. In contrast, the thermal design power (TDP) of the used CPU is specified as 35 W, which is an order of magnitude higher.

4.3.4 Numerical accuracy

The PSNR was evaluated with simulated input data for all four implementation variants and for a diverse number of channels and word lengths. Naturally, a trade-off between the required silicon area and the numerical accuracy can be made. Though, certain points in the design space are more favorable for a realization than others, because they deliver a better resource usage to PSNR ratio. Especially the four architectural options differ clearly and often only the DSP-3M can be considered as Pareto optimal.

This characteristic is demonstrated in Fig. 4.6 where the PSNR value is shown as a function of the area consumption per channel. Due to the normalization of the area usage, implementations with different channel numbers manifest themselves in closely spaced clusters. Interestingly, for some specific PSNR levels, the four multiplier variant (DSP-4M) does not consume more silicon

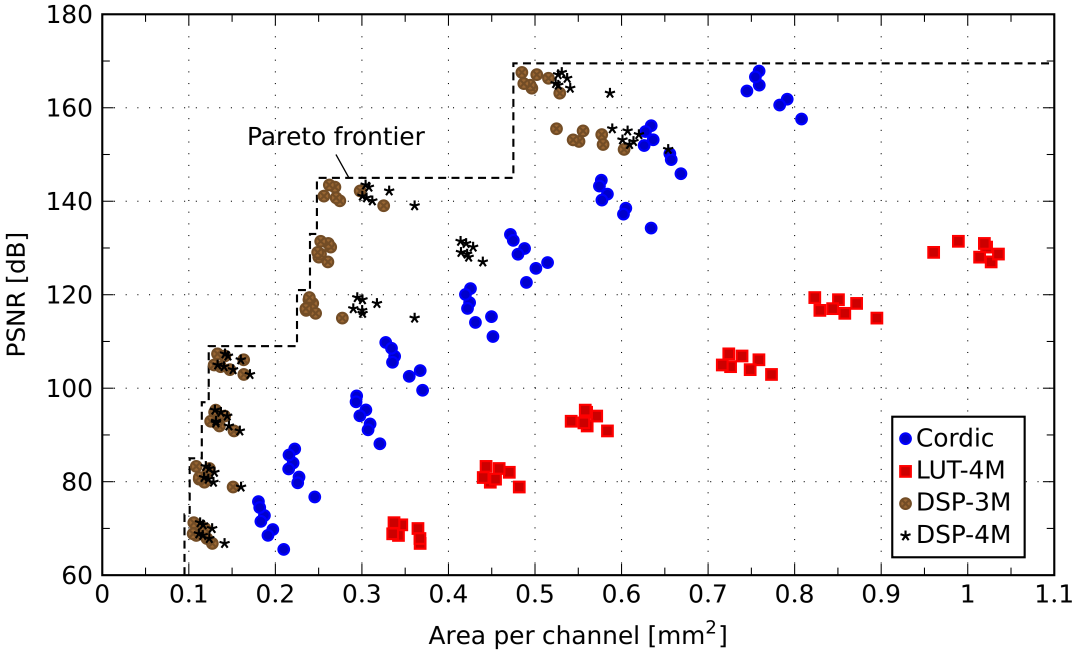


Figure 4.6: Fixed-point accuracy (PSNR) and area consumption of various configurations of the single target ML angle estimation accelerator. All values with 256 steering vectors and between 4 and 32 channels.

area than the version with three multipliers. It seems that in some favorable configurations the additional multiplier can be compensated by the lower number of adders which are required for the DSP-4M implementation form. The Cordic variant does not reach the Pareto frontier even though it approaches the DSP-based versions for some specific word lengths. The LUT-based multiplier variant (LUT-4M) is by far the worst option throughout the investigated design space, due to its very low area efficiency.

A direct comparison between the Cordic variant and the DSP-based multiplier variants can be misleading, because the principal logic operation of the Cordic implementation are additions, which are consequently realized by general purpose LUTs. The corresponding adders, which are required in a large quantity, will exhibit a lower area efficiency than the hardwired multipliers inside the DSP slices. However, this relationship holds only for the specific FPGA architecture, because a dedicated and hardwired adder block is not existent. A benchmark of the Cordic variant on another target device, like for instance a custom ASIC, would be interesting and may potentially lead to different results.

4.4 Two target ML estimator

A prerequisite for the two target ML estimation process is the computation of the scalar products r_θ between the input data and the steering vectors. Nevertheless, a full evaluation of these scalar products can be avoided, because they have already been calculated during the single target estimation process. The most straightforward approach is certainly to cache and reuse the previous results. Consequently, the figures presented below do not include computational logic to evaluate the scalar product and the module relies on the availability of the r_θ values at its input. For more information about the architecture of the module refer to section 3.3.2 and for the theoretical background of the ML estimation in the two target case to section 2.3.3.

The design space for this module comprises mainly three parameters which are shortly described in the following list.

Word length (b) Defines the word length of the scalar products at the input. One complex input value is comprised of two actual data words, the real and the imaginary part with a word size of b bits each.

Number of PEs (K) This is the number of processing elements (PEs) which evaluate the ML function in parallel. This number can be adapted to the application's needs and determines the trade-off between silicon area and data throughput.

Number of steering vectors (N) This parameter defines the size of the angular search space and has an influence on the processing time as well as on the size of the read-only memory for the constant, sensor-specific β_{ij} values.

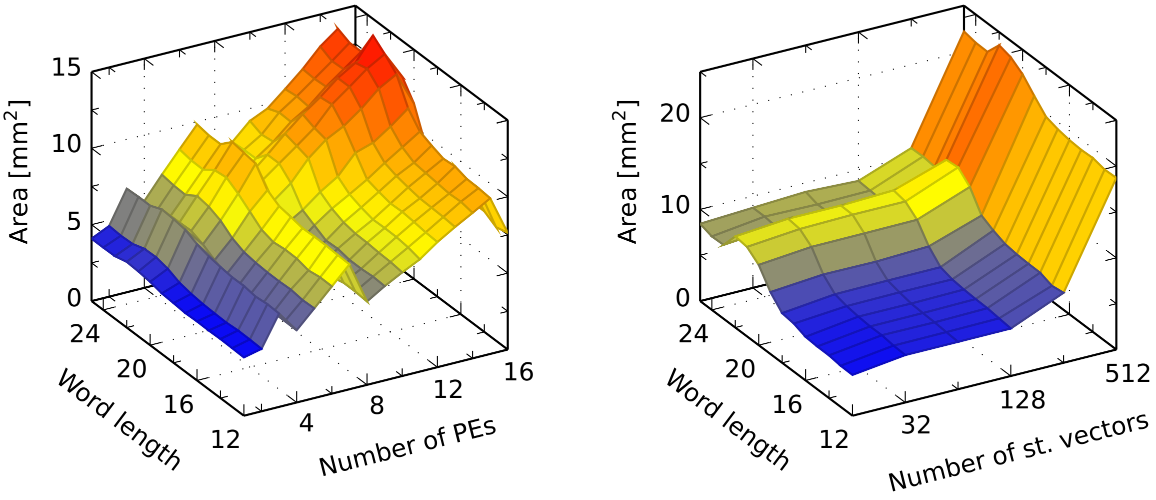
4.4.1 Area

Fig. 4.7 shows the equivalent area of the two-target DML module implemented on a Virtex 7 FPGA. Several observable characteristics of the area consumption can be related to a respective parameter which will be further explained in the following paragraphs.

The dependency on the used word size can be observed from both plots. Two almost constant regions which are connected by a transition zone in-between predominate the shape of both plots. The area consumption remains approximately the same for word lengths below 18 bit, due to the used DSP slices. As long as the input operands are small enough so that they fit into a single DSP block, the resource usage increases only marginally. The same effect is reached above 24 bit for the input words because the operations are then performed by four DSP slices which offer enough capacity for even larger word lengths. The slope of the transition zone is variable because the intermediate word lengths are mapped to a hybrid implementation using both, DSP slices and LUTs for its realization. Please refer also to page 115 where this property has already been explained in-depth.

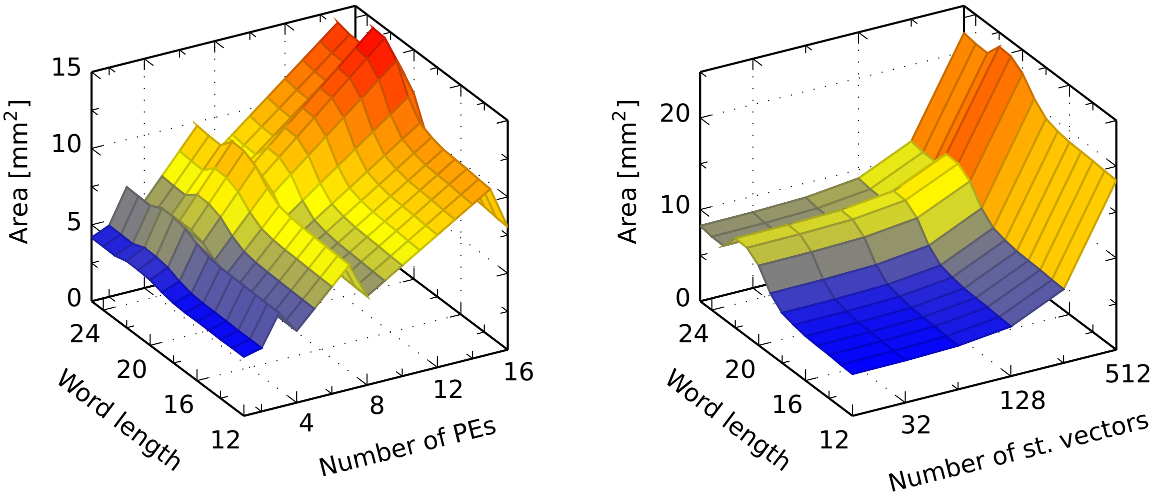
It is known from the module's structure, that the arithmetic unit of the hardware module is independent of the number of steering vectors. Nevertheless, a dependency of the resource usage can be observed which is provoked by the memory storing the β_{ij} values. The more steering vectors have to be processed, the more constant values have to be stored, which is why an increasing resource usage can be observed. The area consumption exhibits a quadratic dependence on the number of steering vectors due to the two-dimensional search space.

Regarding the number of PEs, a quite remarkable, stepped behavior of the



(a) Observation
Number of st. vectors (N) = 256

(b) Observation
Number of PEs (K) = 16



(c) Model function, refer to (4.7)
Number of st. vectors (N) = 256

(d) Model function, refer to (4.7)
Number of PEs (K) = 16

Figure 4.7: Silicon area consumption of the two target ML angle estimation module on a Virtex 7 FPGA.

area consumption can be seen in Fig. 4.7a. With each PE, the full arithmetic unit performing all addition and multiplication operations has to be replicated, so that a linear increase of the resource usage would be obvious. However, a periodic behavior, where at certain points the used resources even decrease, can also be seen in Fig. 4.7a. The reason for this primarily strange characteristic lies in the nature of the memory realization which only supports power of two sizes. Hence, the most efficient implementations are those without unused memory due to the restricted steps of memory sizes.

Model function

The following function is used to approximate the consumed silicon area of the two target ML angle estimation module. Once again, the dependency on the word length b is modeled by a family of curves (4.7), where the coefficients δ differ for every word length b (cf. Tab. 4.5).

$$y_{Area}(K, N) = \delta_K \, K + \delta_{KN} \, N^2 \, K \, 2^{-\lfloor \log_2 K \rfloor} + \delta_0 \tag{4.7}$$

The first summand in (4.7) stands for a linear increase of the silicon area as a function of the number of lanes K. For each additional lane, the entire arithmetic unit has to be replicated which is taken into account by the coefficient δ_K.

The second summand depends primarily on N^2 due to the resource consumption of the two-dimensional coefficient memory. The additional term $K \, 2^{-\lfloor \log_2 K \rfloor}$ models the possible memory overhead for unfavorable values of K, where the operator $\lfloor \; \rfloor$ denotes the *floor* function, i.e. rounds down. Only if K is a power of two, that is for $K \in \{1, 2, 4, 8, \ldots\}$, the memory can be fully utilized and no overhead exists. In this case, the term $K \, 2^{-\lfloor \log_2 K \rfloor}$ becomes one and cancels itself out.

The fitted result has a very high accuracy which can be seen in Tab. 4.5. The regression standard error averages out at $0.3\,\mathrm{mm}^2$ and the coefficient of determination R^2 is consistently above 0.99. Moreover, the visible results in Fig. 4.7 are virtually identical to the observations.

4.4.2 Power

The estimated power dissipation of the two target estimation module is shown in Fig. A.3 in appendix A.3.3 in the same manner as the area consumption. The corresponding model function (A.17) and its coefficients in Tab. A.6 are likewise included in appendix A.3.3.

In general, the figures look very similar to the silicon area usage from the previous section. Each additional logic resource has to be powered so that a strong correlation between area and power consumption is observable. Though, the stepped increase as a function of the number of PEs is by far less pronounced.

Word size	Coefficients [mm^2]			GOF	
b	δ_K	δ_{KN}	δ_0	s [mm^2]	R^2
12	0.2774	$0.0531 \cdot 10^{-3}$	0.1147	0.264	0.999
13	0.2816	$0.0531 \cdot 10^{-3}$	0.1074	0.252	0.999
14	0.2905	$0.0531 \cdot 10^{-3}$	0.0778	0.248	0.999
15	0.2921	$0.0532 \cdot 10^{-3}$	0.0969	0.251	0.999
16	0.2964	$0.0532 \cdot 10^{-3}$	0.1026	0.255	0.999
17	0.3135	$0.0530 \cdot 10^{-3}$	0.0772	0.252	0.999
18	0.3349	$0.0533 \cdot 10^{-3}$	0.0861	0.253	0.999
19	0.4044	$0.0535 \cdot 10^{-3}$	0.1490	0.300	0.998
20	0.5223	$0.0524 \cdot 10^{-3}$	0.3186	0.312	0.998
21	0.5826	$0.0521 \cdot 10^{-3}$	0.3426	0.303	0.998
22	0.5867	$0.0522 \cdot 10^{-3}$	0.3472	0.305	0.998
23	0.4851	$0.0527 \cdot 10^{-3}$	0.2006	0.257	0.999
24	0.4860	$0.0527 \cdot 10^{-3}$	0.2208	0.259	0.999
25	0.5004	$0.0527 \cdot 10^{-3}$	0.2082	0.288	0.999

Table 4.5: Coefficients of the model function (4.7) describing the silicon area consumption of the two target ML angle estimation module

In contrast, the local maximum in the region around 20 bit word length is intensified and it seems that the hybrid LUT/DSP realizations consume a great amount of current.

The overall power consumption is astonishingly small with values below 2 W even for 16 PEs in parallel. A CPU-based variant of the two target estimator, which will be presented in detail in the next section, falls behind clearly in terms of data throughput even though the processor is specified with a maximum power dissipation of 35 W.

