7

Real-Time Data Preprocessing for High-Resolution MIMO Radar Sensors

Frank Meinl[1], Eugen Schubert[1], Martin Kunert[1] and Holger Blume[2]

[1]Advanced Engineering Sensor Systems, Robert Bosch GmbH, Leonberg, Germany
[2]Institute of Microelectronic Systems, Leibniz Universität Hannover, Hannover, Germany

7.1 Introduction

The progress in resolution of automotive radar sensors involves a considerable increase in data-rate and computational throughput. Dedicated processing architectures have to be investigated in order to manage the tremendous amount of data. Even for early prototype development platforms, the performance of existing PC-based frameworks and tools is no longer sufficient to cope with the data processing of many parallel radar receiver channels at very high sampling rates.

This chapter presents a FPGA-based signal processing architecture capable of handling 16 parallel MIMO radar receiving channels with a sampling frequency of 250 MHz each. Raw data is transferred from the AD-Converters to the FPGA where subsequent processing steps are performed, involving FIR-filtering and decimation, two-dimensional FFT transform, local noise level estimation and subsequent target detection. An external DRAM is used for storing multiple radar measurements which are finally evaluated altogether (so-called chirp-sequence modulation).

Data post-processing is outsourced onto a PC running with ADTF, an automotive framework for graph-based real-time data processing. The combination of a fast, FPGA-based preprocessing unit with a more flexible, PC-based development platform maximizes processing performance and minimizes

development time. The less mature angular MIMO processing algorithms can thus be evaluated with the help of C-based algorithms running in ADTF, while the simple, but calculation intensive FFT processing is implemented entirely as a hardware accelerator in a Virtex-7 FPGA device from Xilinx.

7.2 Signal Processing for Automotive Radar Sensors

After AD-conversion, the raw radar signals enter the processing unit, consecutively passing through all necessary signal processing steps. Different levels of data abstraction and representation can be identified, which range from low level time signals up to complex environmental models.

In this chapter, only the extraction of discrete scattering centers will be considered. The result is a list of reflections, each having multiple features, like for instance Cartesian coordinates, radar-cross-section (RCS), relative velocity or signal-to-noise ratio. Further processing of these reflections would incorporate clustering, classification and environment modeling.

An intermediate state is the extraction of relevant targets from the two-dimensional frequency spectrum (cf. Subsection 7.2.2). At this point, the range and velocity of the targets have already been determined, while the angular information is not evaluated yet. Nevertheless, the data rates are already reduced by a significant amount, so that at this stage the data transfer interface between FPGA and PC-based signal processing can be established.

7.2.1 FMCW Radar System Architecture

The usage of frequency-modulated continuous-wave (FMCW) radar sensors can be advantageous in short range applications, especially due to their high range resolution capability and much lower peak power requirements. In contrast to a pulsed radar system, the transmitter and receiver operate at the same time, which imposes some constraints on the transmitted signals. In order to measure the time-of-flight, i.e. the range towards an object, some kind of time-varying information needs to be added to the transmitted waveforms. The signal has to be modulated in an unambiguous, non-repetitive fashion. A constant sine wave, for instance, can't be used for range estimation, due to its ambiguity after the phase has increased by one cycle or 2π, respectively.

One widely used modulation scheme consists of linear modulated frequency chirps (cf. Figure 7.1). Two important parameters are the used bandwidth F and the modulation time T which determine the slope $\frac{F}{T}$ of

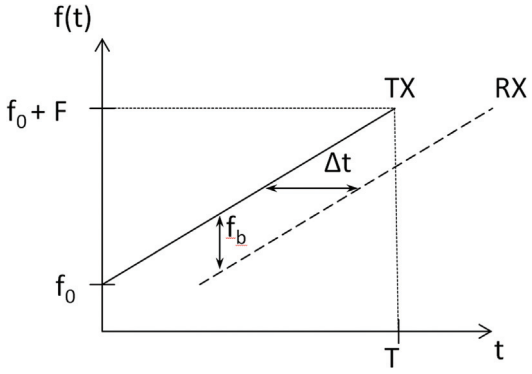

Figure 7.1 FMCW ramp waveform shown as frequency over time f(t). The solid line represents the transmitted signal (TX) while the dashed line is the received signal (RX).

the frequency ramp. Besides, other kinds of modulation schemes exist, e.g. frequency shift keying, various phase modulation or pseudo-noise coding principles.