4.4.3 Performance (Data throughput)

The performance of the two target DML estimation hardware module was evaluated in the same manner as the single target module. A processor based-implementation was used again, with the same framework and parameters as in section 4.3.3. Again, a 24 bit word length for the fixed-point FPGA module was chosen as reference for this benchmark. This time, the processing time per detection is independent from the number of receiving channels because the previously computed scalar products can be reused. Thus, only the number of steering vectors determines the total runtime and a quadratic dependence on

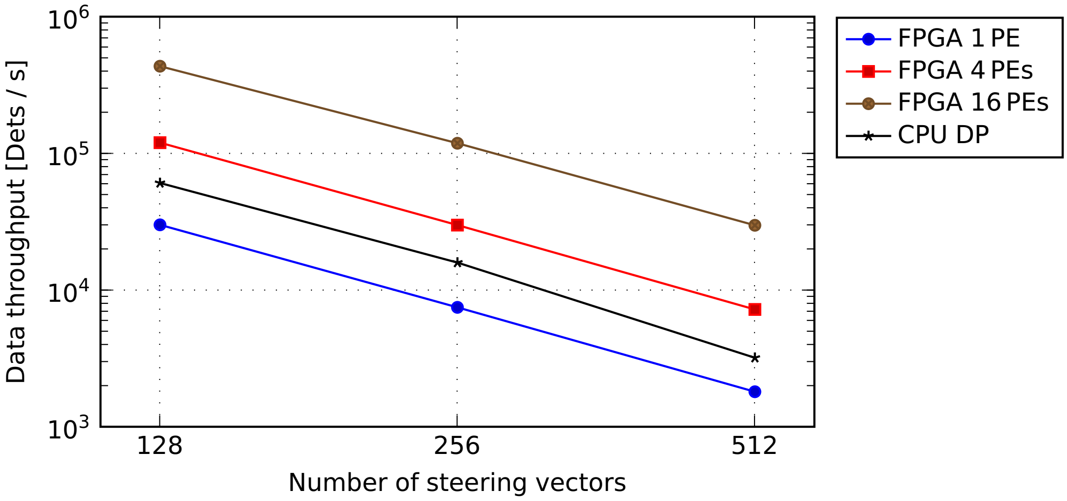


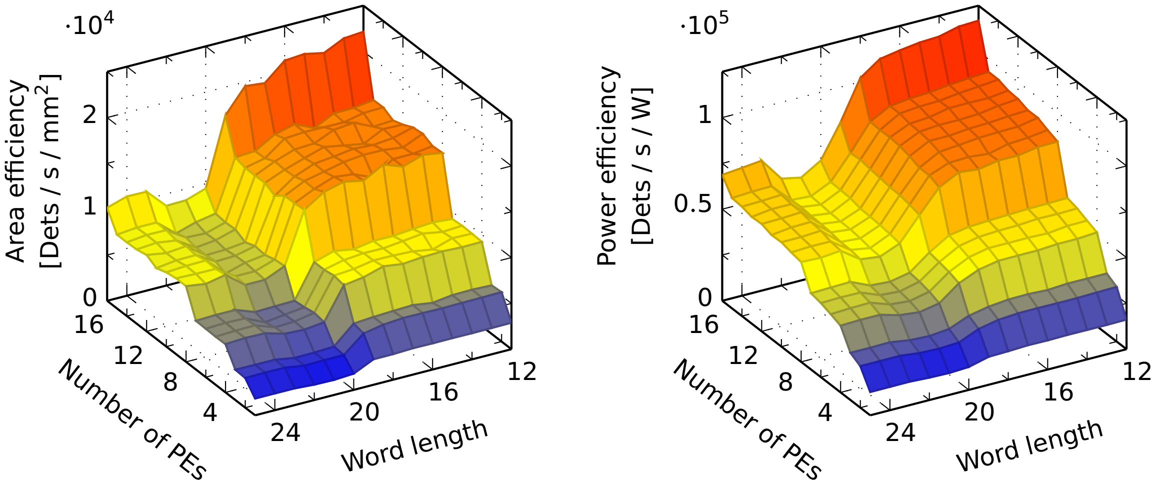
Figure 4.8: Performance in terms of data throughput for various implementations of the two target ML estimator.

this parameter can be noticed. The hardware implementation offers further the possibility to employ multiple PEs in parallel at the cost of a higher resource usage.

The results are shown in Fig. 4.8, where it can be observed that the processor based version is faster than the dedicated hardware implementation with a single PE, but slower than the version with four PEs in parallel. With an even higher number of PEs, the performance of the FPGA module can be further increased if required by the application. For instance with 16 PEs in parallel a speedup of factor 7 compared to the CPU can be achieved, while less than 5 % of the FPGA's resources in terms of silicon area are occupied. In contrast to the CPU, the power consumption is considerably lower and lies at approximately 1.45 W for the parallelization with $K = 16$.

4.4.4 Area and power efficiency

The module offers the possibility to adjust the data throughput by choosing an arbitrary parallelization factor K, which determines the number of processing elements (PEs). Though, this is only a trade-off because the area consumption will also increase with a higher parallelization. For the designer it might be interesting to observe the area efficiency in dependency of K so that an eventual loss in efficiency can be identified for certain configurations.



(a) Area efficiency **(b)** Power efficiency

Figure 4.9: Area and power efficiency of the two target ML angle estimation module on a Virtex 7 FPGA. The number of steering vectors is fixed at 256 for both figures.

For the computation of the area efficiency, the maximum achievable data throughput of the module was used, i.e. the accelerator was operated at $f_{\max}$ and the input values were supposed to be supplied without any interruption. The result is shown in Fig. 4.9a as a function of K and the word length. The reader should take notice of the interchanged axes of abscissas which differ from the previous surface plots. Beside the area efficiency, the power efficiency is also shown even though it originates always from a discrete clock frequency and not from an ideal operating point.

A stepped and terraced characteristic can be observed in Fig. 4.9, i.e. there are certain areas where the efficiency remains virtually constant and there are also some regions with a distinct step of the efficiency. These steps occur in both dimensions and manifest themselves as straight edges across the whole figure. For instance if the word length traverses 18 bit, the efficiency clearly drops due to an increased area consumption but also due to a lower $f_{\max}$ which was to be expected. The steps in dependency of K occur always if a power of two is reached, more precisely at 2, 4, 8 and 16 PEs in this figure. The location of the steps originate from the likewise stepped area consumption which has already been explained in section 4.4.1. More interestingly, the efficiency increases with a higher parallelization which is an exceptional behavior and needs to be further explained.

It was mentioned that the full computational logic has to be replicated for each additional processing element. However, the memory storing the β_{ij} values does not need to be cloned one-to-one because every PE will only operate on a subset of the i, j indices. As a result, the individual memory size of each PE decreases for a larger number of K and hence the area per PE also decreases. This is the reason for a better efficiency at a higher parallelization degree which is quite remarkable.

The observable deviations in the primarily constant and flat regions of the area efficiency are caused by small fluctuations of the maximum operating frequency. When looking at the power efficiency in Fig. 4.9b, it is conspicuous that the surfaces are much smoother than in Fig. 4.9a. The reason is that a higher operating frequency always comes along with an increased power consumption. In terms of efficiency, the local peaks of $f_{\max}$ are canceled out by a higher power consumption and in the end, the power efficiency remains constant.

It remains to underline that only the implementations for which K is a power of two can be considered as Pareto optimal. This is demonstrated in a slightly different data representation in Fig. 4.10 where the data throughput is shown as a function of the area consumption. The different word lengths are connected in increasing order and different parallelization degrees can be identified through different colors and markers. Two reference lines were added which mark the Pareto frontier for 16 and 24 bit word length, respectively. Only the power of two realizations are located on these lines in contrast to the two examples with $K = 6$ and $K = 12$ which drop behind clearly.

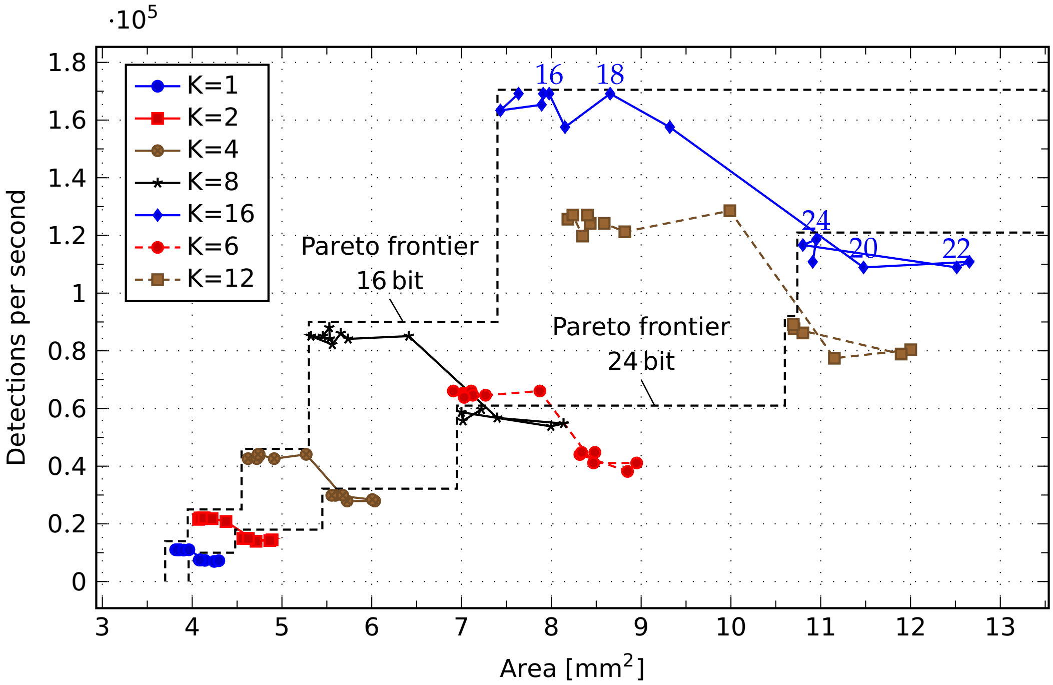


Figure 4.10: Data throughput as a function of silicon area consumption for various parallelization degrees (K). For every different K, several word lengths are shown; they are connected by lines. All values are based on 256 steering vectors.

5

Verification with an Experimental High-Resolution MIMO Radar

During the course of this thesis, a high-resolution prototype radar sensor was designed and built up in my working group at Bosch[1]. One of its purposes is the ability to verify the implemented signal processing architectures under real world conditions. Besides, the operational radar prototype was used successfully for further research and development of automotive radar processing techniques and algorithms. In combination with the presented FPGA-based hardware accelerator, the experimental setup achieves highest performance in terms of range, velocity and angular resolution. It enabled the acquirement of high-resolution measurement data, which was in turn used for further studies and lead to numerous publications [6–23].

The whole system was designed from scratch with the objective to provide a modular radar framework with the ability to exchange certain components easily. Parts of the radio frequency (RF) front end were developed in collaboration with the Institute for Communications Engineering and RF-Systems of the Johannes Kepler University (JKU) in Linz. The remaining baseband and power supply circuitry was developed within Bosch during this thesis' time span. For the FPGA-based processing platform, mainly off-the-shelf components were used which kept the hardware development effort minimal.

The first complete radar setup was ready for operation at the end of 2014, in the beginning solely as a non-moving and stationary sensor. Since mid 2016, it

---

[1] Advanced Engineering Sensor Systems, Chassis Systems Control (CC/ENA2) in Leonberg

has become possible to mount the sensor on a test vehicle, and thus dynamic traffic scenarios in combination with a moving sensor can be investigated. The system is operational in connection with the presented preprocessing architectures, so that a real-time data acquisition and reduction is possible. It is expected that further research will be based on this framework and finally new findings can be made with the help of this radar sensor prototype.

In the following sections, the hardware architecture and some operating parameters will be presented (cf. 5.1 and 5.2). Finally, the signal processing performance of the previously described architectures is evaluated and verified with the help of real world radar measurements. In order to provide data of complex traffic scenarios, the prototype sensor was mounted onto a test vehicle. The corresponding results are discussed in section 5.3. They demonstrate the enhancements in resolution, they provide new insights and finally they help to deduce requirements for future automotive radar systems.

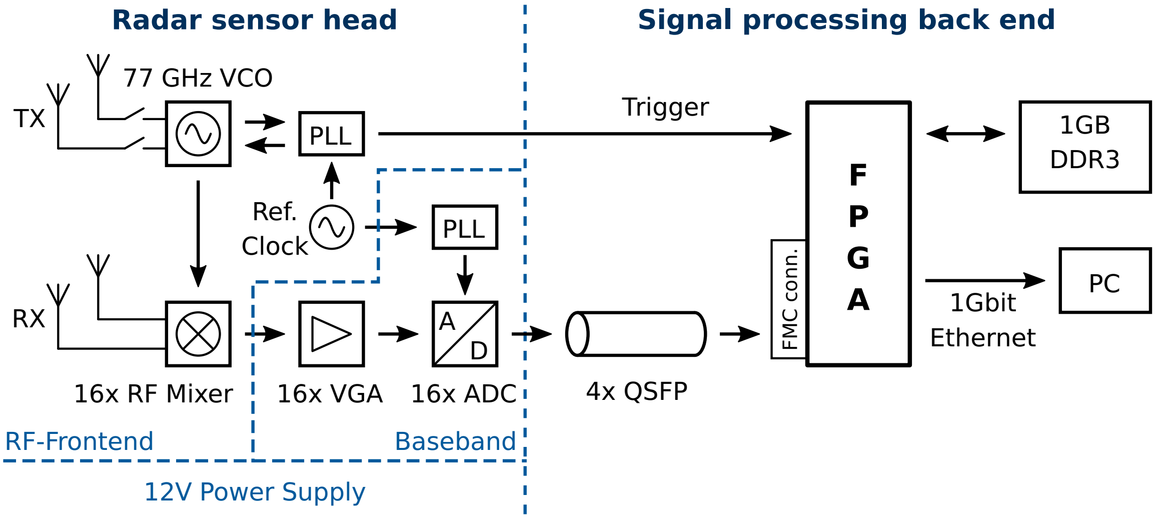
Some parts of the presented results in this chapter were also published in [11].

5.1 Overall radar system architecture

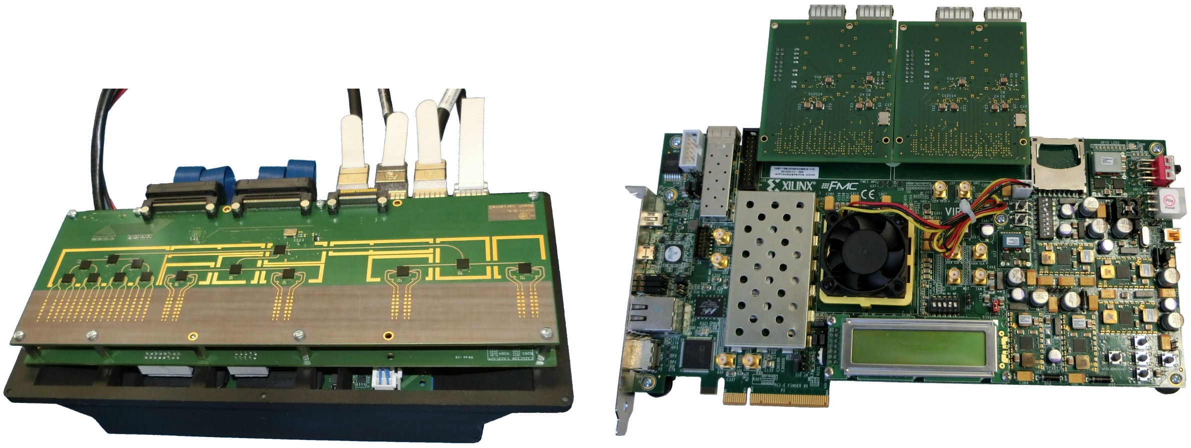
The experimental radar sensor system used for this work is split into two major parts, which can be separately placed inside a test vehicle. The first part consists of an RF front end in connection with additional signal processing and analog-to-digital (A/D) conversion circuits. The second part is completely digital and consists of an FPGA processing platform including an external DRAM memory and some extension modules for a raw data connection to the radar front end. Finally, a PC or laptop is connected via Ethernet to the FPGA to record and visualize the data.

The main advantage of this split architecture is the spatial separation of radar front end and signal processing unit. It helps to speed up the integration of prototype setups with different antenna arrays into a vehicle, because the whole processing platform can be kept without any modifications. The connection between the two parts is made with the help of a digital high-speed link which is transmitting raw data.

The architecture can be seen in the block diagram in Fig. 5.1a and the components are described briefly in the following paragraphs.



(a) Block diagram



(b) Radar sensor head

(c) Signal processing back end

Figure 5.1: Radar prototype system which was developed and used for the verification of the signal processing modules. The two parts are connected via a high speed raw data link.

5.1.1 Radar sensor head

The sensor head is composed of three distinct circuit boards which are all connected via short cables or direct board-to-board connectors. They are mechanically integrated into a plastic housing with an optional radome which can be placed in front of the antennas.

Radio frequency (RF) front end

The signal generation is based on automotive monolithic microwave integrated circuits (MMIC), which support fast chirp linear FMCW ramps at an operating frequency of 77 GHz. A phase-locked loop (PLL) is used in conjunction with a voltage-controlled oscillator (VCO) to generate the waveform.

The timing as well as the bandwidth can be user-defined at runtime, so that a wide range of different modulation settings are supported by the framework. The maximum bandwidth is limited around 3 GHz by the VCO tuning range, which results in a range resolution of about 0.05 m. Compared to state-of-the art systems, the bandwidth and hence the range resolution is approximately 5 to 10 times higher [3, pp. 382,388].

The used patch antennas offer a wide opening angle of ±50 degrees, so that a rather large field of view is covered by a single sensor. In combination with sufficient measurement time, the processing gain of the system is large enough to detect small targets in 100 m range and beyond. Therefore, far objects as well as objects to the left and right of the sensor can be recognized simultaneously.

The modular concept allows the usage of many different antenna layouts without changing the other components of the system. Two RF front ends with different antenna configurations were used primarily during the development of this thesis: 4 Tx – 8 Rx, as well as 8 Tx – 16 Rx.

The number of 16 receiving channels is the limit for the connected baseband circuitry and the subsequent FPGA-based processing system. In contrast, the number of sending antennas is not restricted to eight so that a further increase of the aperture size would be possible without changing the baseband conditioning and processing hardware.

Baseband signal conditioning and analog-to-digital converters (ADC)

The mixer outputs from the RF front end designate the interface to the following processing stage. At this point, the received radar signals have already been demodulated to a baseband signal with a much lower frequency range. The purpose of the following circuitry is to condition and prepare the signals for the subsequent A/D conversion. The required components are located on a separate circuit board which is mounted directly behind the RF front end (cf. Fig. 5.1b). Differential signal pairs are used throughout the system in order to minimize external interference, noise and crosstalk.

First, the signal level is increased by a variable-gain amplifier (VGA) to match the input range of the ADC. Their gain factor can be adjusted to maximize the dynamic range of the system without any overdrive or distortion. Additional coupling and filtering circuits limit the bandwidth below the Nyquist frequency. After the amplification, each channel is converted into the digital domain by a discrete high-speed ADC. Finally, the digitized signals are output as a serialized data stream.

The ADC sampling clock is derived from a common reference clock which is used for the RF ramp generation as well. This assures a coherent sampling which is essential for the evaluation of the employed chirp sequence modulation (cf. section 2.1.4). Besides, an additional PLL is used for jitter cleaning and synchronous clock distribution to all ADC devices on the circuit board. Phase mismatches between the individual channels were minimized as far as possible so that the beamforming process is not deteriorated.

Sensor power supply

A large number of parallel channels, sophisticated low-noise amplifiers, high-speed ADCs as well as low ripple voltage regulators provoke a rather high power consumption in the order of 50 W. This causes quite large supply currents of several amperes which have to be distributed across all ICs of the system. Furthermore, several different voltage levels are required throughout the system. In order to operate the radar sensor from a single 12 V supply, a custom step-down converter circuitry was developed and integrated into the sensor head. The voltage is first converted down by a switching regulator with a high efficiency of more than 90 %. A linear regulator with a smaller voltage drop is then used to filter the voltage ripple from the upstream switching regulator. Further filtering circuit components reduce the noise in the supply voltage to a minimum in order to increase the overall sensitivity of the radar sensor. All voltage conversion circuits are located on a third board which is also integrated into the sensor head.

5.1.2 Signal processing back end

High speed raw data link

The digital connection to the radar sensor is based on a serialized high-speed data link with signaling rates of up to 5 Gbit/s per Rx channel. The total raw bitrate of the link is hence $16 \times 5 = 80$ Gbit/s. Before transmission, the raw data words from the ADCs are first serialized, then scrambled and finally encoded with a 8b/10b code so that a 20 bit code word carries the data of one sample. The whole serialization and encoding process is a standardized procedure and happens inside the ADC circuits [137]. The line code assures enough state transitions, a mean-free signal and an error detection capability. At the FPGA side, the signal is received by integrated multi-gigabit transceiver blocks which perform in turn the deserialization and decoding of the data words. Finally, after descrambling, the raw data words are ready for processing.

The used QSFP connectors integrate four differential lanes into one cable so that the total connector and cable count is divided by four [138]. Another advantage of this off-the-shelf technology is that copper cables as well as fiber optic cables can be used. In the pictures **(b)** and **(c)** of Fig. 5.1, the QSFP connectors can be recognized easily by the four prominent metal cages which are mounted on the circuit boards on both sides of the system.

FPGA-based processing unit

In order to keep the development effort minimal, a commercial available FPGA platform was used. The VC707 board from Xilinx features a Virtex7-485T FPGA in conjunction with a 1 GB external DRAM [139]. The maximum memory interface bandwidth for this system is bounded to approximately 100 GBit/s, which gets fully utilized when running the ADCs at their maximum sample rate. A 1 Gbit/s Ethernet interface is used for the link to a PC which provides measurement and visualization tools. The VC707 FPGA-board features two

FMC connectors with a high pin count which can be used for many applications. The required QSFP interfaces are added to the FPGA by two custom FMC breakout boards, which act as a connector adapter (cf. Fig. 5.1c). In addition, some general purpose pins are used for the connection of low speed control signals, namely a Serial Peripheral Interface (SPI) and trigger lines. The clock signal of the AD-converters is recovered from the data stream and used for the signal processing logic inside the FPGA. Thus, this backend is fully synchronous to the radar front end, although it is separated spatially.

5.2 Acquisition of measurement data

For the research and development of radar signal processing algorithms, the availability of real world datasets is crucial. Certain steps like proof-of-concept, performance evaluation, benchmarking, functional tests and verification are virtually impossible to conduct without measurement data. The continuous acquisition of sensor data during the development cycle helps to find weaknesses and pitfalls early enough so that a high quality of the final product can be ensured.

Requirements on the data format

The required data format differs according to the investigated signal processing steps and according to the development task. A database containing raw sensor data offers the greatest flexibility and thus supports the largest number of possible use cases, because any other data format can be deduced from it. Furthermore, it is possible to exchange the processing algorithms and parameters later in order to reprocess the measurement data. Finally, raw data is often required since it is the only way to ensure and monitor a correct operation of all system components. For instance, a problem related to the analog parts of the circuit boards can be possibly only tracked down by analyzing the full frequency spectrum and the occurring noise components.

Unfortunately, raw data has also the largest storage requirements which can become an issue when a huge number of datasets needs to be recorded. Another considerable challenge during data acquisition is the limited write speed of current available PC systems, which is bounded by some 1000 MB/s, mostly due to the used solid-state drives. When using traditional hard disk drives based on magnetic storage, the writing speed is even lower. Furthermore, another constraint is often the used data interface to the PC, which is a 1 GBit/s Ethernet interface in this case. Other technologies like USB 3.0 or PCIe are available at higher bandwidths, however the continuous data streaming of large sequences to the hard disk drives is still not self-evident. Practical experiments with the presented radar prototype revealed that occasional data loss occurs already at datarates of 1 GBit/s, due to the lack of real-time capabilities of the operating system and the overloaded write caches of the solid-state drives.

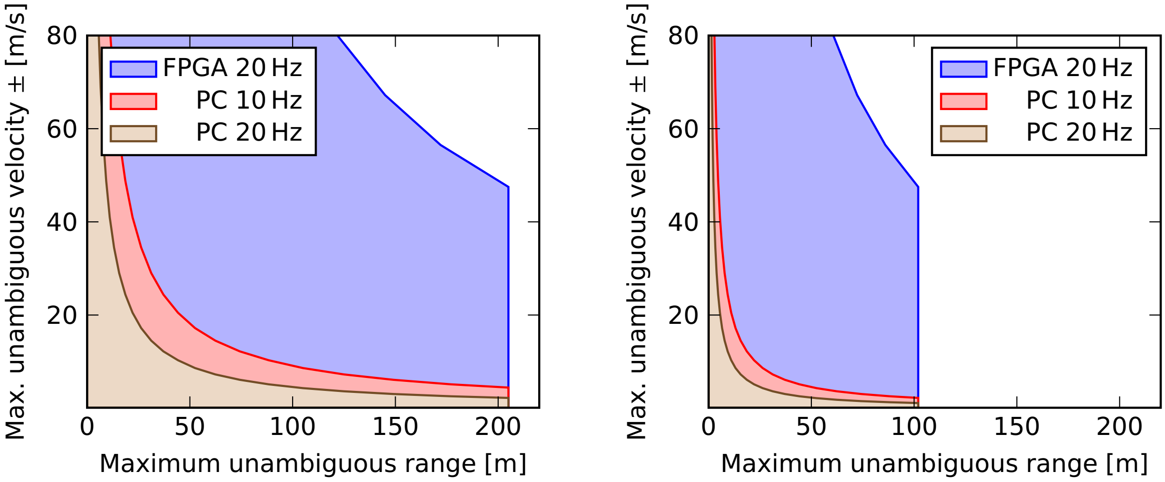
A good alternative to raw data is to store only the raw detections which are extracted by the CFAR processing step. Accordingly, most of the noise components can be discarded before the recording takes place. The space requirements can be reduced by a factor of 100 or more (cf. section 5.3.2), while the preprocessed data is still universal enough for most of the development tasks. For instance a benchmarking of different tracking algorithms would presumably be based on preprocessed radar detections.

Operational modes

According to the different requirements on the measurement data format, the presented prototype was designed to support both, the acquisition of raw data, as well as data preprocessing in real-time. In fact, the on-line data reduction is crucial for some high-resolution parametrizations, which cannot be realized in the raw data mode. Below, both modes of operation will be compared on the basis of their respective measurement cycle rate, which is an essential parameter for real-time applications. Often the terms *frame rate* or *update rate* are used synonymously. A single measurement cycle, which is also referred to as *single snapshot*, corresponds to a full chirp sequence according to the presented processing scheme in chapter 2.1.4.

The diagrams in Fig 5.2 show the operational range of the FPGA-based and the PC-based mode. If the PC-based raw data acquisition mode is used, the unambiguous range and velocity has to be decreased in order to achieve the desired measurement cycle rate of 20 Hz. Even though the unambiguous area in Fig. 5.2 can be traded against update rate, it is only reasonable to some extent. In contrast, when using the FPGA-based preprocessing mode, the unambiguous area can be maximized while the cycle rate of 20 Hz remains constant.

An upper limit for the unambiguous range can be observed in both figures. It



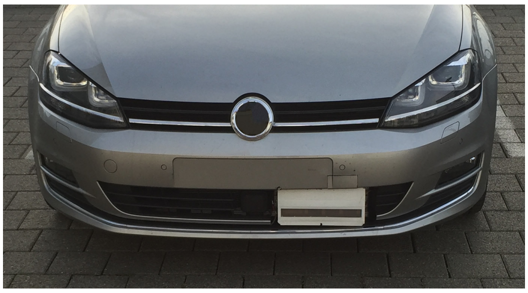
(a) Δr = 0.1 m Δv = 0.092 m/s **(b)** Δr = 0.05 m Δv = 0.046 m/s

Figure 5.2: Operational range of the FPGA-based, CFAR processing mode (blue) and the PC-based, raw data mode (red and brown). The frame rate is fixed at either 10 or 20 Hz. The resolution in range and velocity is doubled from **(a)** to **(b)**.

results from the maximum Range FFT length of 4096 samples which the FPGA implementation can handle. The half of this value (due to the real-valued transform) multiplied by the cell size limits the overall unambiguous range. Even though an increase of the maximum FFT length would be possible in raw data mode, it is not shown in the plots because this operation mode is only of theoretical nature due to the very low unambiguous velocity.

Moreover, the operational range has to be understood as hypothetical space which is supported by the processing back end. Certain configurations cannot be put into action due to physical limitations of the analog components. This could be an upper limit of the frequency ramp slope or a minimum chirp time.

The two different operational modes, namely either CFAR preprocessing or raw data recording, make the prototype system very powerful but yet flexible. Until now, the acquirement of raw data has been a convenient approach for prototyping new systems. However, this experimental radar system shows that a real-time data preprocessing will become more important, because the huge data rates can't be handled by the well-established PC-based development tools anymore. It is expected that FPGA-based processing platforms will be increasingly used in the future because they offer the required computational power as well as high bandwidth interfaces.



(a) Test vehicle equipped with the radar prototype sensor (white housing)

Bandwidth	3 GHz
Measurement time (single snapshot)	40 ms
Number of channels	16
Raw data amount (single snapshot)	128 MByte
Frame rate	20 Hz
False alarm rate	10^{-6}

(b) Modulation parameters

Figure 5.3: Prototype setup used for evaluation measurements.

5.3 Real world evaluation

A test vehicle was equipped with the sensor prototype, according to the setup presented in section 5.1. The mounting position in the vehicle bumper can be seen in Fig. 5.3a and the used modulation parameters are listed in Fig. 5.3b. The power supply and a PC were put into the back of the car, along with other measurement instrumentation. A video camera provides a reference image of the scene which is recorded synchronously to the radar data.

With this setup, real world measurement data can be recorded for research and development purposes. The seamless integration into a commercially available passenger car makes it possible to perform measurements of many different traffic scenarios on public roads. Furthermore, other traffic participants are not distracted by an odd-looking prototype vehicle so that an unbiased behavior of all road users can be supposed.

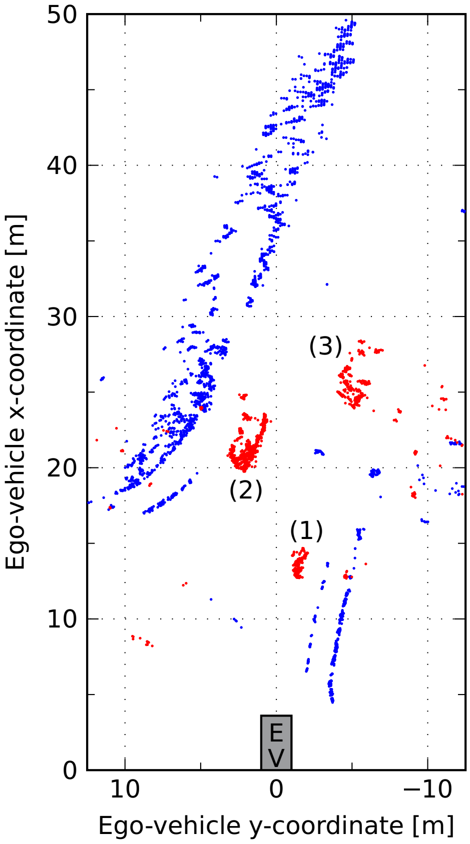
5.3.1 Single snapshot performance

During normal operation, the radar sensor is continuously scanning the environment and provides either the raw data or an already processed version according to the signal processing steps presented in chapter 3. In both operation modes, the smallest meaningful chunk of data is the result of a full chirp sequence measurement. The recording duration is typically in the order of several milliseconds and amounts to 40 ms for the exemplary parametrization shown in Tab. 5.3b. The resulting dataset is often referred to as a *single snapshot* and it is fully independent from previous or following measurement cycles. Temporal filtering and tracking algorithms often take advantage from this fact, because they can easily filter out one-time occurring noise and interference events.

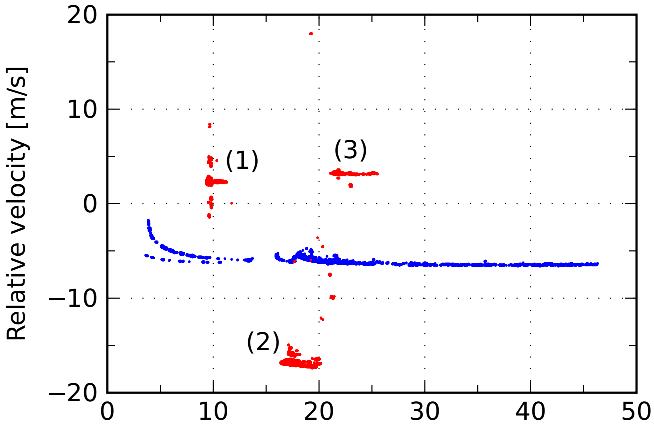
The analysis of a single measurement snapshot is a convenient approach to benchmark the detection and resolution capabilities of a radar system without the influence of enhanced perception algorithms. A three-dimensional point representation of the environment can already be computed and gives a first insight on the provided data quality. Certainly, a multitude of noise and interference components are still included in the data, however the scenery can be interpreted by humans already.

An example measurement, performed with the radar prototype which was introduced in section 5.1, is presented on page 150. Please refer to Fig. 5.3 for details about the radar sensor's mounting position and modulation parameters. In Fig. 5.4, two different data representations are shown, the bird's-eye view in **(a)** and the range-Doppler view in **(b)**. Static and moving targets can be easily separated due to the highly accurate Doppler information and the knowledge of the own vehicle's speed. The moving objects are shown with a red color and they manifest themselves as dense point clusters. Even the shape and orientation of the vehicles are directly recognizable from this measurement which demonstrates the improved resolution capabilities. The road side infrastructure is also very detailed and dense, so that a road trajectory could be estimated from a single snapshot.

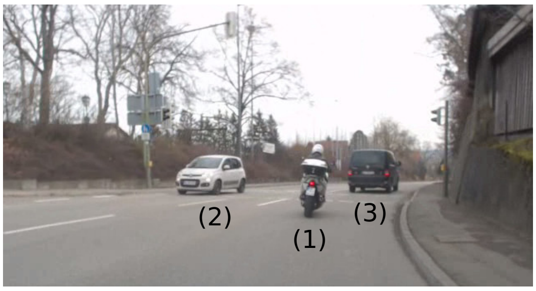
The accuracy of these measurements results is comparable to independent, recent work [140–142] and demonstrates the progress in automotive radar technology. It shows the effectiveness of an increased bandwidth in combination with the employment of MIMO systems with a large channel number. Such systems emphasize the requirement of a powerful, real-time capable and yet efficient processing unit. For instance, the measurement shown in Fig. 5.4 was obtained from raw data with the size of 128 MByte. Only with the help of more advanced hardware accelerators, it will become possible to handle such an increasing amount of data in a reasonable manner.



(a) Bird's-eye view



(b) Range-Doppler view



(c) Video image

Figure 5.4: Single radar measurement snapshot, recorded with the prototype setup shown in Fig. 5.3. Moving target detections are marked with red color. The three road users (1 – 3) can be recognized by dense point clusters.

5.3.2 Requirements for next generation radar systems

The implemented processing steps on the FPGA can reduce the raw data rate to a certain extent by separating the signals from the background noise. Accordingly, the appropriate output after the CFAR processing step are raw radar detections which need to be further processed. The subsequent algorithms like angle estimation will then operate on such lists of detections, which can vary in size. Thus, the runtime of the following units depends on the number of radar targets which are present in the vicinity of the sensor.

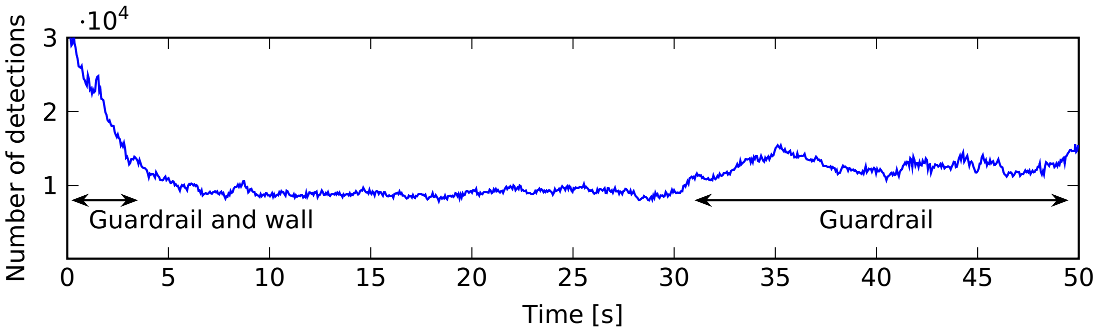
In practice, it is very difficult for the designer to ensure the real-time capability of the subsequent processing units due to the varying nature of the computational load. The peak number, and also the average number of targets has to be specified as accurately as possible in order to optimize the overall cost efficiency of the system. Early measurements with the envisaged final modulation settings are thus required to investigate the usual target number in real world scenarios.

The radar target count is somehow related to the number of objects in the field of view of the sensor. Though, specific physical effects like a micro-Doppler signature can cause a disproportionately high detection count on a single object. A spinning wheel may carry multiple detection points due to its rotation [7, 143]. Furthermore, roadside infrastructure with different radar reflection characteristics can contribute significantly to the final target count. It is thus reasonable to investigate different scenarios when determining an upper bound for the number of radar targets.

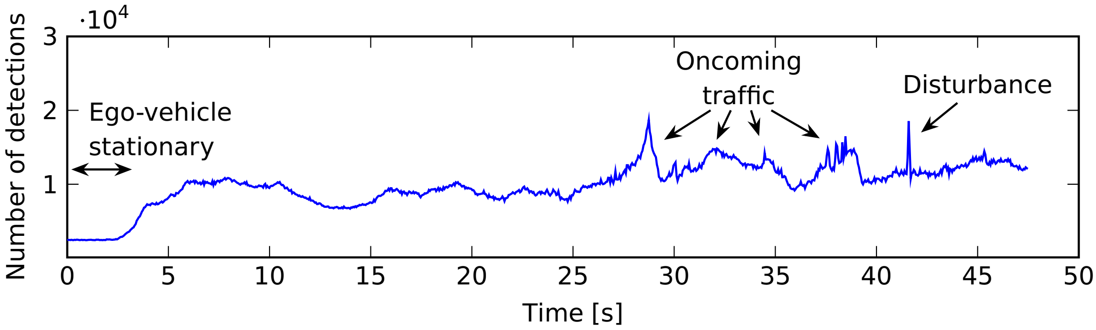
Fig. 5.5 shows the progression over time of the number of targets for two different measurement scenarios, one on a three-lane highway and another one on an interurban road. The same false alarm ratio as well as the same modulation settings were used during all measurements so that the number of targets depends solely on the surrounding traffic environment.

It can be observed that certain structures like a guardrail with a wall behind (cf Fig. 5.5a) contribute significantly to the total number of targets. Similarly, oncoming traffic in interurban scenarios causes substantial, fast varying peaks in the temporal progression. Interestingly, the target count in the case of a moving environment is heavily increased compared to the case of a stationary ego-vehicle. The reason for this behavior is that all roadside targets have a slightly different radial velocity component depending on their azimuthal position [144]. Hence, they can already be separated before the actual angle estimation takes place which shows the excellent separation capability of the radar sensor in the velocity domain.

The total number of targets lies for both scenarios in the order of 10 000 with peaks of up to 40 000 in special situations. Compared to the total number of frequency cells, which is approximately 2 millions for the used modulation, the number of cells occupied by actual targets is rather low. Only around one percent of the cells contain useful information, while the rest consists solely of noise components which are effectively filtered out by the CFAR processing. Consequently, the data rate of the system can be reduced by a factor of 100 or more, which means a significant reduction of the processing load for the downstream modules.



(a) Highway



(b) Interurban

Figure 5.5: Detection count over time for different traffic environments.

6

Conclusion

The current trend toward self-driving cars has a strong impact on the evolution of automotive radar sensors. Firstly, the requirements in terms of resolution are increasing significantly which in turn imposes a higher burden on the processing unit. At the same time, larger quantities of radar sensors are sold, which allows for a development of highly specialized and application specific integrated circuits (ASIC).

This work addresses one crucial aspect during the development and implementation of complex and specialized hardware accelerators: The selection and definition of an optimal architecture which fits as best as possible to all intended use cases. Answers and best approaches which support the design of a processing unit for automotive high-resolution MIMO radar systems have been provided step-by-step.

At first, the working principle and physical background of a radar system using an FMCW modulation was analyzed in chapter 2. Furthermore, state of the art technologies like chirp sequence modulation and MIMO antenna arrays were explained and the underlying processing algorithms were investigated. Important steps in the signal processing pipeline were identified, for instance the target detection which reduces the data rate to a significant extent.

The conducted study of algorithm and processing techniques has lead to the layout of a stream-based architecture, which is suitable for an implementation as a hardware accelerator. It was presented in chapter 3 and comprises all necessary steps in order to arrive at a list of target detections, while starting from the received signals which are sampled by the A/D converters. The steps are, in this order, two-dimensional spectral transformation based on FFTs, non-coherent integration (NCI), constant false alarm rate processing (CFAR), target detection and maximum likelihood (ML) angle estimation.

These individual tasks were decomposed in proper modules which can be parametrized according to the desired data throughput, numerical accuracy, spectral resolution, statistical performance as well as their resource and power consumption. For each module, several algorithmic and implementation specific characteristics were observed and certain optimizations were applied. Key findings are the importance of the rounding scheme for the 2D FFT, the influence of the addressing scheme on data throughput when using DRAM for the matrix transformation, the advantage of using NCI before the actual CFAR processing, the usefulness of a Cordic processor for the angle estimation as well as an improved scheduling scheme for the evaluation of symmetric data structures. These results were summarized, amongst others, in section 3.4 on page 98.

Based on the developed processing architecture, an implementation of all modules on an FPGA was realized in a subsequent step. In addition, an experimental radar system based on a Virtex 7 FPGA was built up so that a verification of all modules under real world conditions was possible. Measurement data which had been gathered and processed by the presented signal processing architecture was shown in section 5.3 on page 148.

Beside the actual realization of all processing blocks, an extensive design space exploration was performed in the course of this work. For all of the three modules comprising target detection, single and two target ML estimator, the implementation results on a Virtex 7 FPGA were investigated in-depth. More precisely, their resource and power consumption, maximum clock frequency, numerical accuracy and data throughput capabilities were measured for diverse configurations. In numbers, the obtained results are based on more than 3600 discrete module variants, all with a different parameter set. Starting from this comprehensive and accurate database, suitable model functions were selected and fitted to the observed characteristics. They were all presented, along with other interesting figures, in chapter 4.

With the help of the gathered results from this work, it becomes possible to discover and understand important relations between the individual design parameters, to find Pareto optimal implementations, to choose between different realization variants and to estimate the requirements for an arbitrary system configuration.

6.1 Future work

Starting from the outcome of this thesis, some directions for future work can be set. An extension of the obtained results for other modulation schemes or other target devices and especially ASICs seems reasonable.

A transition away from FMCW and toward more advanced modulation schemes like orthogonal frequency-division multiplexing (OFDM) will definitely entail changes in the initial processing steps of the spectral transformation [145]. New modules will have to be developed and investigated which account for the changes in the underlying processing schemes. The importance of such accelerators will further gain in importance as sampling rates can increase easily to one gigasample per second and beyond if no optimization measures on the system side are taken [146].

Although the provided results of the design space exploration can be used for the estimation and dimensioning of future radar systems, it should be kept in mind that all values were obtained from an FPGA-based implementation. It is expected that certain characteristics will change in the case of a custom implementation as part of an integrated circuit. Therefore, the behavior of the modules' resource and power consumption should be investigated again on the desired target architecture in order to provide more accurate results.

The presented signal processing pipeline comprises only the low-level processing tasks of an actual automotive radar system. Usually, the raw target detections are subsequently fed into some sort of clustering and tracking algorithms, a classification and segmentation step, as well as enhanced perception methods like a grid-based mapping of static objects. Most of these tasks benefit from the already reduced amount of data so that a real-time implementation is often feasible without further measures. Nevertheless, certain modules can still be quite demanding due to their sophisticated algorithms like for instance a semantic segmentation based on convolutional neural networks (CNNs), density-based clustering or an occupancy grid mapping. Such radar specific processing techniques are also candidates for a hardware acceleration and need to be further investigated in future work.

Appendix

A.1 FMCW baseband signal derivation

Recalling the expression for the transmitted and received waveform in the case of a single target from (2.5), page 9, where τ is the two-way propagation delay and m the ramp slope:

$$s_t(t) = A_t \cos\left(\phi_t\right) \tag{A.1}$$

$$\text{with} \quad \phi_t = 2\pi f_c t + \pi m t^2 + \phi_0 \tag{A.2}$$

$$s_r(t) = s_t(t-\tau) = A_r \cos\left(\phi_r\right) \tag{A.3}$$

$$\text{with} \quad \phi_r = 2\pi f_c (t-\tau) + \pi m (t-\tau)^2 + \phi_0 \tag{A.4}$$

At the receiver side, the two signals $s_t(t)$ and $s_r(t)$ are multiplied by using a frequency mixer to generate the baseband signal $s_b(t)$.

$$\begin{aligned} s_b(t) &= s_t(t)\, s_r(t) && \text{(A.5)} \\ &= \underbrace{\frac{1}{2} A_t A_r}_{A_b} \left(\cos\left(\underbrace{\phi_t - \phi_r}_{\phi_b} \right) + \underbrace{\cos\left(\phi_t + \phi_r\right)}_{=0} \right) && \text{(A.6)} \\ &= A_b \cos\left(\phi_b\right) && \text{(A.7)} \end{aligned}$$

The second cosine term can be neglected because its frequency is the sum of the signal frequencies of $s_t(t)$ and $s_r(t)$ which is simply filtered out. The received phase term now simplifies to the following expression:

$$\begin{aligned}\phi_b &= 2\pi f_c t + \pi m t^2 - 2\pi f_c t + 2\pi f_c \tau - \pi m t^2 + 2\pi m t \tau - \pi m \tau^2 && \text{(A.8)}\\ &= 2\pi \left(f_c \tau + m t \tau - \frac{m\tau^2}{2} \right) && \text{(A.9)}\end{aligned}$$

The propagation delay τ depends on the range and can be expressed by the following equation, where r remains constant over time and a possible radial movement is modeled by the radial velocity v_r.

$$\tau = \frac{2r}{c} + \frac{2v_r t}{c} = \frac{2}{c}\left(r + v_r t\right) \qquad \text{(A.10)}$$

Inserting (A.10) into (A.9) gives the following expression, which can be simplified by neglecting the minor terms in the sum. At this point, the advantage of using the center frequency f_c instead of f_0 as a parameter for the frequency ramp modeling can be observed. With this choice, the ramp duration and likewise the variable t is fixed to the interval $[-\frac{T}{2}, \frac{T}{2}]$. This minimizes the introduced errors due to the simplification of (A.11).

$$\begin{aligned}\phi_b &= 2\pi\frac{2}{c}\left(f_c r + f_c v_r t + m r t + m v_r t^2 - \frac{m}{2}\frac{2}{c}\left(r^2 + 2 r v_r t + v_r^2 t^2\right)\right) && \text{(A.11)}\\ &= 2\pi\frac{2}{c}\left(f_c r + f_c v_r t + m r t + m r t\frac{v_r t}{r} - m\frac{r^2}{c} - m r t\frac{2v_r}{c} - m r t\frac{v_r}{c}\frac{v_r t}{r}\right)\\ &= 2\pi\frac{2}{c}\left(f_c r\left(1 - m\frac{r}{f_c c}\right) + f_c v_r t + m r t\left(1 + \underbrace{\frac{v_r t}{r}}_{<<1} - \underbrace{\frac{2v_r}{c}}_{<<1} - \underbrace{\frac{v_r}{c}\frac{v_r t}{r}}_{<<1}\right)\right)\\ &= 2\pi\frac{2}{c}\left(f_c r\left(1 - \underbrace{\frac{B}{f_c}\frac{\frac{r}{c}}{T}}_{<<1}\right) + f_c v_r t + m r t\right)\\ &= 2\pi\frac{2}{c}\left(f_c r + f_c v_r t + m r t\right) && \text{(A.12)}\end{aligned}$$

A.2 FPGA silicon area estimation

For the design space exploration presented in chapter 4, one of the most important metrics is the area consumption of the implemented module. Unfortunately, the used FPGA tools only report the number of consumed FPGA building blocks like LUTs, DSP slices and memory blocks. This makes it cumbersome to compare different implementations because the overall circuit is composed of several different logic elements. Furthermore, a certain resource type could be replaced by another type while achieving the same functionality so that it is very important to observe the resource usage of all available building blocks. In order to form a single metric which estimates the total area consumption, an individual weighting factor for each resource category is required. With the help of such a coefficient, the individual blocks can be properly summed up according to their actual area contribution.

Besides, the coefficients could not only express the silicon area ratio between certain logic elements, but also the absolute silicon area per block expressed in mm^2. Such parameters are not provided directly in the datasheet, but they can be estimated to a certain extent based on other available information. The used coefficients in this work and their derivation is now described below.

For the used 7-series FPGA from Xilinx, the outer dimensions of the chip die of several different components can be found in the respective user guide [147]. The values are summarized in Tab. A.1. Even though the total die size of the actual used Virtex 7 device 485T is not included in this list, an estimation based on other devices should be sufficient for this work. All 7-series FPGAs are produced with a common 28 nm process [147, p. 308] so that each building block should provoke almost the same area footprint across different devices.

	Die size		
Device	w [mm]	h [mm]	Area [mm^2]
XC7K70T	5.99	9.68	57.98
XC7K160T	8.54	12.05	102.91
XC7K325T	9.87	16.92	167.00
XC7K410T	12.93	16.92	218.78
XC7A200T	11.10	12.05	133.76

Table A.1: FPGA die size

It is further known that the 7-series FPGAs are organized in arrays where every column can realize a different functionality [122, p. 16]. For instance a DSP column consists solely of DSP blocks which include a hardwired multiplier block. The number of elements per columns $n_{r,x}$ as well as the number of columns $n_{c,x}$ of a certain resource type x are also available from the manufacturer. The values can be found in [123, p. 19], [124, p. 14] or in the *Device* view of the Vivado tools. They are summarized in Tab. A.2.

	Number of columns $n_{c,x}$			Elements per column $n_{r,x}$		
Device	Slices	DSP	BRAM18	Slices	DSP	BRAM18
XC7K70T	62	3	4	200	80	80
XC7K160T	110	6	7	250	100	100
XC7K325T	154	6	7	350	140	140
XC7K410T	190	11	12	350	140	140
XC7A200T	164	9	9	250	100	100

Table A.2: Row and column layout for different devices

The height of each building block h_x can be estimated directly from the overall height h divided by the number of elements $n_{r,x}$ because one column is only occupied by a single resource type. Several variants of the following equation can be used to find a least squares solution for the h_x coefficients.

$$h = h_x \cdot n_{r,x} \tag{A.13}$$

The estimation of the width w_x of the building blocks is slightly more complex, because the overall width w of the device results from the mixture of different column types. Hence, an equation of the form below has to be solved.

$$w = w_{\text{SLICE}} \cdot n_{c,\text{SLICE}} + w_{\text{DSP}} \cdot n_{c,\text{DSP}} + w_{\text{BRAM}} \cdot n_{c,\text{BRAM}} + w_{\text{MISC}} \tag{A.14}$$

Beside the coefficients for the three major resource types Slices, DSP blocks and BRAMs, a fourth coefficient has been added which shall account for the additional area consumption caused by other blocks like for instance clocking resources, I/O banks and transceivers. It is assumed that this width contribution is the same for all FPGA devices. As a result, the equivalent column width for the miscellaneous resources is only counted once, i.e. $n_{c,\text{MISC}}$ is set to one for all equations.

Multiple equations like (A.14) are required, at least as many as w_x coefficients, so that a system of linear equations with a unique solution can be formed. With the available data from Tab. A.1 and Tab. A.2, the system of equations is overdetermined and a least squares solution was found. It is presented in Tab. A.3 along with the estimates for the h_x coefficients and the final area estimation of each building block.

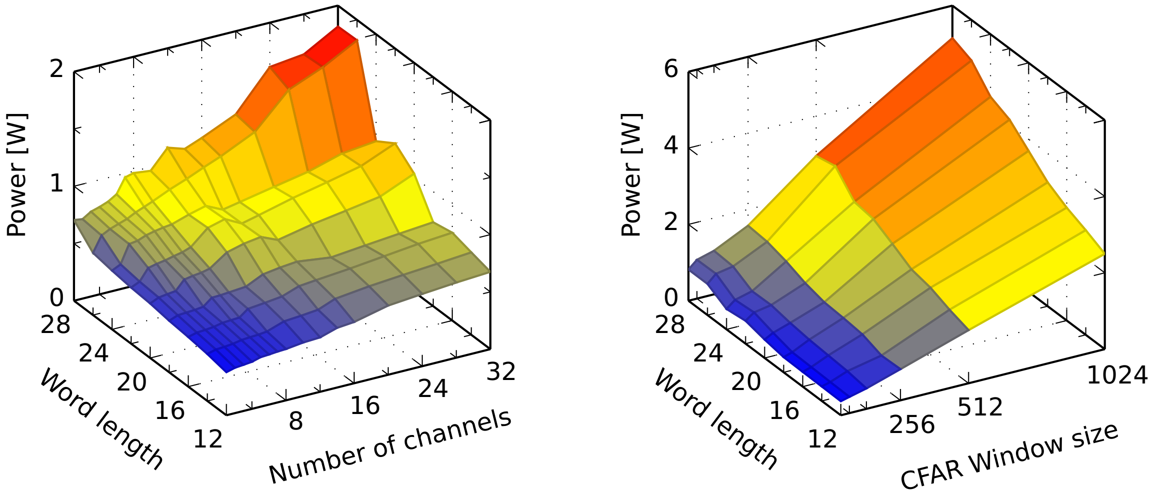
	This work			[148]
	Element width w_x [mm]	Element height h_x [mm]	Element area a_x [mm^2]	Element area a_x [mm^2]
Slice	0.0294	0.0483	0.00142	0.00108
DSP	0.1407	0.1208	0.01699	0.06533
BRAM18	0.2567	0.1208	0.03100	0.06166

Table A.3: Area estimation of the major building blocks of a Xilinx 7-series FPGA

In the rightmost column of Tab. A.3, the estimated values from previous work are given for a comparison. They are based on a simpler estimation method which assumes an equal area share between all three building blocks. Though, the values are quite similar and are within the same order of magnitude.

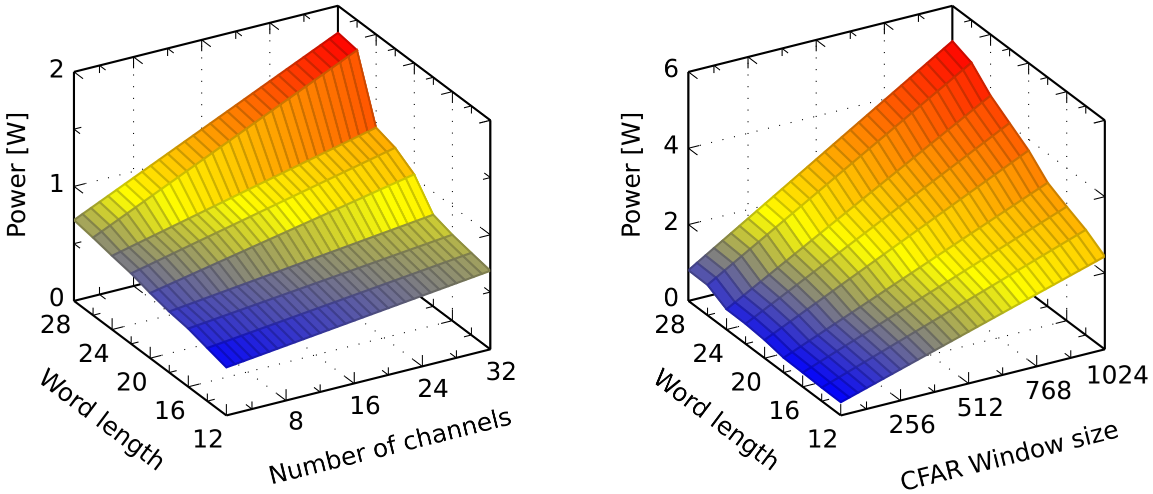
A.3 Power consumption data

A.3.1 Target detection module



(a) Observation
CFAR Window size (N) = 128

(b) Observation
Number of channels (M) = 16



(c) Model function, refer to (A.15)
CFAR Window size (N) = 128

(d) Model function, refer to (A.15)
Number of channels (M) = 16

Figure A.1: Power consumption of the target detection module on a Virtex 7 FPGA.

Model function

The following function is used to approximate the power dissipation of the target detection module.

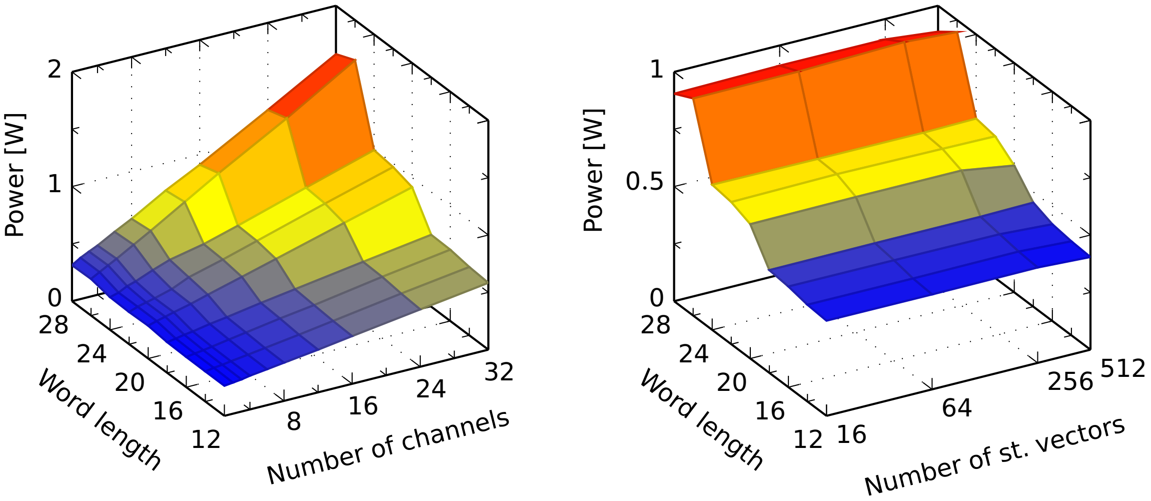
$$y_{Power}(M, N) = \zeta_M\, M + \zeta_N\, N + \zeta_{MN}\, M\, N + \zeta_0 \tag{A.15}$$

It is the same model function which has already been used for the silicon area, even though the coefficients are different. The function output, which can be seen in Fig. A.1c and Fig. A.1d, is very similar to the observations. The coefficients have been obtained separately for each word length b by a least square fitting and are presented in Tab. A.4.

Word size	Coefficients [mW]				GOF	
b	ζ_M	ζ_N	ζ_{MN}	ζ_0	s [mW]	R^2
12	6.40	1.83	$17.15 \cdot 10^{-3}$	177.0	79.2	0.988
14	6.20	2.17	$15.51 \cdot 10^{-3}$	177.1	82.9	0.990
16	6.57	2.48	$9.65 \cdot 10^{-3}$	183.1	78.4	0.993
18	6.29	2.58	$18.40 \cdot 10^{-3}$	192.9	77.2	0.994
20	11.16	2.80	$28.43 \cdot 10^{-3}$	195.1	102.2	0.992
22	13.42	3.14	$26.19 \cdot 10^{-3}$	196.6	143.2	0.987
24	12.96	3.33	$35.35 \cdot 10^{-3}$	200.2	120.1	0.992
26	30.19	3.63	$31.16 \cdot 10^{-3}$	179.3	133.7	0.992
28	30.19	3.81	$30.63 \cdot 10^{-3}$	183.6	154.9	0.990

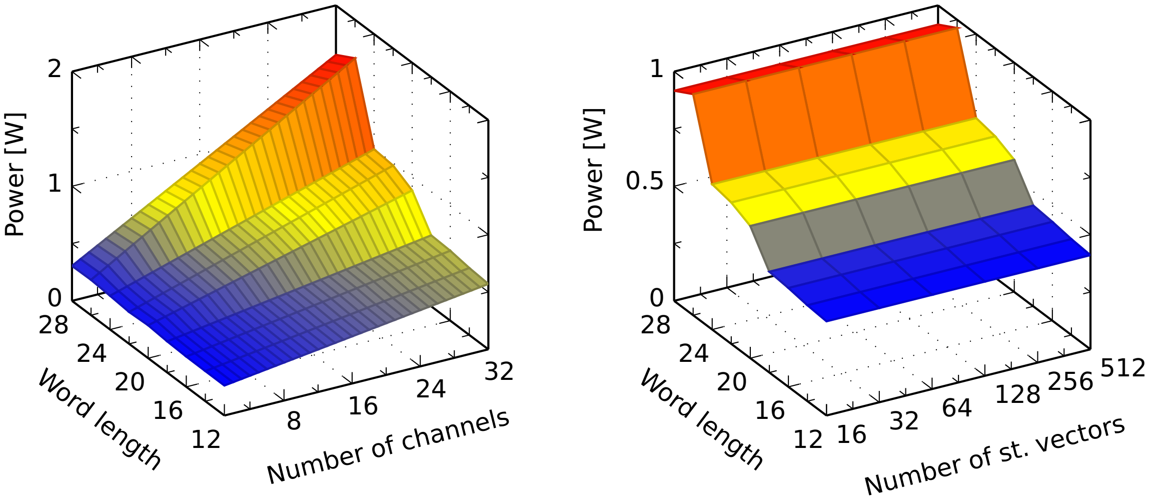
Table A.4: Coefficients of the model function (A.15) describing the power consumption of the target detection module

A.3.2 Single target ML estimator



(a) Observation
Number of st. vectors (N) = 256

(b) Observation
Number of channels (M) = 16



(c) Model function, refer to (A.16)
Number of st. vectors (N) = 256

(d) Model function, refer to (A.16)
Number of channels (M) = 16

Figure A.2: Power consumption of the single target ML angle estimation module on a Virtex 7 FPGA. All values are based on the DSP-3M implementation variant.

Model function

The following function is used to approximate the power dissipation of the single target ML angle estimation module.

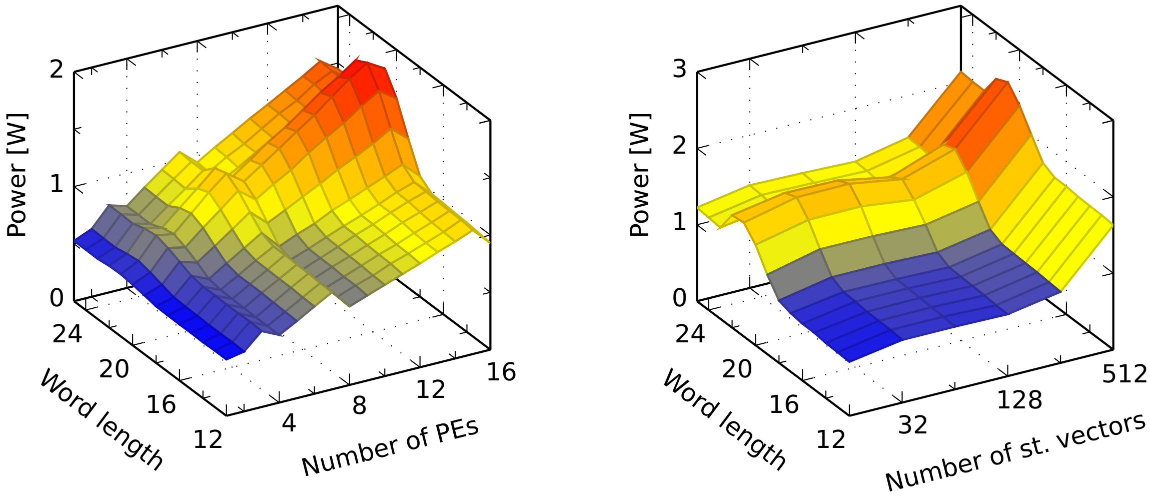
$$y_{Power}(M, N) = \eta_M \, M + \eta_0 \tag{A.16}$$

Again, the model function from the silicon area is reused, but with different coefficients. They have been obtained by a multiple linear regression (cf. section 4.1.3) and are listed in Tab. A.5. The low values of s and the high values of R^2 close to 1 indicate a good accuracy of the model, which can also be observed in the two graphical representations in Fig. A.2c and Fig. A.2d.

Word size	Coefficients [mW]		GOF	
b	η_M	η_0	s [mW]	R^2
12	9.96	251.4	4.05	0.998
14	10.40	251.9	4.02	0.999
16	11.17	252.5	5.69	0.997
18	11.22	260.1	7.68	0.995
20	19.34	267.1	10.24	0.997
22	22.59	255.1	7.79	0.999
24	22.63	271.5	11.09	0.998
26	43.74	264.4	15.36	0.999
28	40.78	264.1	9.76	0.999

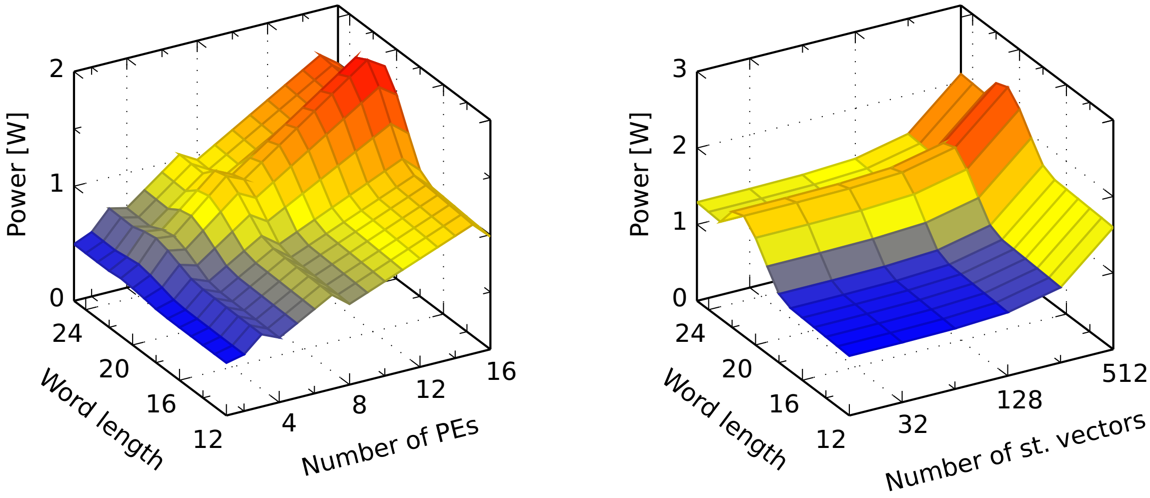
Table A.5: Coefficients of the model function (A.16) describing the power consumption of the single target ML angle estimation module

A.3.3 Two target ML estimator



(a) Observation
Number of st. vectors (N) = 256

(b) Observation
Number of PEs (K) = 16



(c) Model function, refer to (A.17)
Number of st. vectors (N) = 256

(d) Model function, refer to (A.17)
Number of PEs (K) = 16

Figure A.3: Power consumption of the two target ML angle estimation module on a Virtex 7 FPGA.

Model function

The following function is used to approximate the power dissipation of the two target ML angle estimation module.

$$y_{Power}(K, N) = \lambda_K K + \lambda_{KN} N^2 K 2^{-\lfloor \log_2 K \rfloor} + \lambda_0 \tag{A.17}$$

The function is identical to the model for the silicon area, but consists of different coefficients which are shown in Tab. A.6. The result of the function which can be seen in Fig. A.3c and Fig. A.3d resembles the observations very well.

Word size	Coefficients [mW]			GOF	
b	λ_K	λ_{KN}	λ_0	s [mW]	R^2
12	34.92	$3.111 \cdot 10^{-3}$	220.1	60.16	0.983
13	35.61	$3.116 \cdot 10^{-3}$	219.1	60.39	0.983
14	36.19	$3.113 \cdot 10^{-3}$	218.8	61.05	0.982
15	36.77	$3.113 \cdot 10^{-3}$	219.0	60.60	0.983
16	37.37	$3.111 \cdot 10^{-3}$	218.6	59.84	0.983
17	37.94	$3.105 \cdot 10^{-3}$	219.3	60.06	0.983
18	42.38	$3.119 \cdot 10^{-3}$	219.4	61.73	0.983
19	57.59	$3.114 \cdot 10^{-3}$	220.3	64.13	0.984
20	74.01	$3.123 \cdot 10^{-3}$	216.4	67.12	0.986
21	84.85	$3.122 \cdot 10^{-3}$	214.3	69.25	0.986
22	81.73	$3.115 \cdot 10^{-3}$	205.9	69.72	0.986
23	65.13	$3.105 \cdot 10^{-3}$	222.9	61.18	0.987
24	65.92	$3.105 \cdot 10^{-3}$	223.4	61.13	0.987
25	66.59	$3.103 \cdot 10^{-3}$	224.6	61.67	0.987

Table A.6: Coefficients of the model function (A.17) describing the power consumption of the two target ML angle estimation module

List of abbreviations

A/D	Analog-to-digital
ACC	Adaptive cruise control
ADAS	Advanced driver-assistance system
ADC	Analog-to-digital converter
AEB	Autonomous emergency braking
ASIC	Application-specific integrated circuit
BPSK	Binary phase-shift keying
BRAM	Block random-access memory
CA-CFAR	Cell-averaging constant false alarm rate
CDM	Code-division multiplexing
cf.	Confer
CFAR	Constant false alarm rate
CMLE	Conditional maximum likelihood estimation
CMOS	Complementary Metal-Oxide-Semiconductor
CNN	Convolutional neural network
CPU	Central processing unit
CUT	Cell under test
CW	Continuous wave
DFT	Discrete Fourier transform
DIF	Decimation-in-frequency
DIT	Decimation-in-time
DML	Deterministic maximum likelihood
DOA	Direction of arrival
DP	Double-precision
DRAM	Dynamic random-access memory
DSE	Design space exploration
DSP	Digital signal processing
DSP-3M,4M	Implementation variants, refer to p. 122

FDM	Frequency-division multiplexing
FFT	Fast Fourier transform
FIFO	First in, first out
FMCW	Frequency-modulated continuous wave
FPGA	Field-programmable gate array
FoV	Field of view
GOF	Goodness of fit
GPU	Graphics processing unit
GaAs	Gallium arsenide
IC	Integrated circuit
IF	Intermediate frequency
i.i.d.	Independent and identically distributed
LSB	Least significant bit
LUT-4M	Implementation variant, refer to p. 122
LUT	Lookup table
MDC	Multi-path delay commutator
MIMO	Multiple-input and multiple-output
MIPS	Million instructions per second
ML	Maximum likelihood
MMIC	Monolithic microwave integrated circuit
MSB	Most significant bit
MSE	Mean squared error
MSPS	Megasamples per second
NCI	Non-coherent integration
OFDM	Orthogonal frequency-division multiplexing
OS-CFAR	Ordered-statistic constant false alarm rate
PE	Processing element
PLL	Phase-locked loop
PSNR	Peak signal-to-noise ratio
r-D	Range-Doppler
RAM	Random-access memory
RF	Radio frequency
RSS	Residual sum of squares
RTL	Register-transfer level
Rx	Receive
SAR	Synthetic-aperture radar

SDF	Single-path delay feedback
SDRAM	Synchronous dynamic random-access memory
SIMD	Single instruction, Multiple data
SML	Statistical maximum likelihood
SNR	Signal-to-noise ratio
SP	Single-precision
SPI	Serial Peripheral Interface
SQNR	Signal-to-quantization-noise ratio
SiGe	Silicon-germanium
SoC	System on chip
TDM	Time-division multiplexing
TDP	Thermal design power
Tx	Transmit
ULA	Uniform linear array
UMLE	Unconditional maximum likelihood estimation
VCO	Voltage-controlled oscillator
VGA	Variable-gain amplifier
VHDL	Very High Speed Integrated Circuit Hardware Description Language

List of own publications

- F. Meinl, M. Kunert, and H. Blume, “Massively parallel signal processing challenges within a driver assistant prototype framework. First case study results with a novel MIMO-radar,” in *Proc. Int Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV) Conf*, Jul. 2014, pp. 351–357
- F. Meinl, E. Schubert, M. Kunert, and H. Blume, “Realtime FPGA-based processing unit for a high-resolution automotive MIMO radar platform,” in *Proc. European Radar Conf. (EuRAD)*, Sep. 2015, pp. 213–216
- F. Meinl, M. Kunert, and H. Blume, “Hardware acceleration of maximum-likelihood angle estimation for automotive MIMO radars,” in *Proc. Conf. Design and Architectures for Signal and Image Processing (DASIP)*, Oct. 2016, pp. 168–175
- F. Meinl, M. Stolz, M. Kunert, and H. Blume, “An experimental high performance radar system for highly automated driving,” in *Proc. IEEE MTT-S Int. Conf. Microwaves for Intelligent Mobility (ICMIM)*, Mar. 2017, pp. 71–74
- F. Meinl, E. Schubert, M. Kunert, and H. Blume, “Real-time data preprocessing for high-resolution MIMO radar sensors,” in *Towards a Common Software/Hardware Methodology for Future Advanced Driver Assistance Systems*, G. Payá-Vayá and H. Blume, Eds. River Publishers, 2017, ch. 7, pp. 133–156
- E. Schubert, F. Meinl, M. Kunert, and W. Menzel, “High resolution automotive radar measurements of vulnerable road users – Pedestrians & cyclists,” in *Proc. IEEE MTT-S Int Microwaves for Intelligent Mobility (ICMIM) Conf*, Apr. 2015, pp. 1–4

- E. Schubert, M. Kunert, F. Meinl, and W. Menzel, “Target modeling and deduction of automotive radar resolution requirements for pedestrian classification,” *International Journal of Microwave and Wireless Technologies*, vol. 7, no. 3-4, pp. 433 – 441, Apr. 2015
- E. Schubert, F. Meinl, M. Kunert, and W. Menzel, “Clustering of high resolution automotive radar detections and subsequent feature extraction for classification of road users,” in *Proc. 16th Int. Radar Symp. (IRS)*, Jun. 2015, pp. 174 –179
- M. Stolz, E. Schubert, F. Meinl, M. Kunert, and W. Menzel, “Multi-target reflection point model of cyclists for automotive radar,” in *Proc. European Radar Conf. (EURAD)*, Oct. 2017, pp. 94 – 97
- M. Stolz, M. Wolf, F. Meinl, M. Kunert, and W. Menzel, “A new antenna array and signal processing concept for an automotive 4D radar,” in *Proc. 15th European Radar Conf. (EuRAD)*, Sep. 2018, pp. 63 – 66

Bibliography

[1] J. Lombacher, M. Hahn, J. Dickmann, and C. Wöhler, "Potential of radar for static object classification using deep learning methods," in *Proc. IEEE MTT-S Int. Conf. Microwaves for Intelligent Mobility (ICMIM)*, May 2016, pp. 1–4.

[2] J. Dickmann, J. Klappstein, M. Hahn, N. Appenrodt, H. Bloecher, K. Werber, and A. Sailer, "Automotive radar the key technology for autonomous driving: From detection and ranging to environmental understanding," in *Proc. IEEE Radar Conf. (RadarConf)*, May 2016, pp. 1–6.

[3] H. Winner, S. Hakuli, F. Lotz, and C. Singer, Eds., *Handbook of Driver Assistance Systems*. Springer International Publishing, 2016.

[4] P. Swami, A. Jain, P. Goswami, K. Chitnis, A. Dubey, and P. Chaudhari, "High performance automotive radar signal processing on TI's TDA3X platform," in *Proc. IEEE Radar Conf. (RadarConf)*, May 2017, pp. 1317–1320.

[5] H. Blume, H. Hubert, H. T. Feldkamper, and T. G. Noll, "Model-based exploration of the design space for heterogeneous systems on chip," in *Proc. IEEE Int. Conf. Application- Specific Systems, Architectures and Processors*, Jul. 2002, pp. 29–40.

[6] E. Schubert, F. Meinl, M. Kunert, and W. Menzel, "Clustering of high resolution automotive radar detections and subsequent feature extraction for classification of road users," in *Proc. 16th Int. Radar Symp. (IRS)*, Jun. 2015, pp. 174–179.

[7] E. Schubert, F. Meinl, M. Kunert, and W. Menzel, "High resolution automotive radar measurements of vulnerable road users – Pedestrians & cyclists," in *Proc. IEEE MTT-S Int Microwaves for Intelligent Mobility (ICMIM) Conf*, Apr. 2015, pp. 1–4.

[8] E. Schubert, M. Kunert, F. Meinl, and W. Menzel, "Target modeling and deduction of automotive radar resolution requirements for pedestrian classification," *International Journal of Microwave and Wireless Technologies*, vol. 7, no. 3-4, pp. 433–441, Apr. 2015.

[9] F. Meinl, E. Schubert, M. Kunert, and H. Blume, "Realtime FPGA-based processing unit for a high-resolution automotive MIMO radar platform," in *Proc. European Radar Conf. (EuRAD)*, Sep. 2015, pp. 213–216.

[10] T. Poguntke and K. Ochs, "Linear time-variant system identification using FMCW radar systems," in *Proc. IEEE 59th Int. Midwest Symp. Circuits and Systems (MWSCAS)*, Oct. 2016, pp. 1–4.

[11] F. Meinl, M. Stolz, M. Kunert, and H. Blume, "An experimental high performance radar system for highly automated driving," in *Proc. IEEE MTT-S Int. Conf. Microwaves for Intelligent Mobility (ICMIM)*, Mar. 2017, pp. 71–74.

[12] J. Schlichenmaier, N. Selvaraj, M. Stolz, and C. Waldschmidt, "Template matching for radar-based orientation and position estimation in automotive scenarios," in *Proc. IEEE MTT-S Int. Conf. Microwaves for Intelligent Mobility (ICMIM)*, Mar. 2017, pp. 95–98.

[13] M. Stolz, E. Schubert, F. Meinl, M. Kunert, and W. Menzel, "Multi-target reflection point model of cyclists for automotive radar," in *Proc. European Radar Conf. (EURAD)*, Oct. 2017, pp. 94–97.

[14] M. Li, Z. Feng, M. Stolz, M. Kunert, R. Henze, and F. Küçükay, "High resolution radar-based occupancy grid mapping and free space detection," in *Proc. 4th Int. Conf. Vehicle Technology and Intelligent Transport Systems (VEHITS)*, vol. 1, INSTICC. SciTePress, Mar. 2018, pp. 70–81.

[15] M. Stolz, M. Li, Z. Feng, M. Kunert, and W. Menzel, "Direction of movement estimation of cyclists with a high-resolution automotive radar," in *Proc. IEEE MTT-S Int. Conf. Microwaves for Intelligent Mobility (ICMIM)*, Apr. 2018, pp. 1–4.

[16] M. Stolz, M. Wolf, F. Meinl, M. Kunert, and W. Menzel, "A new antenna array and signal processing concept for an automotive 4D radar," in *Proc. 15th European Radar Conf. (EuRAD)*, Sep. 2018, pp. 63–66.

[17] J. Schlichenmaier, L. Yan, M. Stolz, and C. Waldschmidt, "Instantaneous actual motion estimation with a single high-resolution radar sensor," in *Proc. IEEE MTT-S Int. Conf. Microwaves for Intelligent Mobility (ICMIM)*, Apr. 2018, pp. 1–4.

[18] Z. Feng, M. Stolz, M. Li, M. Kunert, and W. Wiesbeck, "Verification of a lane detection method with automotive radar based on a new type of road marking," in *Proc. IEEE MTT-S Int. Conf. Microwaves for Intelligent Mobility (ICMIM)*, Apr. 2018, pp. 1–4.

[19] A. Aparicio, I. Cieslik, J. Stoll, M. Kunert, F. Flohr, M. Arbitmann, T. Wimmer, J. Bräutigam, and D. Gavrila, "Advancing active safety towards the protection of vulnerable road users by evolution of ADAS solutions that meet real-world deployment challenges: The project PROSPECT," in *Proc. Transport Research Arena (TRA)*. Zenodo, Apr. 2018.

[20] M. Stolz, M. Li, Z. Feng, M. Kunert, and W. Menzel, "High resolution automotive radar data clustering with novel cluster method," in *Proc. IEEE Radar Conf. (RadarConf18)*, Apr. 2018, pp. 0164–0168.

[21] E. Schubert, "Klassifikation leicht verwundbarer Verkehrsteilnehmer mit hochauflösendem Automobilradar," Ph.D. dissertation, Universität Ulm, 2018.

[22] M. Li, M. Stolz, Z. Feng, M. Kunert, R. Henze, and F. Küçükay, "An adaptive 3D grid-based clustering algorithm for automotive high resolution radar sensor," in *Proc. IEEE Int. Conf. Vehicular Electronics and Safety (ICVES)*, Sep. 2018, pp. 1–7.

[23] Z. Feng, M. Li, M. Stolz, M. Kunert, and W. Wiesbeck, "Lane detection with a high-resolution automotive radar by introducing a new type of road marking," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–18, 2018.

[24] M. Skolnik, *Introduction to Radar Systems*, ser. Electrical engineering series. McGraw-Hill, 2001.

[25] H. P. Forstner, H. Knapp, H. Jager, E. Kolmhofer, J. Platz, F. Starzer, M. Treml, A. Schinko, G. Birschkus, J. Bock, K. Aufinger, R. Lachner,

T. Meister, H. Schafer, D. Lukashevich, S. Boguth, A. Fischer, F. Reininger, L. Maurer, J. Minichshofer, and D. Steinbuch, "A 77GHz 4-channel automotive radar transceiver in SiGe," in *Proc. IEEE Radio Frequency Integrated Circuits Symp*, Jun. 2008, pp. 233–236.

[26] F. Meinl, M. Kunert, and H. Blume, "Hardware acceleration of maximum-likelihood angle estimation for automotive MIMO radars," in *Proc. Conf. Design and Architectures for Signal and Image Processing (DASIP)*, Oct. 2016, pp. 168–175.

[27] F. Meinl, E. Schubert, M. Kunert, and H. Blume, "Real-time data preprocessing for high-resolution MIMO radar sensors," in *Towards a Common Software/Hardware Methodology for Future Advanced Driver Assistance Systems*, G. Payá-Vayá and H. Blume, Eds. River Publishers, 2017, ch. 7, pp. 133 – 156.

[28] M. A. Richards, *Fundamentals of Radar Signal Processing*. Tata McGraw-Hill Education, 2005.

[29] A. G. Stove, "Linear FMCW radar techniques," *IEE Proceedings F-Radar and Signal Processing*, vol. 139, no. 5, pp. 343–350, Oct. 1992.

[30] M. Reiher, "Optimierung von Sendesignalen zur Vermeidung von Scheinzielen für frequenzmodulierte Dauerstrich-Radarsysteme im Automobil," Ph.D. dissertation, Universität Stuttgart, Feb. 2012.

[31] S. Lutz, D. Ellenrieder, T. Walter, and R. Weigel, "On fast chirp modulations and compressed sensing for automotive radar applications," in *Proc. 15th Int. Radar Symp. (IRS)*, Jun. 2014, pp. 1–6.

[32] V. Winkler, "Range Doppler detection for automotive FMCW radars," in *Proc. European Radar Conf*, Oct. 2007, pp. 166–169.

[33] F. Roos, D. Ellenrieder, N. Appenrodt, J. Dickmann, and C. Waldschmidt, "Range migration compensation for chirp-sequence based radar," in *Proc. German Microwave Conf. (GeMiC)*, Mar. 2016, pp. 317–320.

[34] M. Andres, W. Menzel, H. L. Bloecher, and J. Dickmann, "Detection of slow moving targets using automotive radar sensors," in *Proc. The 7th German Microwave Conf*, Mar. 2012, pp. 1–4.

[35] T. Glisson, C. Black, and A. Sage, "The digital computation of discrete spectra using the fast Fourier transform," *IEEE Transactions on Audio and Electroacoustics*, vol. 18, no. 3, pp. 271–287, Sep. 1970.

[36] F. Meinl, M. Kunert, and H. Blume, "Massively parallel signal processing challenges within a driver assistant prototype framework. First case study results with a novel MIMO-radar," in *Proc. Int Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV) Conf*, Jul. 2014, pp. 351–357.

[37] J. B. Billingsley, A. Farina, F. Gini, M. V. Greco, and L. Verrazzani, "Statistical analyses of measured radar ground clutter data," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 35, no. 2, pp. 579–593, Apr. 1999.

[38] M. A. Richards, "The discrete-time Fourier transform and discrete Fourier transform of windowed stationary white noise," Georgia Institute of Technology, Tech. Rep., 2007.

[39] H. Rohling, "Radar CFAR thresholding in clutter and multiple target situations," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-19, no. 4, pp. 608–621, Jul. 1983.

[40] H. Rohling and R. Mende, "OS CFAR performance in a 77 GHz radar sensor for car application," in *Proc. Int. Radar Conf*, Oct. 1996, pp. 109–114.

[41] M. Shbat and V. Tuzlukov, "Evaluation of detection performance under employment of the generalized detector in radar sensor systems," *Radioengineering*, vol. 23, no. 1, pp. 50–65, 2014.

[42] M. A. Richards, "Noncoherent integration gain, and its approximation," Georgia Institute of Technology, Tech. Rep., Jun. 2010.

[43] A. J. Fenn, D. H. Temme, W. P. Delaney, and W. E. Courtney, "The development of phased-array radar technology," *Lincoln Laboratory Journal*, vol. 12, no. 2, pp. 321–340, 2000.

[44] M. Soumekh, *Synthetic Aperture Radar Signal Processing*. New York: Wiley, 1999, vol. 7.

[45] H. Wu and T. Zwick, "Automotive SAR for parking lot detection," in *Proc. German Microwave Conf*, Mar. 2009, pp. 1–8.

[46] H. L. V. Trees, *Optimum Array Processing*, ser. Detection, Estimation, and Modulation Theory. John Wiley & Sons, Inc., Mar. 2002, vol. 4.

[47] H. Krim and M. Viberg, "Two decades of array signal processing research: The parametric approach," *IEEE Signal Processing Magazine*, vol. 13, no. 4, pp. 67–94, Jul. 1996.

[48] A. J. Paulraj, D. A. Gore, R. U. Nabar, and H. Bolcskei, "An overview of MIMO communications - A key to gigabit wireless," *Proceedings of the IEEE*, vol. 92, no. 2, pp. 198–218, Feb. 2004.

[49] M. Zatman, "How narrow is narrowband?" *Sonar and Navigation IEE Proceedings-Radar*, vol. 145, no. 2, pp. 85–91, Apr. 1998.

[50] M. Gardill, "Characterization and design of small array antennas for direction-of-arrival estimation for ultra-wideband industrial FMCW radar systems," Ph.D. dissertation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Aug. 2015.

[51] K. Carver and J. Mink, "Microstrip antenna technology," *IEEE Transactions on Antennas and Propagation*, vol. 29, no. 1, pp. 2–24, Jan. 1981.

[52] M. Viberg, M. Lanne, and A. Lundgren, "Calibration in array processing," in *Classical and Modern Direction-of-Arrival Estimation*, T. E. Tuncer and B. Friedlander, Eds. Boston: Academic Press, 2009, ch. 3, pp. 93 – 124.

[53] F. Vincent, O. Besson, and E. Chaumette, "Approximate maximum likelihood estimation of two closely spaced sources," *Signal Processing*, vol. 97, pp. 83 – 90, Apr. 2014.

[54] P. Häcker and B. Yang, "Single snapshot DOA estimation," *Advances in Radio Science*, vol. 8, pp. 251–256, oct 2010.

[55] I. Ziskind and M. Wax, "Maximum likelihood localization of multiple sources by alternating projection," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 10, pp. 1553–1560, Oct. 1988.

[56] J. Capon, "High-resolution frequency-wavenumber spectrum analysis," *Proceedings of the IEEE*, vol. 57, no. 8, pp. 1408–1418, Aug. 1969.

[57] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 276–280, Mar. 1986.

[58] R. Roy and T. Kailath, "ESPRIT-estimation of signal parameters via rotational invariance techniques," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 7, pp. 984–995, Jul. 1989.

[59] A. Barabell, "Improving the resolution performance of eigenstructure-based direction-finding algorithms," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, vol. 8, Apr. 1983, pp. 336–339.

[60] P. Heidenreich, "Antenna array processing: Autocalibration and fast high-resolution methods for automotive radar," Ph.D. dissertation, Technische Universität Darmstadt, August 2012.

[61] M. Schoor, "Hochauflösende Winkelschätzung für automobile Radarsysteme," Ph.D. dissertation, Universität Stuttgart, 2010.

[62] M. Wax and T. Kailath, "Detection of signals by information theoretic criteria," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 33, no. 2, pp. 387–392, Apr. 1985.

[63] P. Stoica and Y. Selen, "Model-order selection: A review of information criterion rules," *IEEE Signal Processing Magazine*, vol. 21, no. 4, pp. 36–47, Jul. 2004.

[64] J. Li and P. Stoica, *MIMO Radar Signal Processing*. John Wiley & Sons Inc., 2008.

[65] F. C. Robey, S. Coutts, D. Weikle, J. C. McHarg, and K. Cuomo, "MIMO radar theory and experimental results," in *Proc. of the Thirty-Eighth Asilomar Conf. Signals, Systems and Computers*, vol. 1, Nov. 2004, pp. 300–304 Vol.1.

[66] R. Feger, C. Pfeffer, and A. Stelzer, “A frequency-division MIMO FMCW radar system based on delta –sigma modulated transmitters,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 62, no. 12, pp. 3572–3581, Dec. 2014.

[67] R. Feger, H. Haderer, and A. Stelzer, “Optimization of codes and weighting functions for binary phase-coded FMCW MIMO radars,” in *Proc. IEEE MTT-S Int. Conf. Microwaves for Intelligent Mobility (ICMIM)*, May 2016, pp. 1–4.

[68] C. M. Schmid, R. Feger, C. Pfeffer, and A. Stelzer, “Motion compensation and efficient array design for TDMA FMCW MIMO radar systems,” in *Proc. 6th European Conf. Antennas and Propagation (EUCAP)*, Mar. 2012, pp. 1746–1750.

[69] J. Bechter, F. Roos, and C. Waldschmidt, “Compensation of motion-induced phase errors in TDM MIMO radars,” *IEEE Microwave and Wireless Components Letters*, vol. 27, no. 12, pp. 1164 –1166, Dec. 2017.

[70] Bosch Media Service, “Bosch presents new radar sensor,” Press release, Aug. 2013, PI 8268 CC Ks/af.

[71] Bosch Media Service, “Did you know that... facts about automated driving and parking,” Press release, Mar. 2017, PI 9617 CC joe/BT.

[72] Bosch Media Service, “New conceptions of mobility, new perceptions of technology: Bosch is driving transformation,” Annual Press Conference, May 2017, RF 9565-de my/ajo.

[73] J. Hasch, E. Topak, R. Schnabel, T. Zwick, R. Weigel, and C. Waldschmidt, “Millimeter-wave technology for automotive radar sensors in the 77 GHz frequency band,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 60, no. 3, pp. 845–860, Mar. 2012.

[74] V. Giannini, D. Guermandi, Q. Shi, A. Medra, W. V. Thillo, A. Bourdoux, and P. Wambacq, “A 79 GHz phase-modulated 4 GHz-BW CW radar transmitter in 28 nm CMOS,” *IEEE Journal of Solid-State Circuits*, vol. 49, no. 12, pp. 2925–2937, Dec. 2014.

[75] Federal Communications Commission, "FCC unlocks new airwaves for vehicular radar use (FCC-17-94)," Jul. 2017.

[76] *Highly integrated automotive radar MCU for advanced driver assistance systems (MPC577xK-MCU - Fact Sheet REV 1)*, Freescale Semiconductor, Inc., Jun. 2013.

[77] *MPC5775K Data Sheet (Document Number MPC5775K - Rev. 9.1)*, NXP Semiconductors, Oct. 2016.

[78] *TC297TA - AURIX family. Dedicated to driver assistance systems. (Product Brief v01_00)*, Infineon Technologies AG, Aug. 2016.

[79] J. Singh, B. Ginsburg, S. Rao, and K. Ramasubramanian, *AWR1642 mmWave sensor: 76–81-GHz radar-on-chip for short-range radar applications (White Paper SPYY006)*, Texas Instruments, Inc., May 2017.

[80] *AWR1642 Single-Chip 77- and 79-GHz FMCW Radar Sensor (Datasheet SWRS203A - A Revision)*, Texas Instruments, Inc., Apr. 2018.

[81] F. B. Khalid, D. T. Nugraha, A. Roger, and R. Ygnace, "Integrated target detection hardware unit for automotive radar applications," in *Proc. European Radar Conf. (EURAD)*, Oct. 2017, pp. 219–222.

[82] S. Langemeyer, "Architektur eines 2D-FFT-Coprozessor-Systems für die Echtzeit-SAR-Bilddatenverarbeitung," Ph.D. dissertation, Leibniz Universität Hannover, 2011.

[83] V. Winkler, J. Detlefsen, U. Siart, J. Buchler, and M. Wagner, "FPGA-based signal processing of an automotive radar sensor," in *Proc. EURAD. First European Radar Conf*, Oct. 2004, pp. 245–248.

[84] M. R. Bales, T. Benson, R. Dickerson, D. Campbell, R. Hersey, and E. Culpepper, "Real-time implementations of ordered-statistic CFAR," in *Proc. IEEE Radar Conf*, May 2012, pp. 0896–0901.

[85] B. Magaz and M. L. Bencheikh, "An efficient FPGA implementation of the OS-CFAR processor," in *Proc. Int. Radar Symp*, May 2008, pp. 1–4.

[86] R. Perez-Andrade, R. Cumplido, C. Feregrino-Uribe, and F. M. Del Campo, "A versatile hardware architecture for a constant false alarm rate processor based on a linear insertion sorter," *Digital Signal Processing*, vol. 20, no. 6, pp. 1733–1747, 2010.

[87] E. Seguin, R. Tessier, E. Knapp, and R. W. Jackson, "A dynamically-reconfigurable phased array radar processing system," in *Proc. 21st Int. Conf. Field Programmable Logic and Applications*, Sep. 2011, pp. 258–263.

[88] F. Winterstein, G. Sessler, M. Montagna, M. Mendijur, G. Dauron, and P. Besso, "Scalable front-end digital signal processing for a phased array radar demonstrator," in *Proc. 13th Int. Radar Symp*, May 2012, pp. 177–182.

[89] R. L. Walke, R. W. M. Smith, and G. Lightbody, "Architectures for adaptive weight calculation on ASIC and FPGA," in *Proc. of the Thirty-Third Asilomar Conf. Signals, Systems, and Computers*, vol. 2, Oct. 1999, pp. 1375–1380.

[90] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–297, may 1965.

[91] L. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, ser. Prentice-Hall signal processing series. Prentice-Hall, 1975.

[92] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementations," *IEEE Transactions on Computers*, vol. C-33, no. 5, pp. 414–426, May 1984.

[93] K. J. Yang, S. H. Tsai, and G. C. H. Chuang, "MDC FFT/IFFT processor with variable length for MIMO-OFDM systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 4, pp. 720–731, Apr. 2013.

[94] S. He and M. Torkelson, "A new approach to pipeline FFT processor," in *Proc. Int. Conf. Parallel Processing*, Apr. 1996, pp. 766–770.

[95] *Fast Fourier Transform v9.0 (LogiCORE IP Product Guide PG109)*, Xilinx, Inc., Oct. 2017.

[96] A. V. Oppenheim and C. J. Weinstein, "Effects of finite register length in digital filtering and the fast Fourier transform," *Proceedings of the IEEE*, vol. 60, no. 8, pp. 957–976, Aug. 1972.

[97] W. H. Chang and T. Q. Nguyen, "On the fixed-point accuracy analysis of FFT algorithms," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4673–4682, Oct. 2008.

[98] Tran-Thong and B. Liu, "Fixed-point fast Fourier transform error analysis," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 24, no. 6, pp. 563–573, Dec. 1976.

[99] M. D. Ercegovac and T. Lang, *Digital Arithmetic*, 1st ed. Morgan Kaufmann Publishers, Jun. 2003.

[100] C. J. Weinstein, "Quantization effects in digital filters," MIT Lincoln Laboratory, Lexington, MA, Tech. Rep., 1969.

[101] D. Menard, D. Novo, R. Rocher, F. Catthoor, and O. Sentieys, "Quantization mode opportunities in fixed-point system design," in *Proc. 18th European Signal Processing Conf*, Aug. 2010, pp. 542–546.

[102] C. Barnes, B. Tran, and S. Leung, "On the statistics of fixed-point roundoff error," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 33, no. 3, pp. 595–606, Jun. 1985.

[103] B. Widrow, I. Kollar, and M.-C. Liu, "Statistical theory of quantization," *IEEE Transactions on Instrumentation and Measurement*, vol. 45, no. 2, pp. 353–361, Apr. 1996.

[104] "IEEE standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1–70, Aug. 2008.

[105] D. Schlichthärle, *Digital Filters*. Springer-Verlag GmbH, 2011.

[106] L. Ecco and R. Ernst, "Improved DRAM timing bounds for real-time DRAM controllers with read/write bundling," in *Proc. IEEE Real-Time Systems Symp*, Dec. 2015, pp. 53–64.

[107] W. Shin, J. Jang, J. Choi, J. Suh, and L. S. Kim, "Bank-group level parallelism," *IEEE Transactions on Computers*, vol. 66, no. 8, pp. 1428–1434, Aug. 2017.

[108] *Automotive DDR4 SDRAM Data Sheet (8gb_auto_ddr4_dram.pdf - Rev. D)*, Micron Technology, Inc., Nov. 2017.

[109] Y. Dou, J. Zhou, Y. Lei, and X. Zhou, "FPGA SAR processor with window memory accesses," in *Proc. IEEE Int. Conf. Application-Specific Systems, Architectures and Processors (ASAP)*, Jul. 2007, pp. 95–100.

[110] B. Akin, P. A. Milder, F. Franchetti, and J. C. Hoe, "Memory bandwidth efficient two-dimensional fast Fourier transform algorithm and implementation for large problem sizes," in *Proc. IEEE 20th Int. Symp. Field-Programmable Custom Computing Machines*, Apr. 2012, pp. 188–191.

[111] R. Chen and V. K. Prasanna, "Energy optimizations for FPGA-based 2-D FFT architecture," in *Proc. IEEE High Performance Extreme Computing Conf. (HPEC)*, Sep. 2014, pp. 1–6.

[112] S. Langemeyer, P. Pirsch, and H. Blume, "Using SDRAMs for two-dimensional accesses of long $2^n \times 2^m$-point FFTs and transposing," in *Proc. Int. Conf. Embedded Computer Systems: Architectures, Modeling and Simulation*, Jul. 2011, pp. 242–248.

[113] J. Zhou, Y. Dou, Y. Lei, and Y. Dong, "Window memory accesses method in alternate row/column matrix access systems," in *Proc. 2nd Int. Conf. Computer Engineering and Technology*, vol. 3, Apr. 2010, pp. V3–201–V3–205.

[114] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, Jan. 1978.

[115] F. D. Enggar, A. M. Muthiah, O. D. Winarko, O. N. Samijayani, and S. Rahmatia, "Performance comparison of various windowing on FMCW radar signal processing," in *Proc. Int. Symp. Electronics and Smart Devices (ISESD)*, Nov. 2016, pp. 326–330.

[116] M. Pfitzner, "FPGA-basierte Hardware-Architektur für die Echtzeit-SAR-Bilddatengenerierung mit integrierter Motion Compensation," Ph.D. dissertation, Leibniz Universität Hannover, Jun. 2015.

[117] J. M. Simkins and B. D. Philofsky, "Structures and methods for implementing ternary adders/subtractors in programmable logic devices," U.S. patent US 7 274 211 B1, Sep. 25, 2007.

[118] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330–334, Sep. 1959.

[119] A. Mallik, J. Ryckaert, A. Mercha, D. Verkest, K. Ronse, and A. Thean, "Maintaining Moore's law: Enabling cost-friendly dimensional scaling," in *Proc. SPIE Extreme Ultraviolet (EUV) Lithography VI*, vol. 9422. International Society for Optics and Photonics, Apr. 2015, pp. 1N1–1N12.

[120] I. Kuon, R. Tessier, and J. Rose, "FPGA architecture: Survey and challenges," *Foundations and Trends in Electronic Design Automation*, vol. 2, no. 2, pp. 135–253, 2007.

[121] V. Sklyarov, I. Skliarova, A. Barkalov, and L. Titarenko, *Synthesis and Optimization of FPGA-Based Systems*. Springer International Publishing, 2014.

[122] *7 Series FPGAs Configurable Logic Block (User Guide UG474 v1.8)*, Xilinx, Inc., Sep. 2016.

[123] *7 Series DSP48E1 Slice (User Guide UG479 v1.10)*, Xilinx, Inc., Mar. 2018.

[124] *7 Series FPGAs Memory Resources (User Guide UG473 v1.12)*, Xilinx, Inc., Sep. 2016.

[125] F. Li, Y. Lin, L. He, D. Chen, and J. Cong, "Power modeling and characteristics of field programmable gate arrays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 11, pp. 1712–1724, Nov. 2005.

[126] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, Feb. 2007.

[127] J. Lever, M. Krzywinski, and N. Altman, “Model selection and overfitting,” *Nature Methods*, vol. 13, no. 9, pp. 703–704, Aug. 2016.

[128] D. M. Hawkins, “The problem of overfitting,” *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1, pp. 1–12, 2004, PMID: 14741005.

[129] M. A. Babyak, “What you see may not be what you get: A brief, nontechnical introduction to overfitting in regression-type models,” *Psychosomatic Medicine*, vol. 66, no. 3, 2004.

[130] J. O. Rawlings, S. G. Pantula, and D. A. Dickey, *Applied Regression Analysis: A Research Tool*, 2nd ed. Springer-Verlag, 1998.

[131] I. Pardoe, *Applied Regression Modeling*, 2nd ed. John Wiley & Sons, Inc., 2012.

[132] W. Xue and Y. Jiang, “Simulation of fast threshold CFAR processing algorithm with CUDA,” in *Proc. Int. Conf. Computer Distributed Control and Intelligent Environmental Monitoring*, Mar. 2012, pp. 621–624.

[133] *NVIDIA GeForce GTX 200 GPU Architectural Overview (Technical Brief TB-04044-001_v01)*, NVIDIA Corporation, May 2008.

[134] X. Wang, C. Chen, and W. Jiang, “Implementation of real-time LCMV adaptive digital beamforming technology,” in *Proc. Int. Conf. Electronics Technology (ICET)*, May 2018, pp. 134–137.

[135] S. Sana, S. Waqar, H. Yusaf, M. Waqas, and F. A. Siddiqui, “Software defined digital beam forming processor,” in *Proc. 13th Int. Bhurban Conf. Applied Sciences and Technology (IBCAST)*, Jan. 2016, pp. 671–676.

[136] D. Thompson, M. Yeary, B. Marr, and K. Prager, “Benchmarking 3D transistors for digital beamforming applications,” in *Proc. IEEE Radar Conf. (RadarConf)*, May 2017, pp. 1283–1286.

[137] H. Saheb and S. Haider, “Scalable high speed serial interface for data converters: Using the JESD204B industry standard,” in *Proc. 9th Int. Design and Test Symp. (IDT)*, Dec. 2014, pp. 6–11.

[138] R. Ammendola, A. Biagioni, G. Chiodi, O. Frezza, F. L. Cicero, A. Lonardo, R. Lunadei, P. S. Paolucci, D. Rossetti, A. Salamon, G. Salina, F. Simula, L. Tosoratto, and P. Vicini, “High-speed data transfer with FPGAs and QSFP+ modules,” *Journal of Instrumentation*, vol. 5, no. 12, Dec. 2010.

[139] *VC707 Evaluation Board for the Virtex-7 FPGA (User Guide UG885 v1.7.1)*, Xilinx, Inc., Aug. 2016.

[140] F. Roos, D. Kellner, J. Dickmann, and C. Waldschmidt, “Reliable orientation estimation of vehicles in high-resolution radar images,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 9, pp. 2986–2993, Sep. 2016.

[141] C. Sturm, G. Li, G. Heinrich, and U. Lübbert, “79 GHz wideband fast chirp automotive radar sensor with agile bandwidth,” in *Proc. IEEE MTT-S Int. Conf. Microwaves for Intelligent Mobility (ICMIM)*, May 2016, pp. 1–3.

[142] S. Brisken, F. Ruf, and F. Höhne, “Recent evolution of automotive imaging radar and its information content,” *IET Radar, Sonar & Navigation*, vol. 12, pp. 1078–1081, Oct. 2018.

[143] D. Kellner, M. Barjenbruch, J. Klappstein, J. Dickmann, and K. Dietmayer, “Wheel extraction based on micro doppler distribution using high-resolution radar,” in *Proc. IEEE MTT-S Int. Conf. Microwaves for Intelligent Mobility (ICMIM)*, Apr. 2015, pp. 1–4.

[144] D. Kellner, M. Barjenbruch, J. Klappstein, J. Dickmann, and K. Dietmayer, “Instantaneous ego-motion estimation using Doppler radar,” in *Proc. 16th Int. IEEE Conf. Intelligent Transportation Systems (ITSC 2013)*, Oct. 2013, pp. 869–874.

[145] G. Hakobyan, “Orthogonal frequency division multiplexing multiple-input multiple-output automotive radar with novel signal processing algorithms,” Ph.D. dissertation, Universität Stuttgart, May 2018.

[146] B. Schweizer, C. Knill, D. Schindler, and C. Waldschmidt, “Stepped-carrier OFDM-radar processing scheme to retrieve high-resolution range-velocity profile at low sampling rate,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 66, no. 3, pp. 1610–1618, Mar. 2018.

[147] *7 Series FPGAs Packaging and Pinout (Product Specification UG475 v1.17)*, Xilinx, Inc., Aug. 2018.

[148] C. Banz, *Design and Analysis of Architectures for Stereo Vision*, ser. Berichte aus der Informationstechnik. Shaker Verlag GmbH, Germany, 2013, Ph.D. dissertation.

Curriculum vitae

Frank Meinl
born 1988 in Munich, Germany

Education

05/2013 – 02/2020	**Leibniz University Hannover** Institute of Microelectronic Systems External PhD student; Supervisor: Prof. Holger Blume Degree: Doktor-Ingenieur (Dr.-Ing.)
10/2007 – 03/2013	**Technical University of Munich** Electrical Engineering and Computer Science Degree: Diplom-Ingenieur (Dipl.-Ing.)
09/2009 – 06/2011	**École Centrale Paris** Two-year double degree program "TIME" Degree: Diplôme d'Ingénieur

Employment

05/2013 –	**Robert Bosch GmbH**, Leonberg, Germany Chassis Systems Control, Advanced Engineering Radar Member of Technical Staff (05/2017 –) PhD candidate program (05/2013 – 04/2017)
09/2012 – 03/2013	**Infineon Technologies**, Neubiberg, Germany Chip Card & Security division Diploma thesis: "Benchmarking of Hybrid Hardware-Software Countermeasures against Fault Attacks"