In the case of a linear frequency modulation, the time-of-flight Δt can be directly translated into a frequency difference (so called beat frequency f_b). With the help of a mixer device in the receiver, this frequency difference can be measured efficiently and estimated by subsequent signal processing blocks. Finally, the target range r can be obtained from the estimated beat frequency value. However, as moving targets engender an additional frequency shift f_d (Doppler frequency), the measured frequency will consist of a superposition of a range and a velocity dependent component.

$$f_b = f_r + f_d = \frac{2r}{c}\frac{F}{T} - \frac{2v_r}{\lambda}$$

With the help of advanced modulation waveforms, the occurrence of range-Doppler ambiguities can be significantly reduced, while being able to estimate both frequency components individually at the same time [1]. This can be achieved by using multiple, aligned FMCW chirps. Furthermore, these ramp signals should have a very steep slope, so that the range dependent frequency part f_r dominates in the beat frequency f_b. For a sufficient small target velocity, the Doppler frequency f_d is likewise small enough so that the range estimation can be carried out directly from f_b by simply neglecting the minor f_d contribution. However, the Doppler information is not completely lost and can be regained from the inherent phase measurement which is present in the consecutive frequency ramps. For this purpose, it is necessary that the

ramp sequence is strictly aligned and that the data sampling occurs always at the same time instant w.r.t. the chirp modulation. The underlying processing technique is shown in Figure 7.2 and relies on a two-dimensional spectrum analysis. The big advantage is the unambiguous determination of both the range and velocity frequency component of each target.

For the angle estimation, two different measurement principles can be used. One possibility is a steerable antenna, which has a high directivity. Only targets which reside inside the antenna beam will contribute to the received signal in a significant manner. The detection space has to be scanned individually, i.e. each possible direction of arrival (DOA) will be measured separately. An alternative to a mechanical steered antenna is the use of an antenna array, where each antenna element is fed by a time delayed version of the transmit signal. The phase shift of the antenna feeds can be changed electronically. Depending on the phase relationships of the antenna elements, the directivity can be swiveled, which is also referred to as electronic beam steering or phased array.

The second class of angle estimation relies on a phase measurement of the received signals. Within a static antenna array, the measured phase differences will depend on the DOA of the target reflections. This property is exploited by many different algorithms in the field of array processing [2]. A major

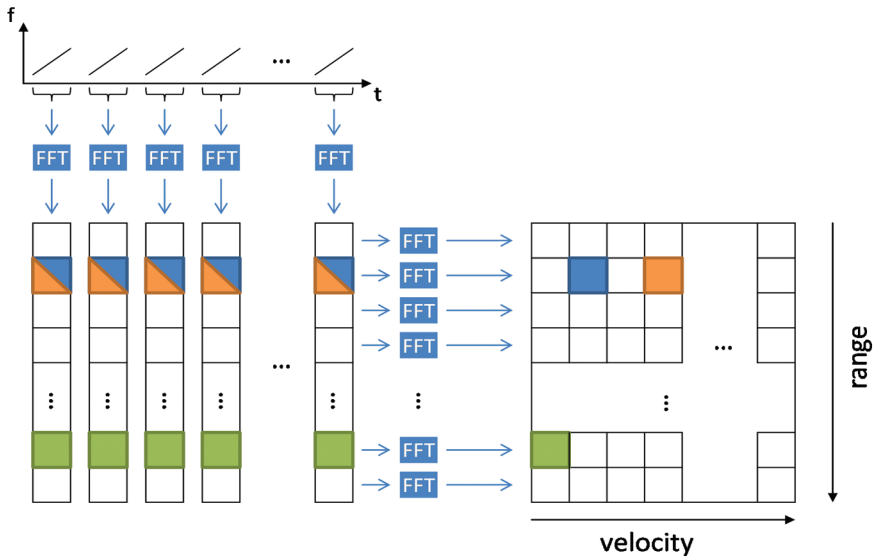

Figure 7.2 Chirp-sequence modulation.

advantage of a fixed antenna array is the simultaneous measurement over a wide opening angle. The region of interest does not have to be scanned and data can be collected in a single, instantaneous snapshot. In general, the achievable angular resolution and separability depends on the number of channels as well as on the aperture size of the array.

In the case of a receiving array, each channel will require a dedicated frequency mixer, amplifier and AD-converter, which increases the total cost of the system. Hence, the usage of advanced algorithms can be considered in order to increase resolution without additional receiving channels [3]. These algorithms are often said to achieve a superresolution because they perform better than a conventional Bartlett beamscan algorithm (cf. [2], pp. 1142). Another possibility is the usage of multiple transmitting channels (multiple input – multiple output – MIMO). A MIMO system has a better efficiency because the number of virtual channels is larger than the real number of channels, thus resulting in lower hardware effort.

In Figure 7.3, a linear MIMO antenna array is shown with two transmitter antennas, which are depicted as circles on the left. The physical receiving array (blue) is extended by several virtual antenna positions. The underlying signal processing remains the same as in the single transmitter case, however the full virtual array can be used resulting in an increased accuracy and object separation capability. In order to separate the signals originating from different transmitting antennas at the receiver side, some kind of orthogonality has to be introduced. A straight forward approach is to use a time-division multiplexing (TDM) approach, i.e. only one transmitter operates at the same time. Other possible techniques comprise frequency-division multiplexing (FDM) or code-division multiple access (CDMA).

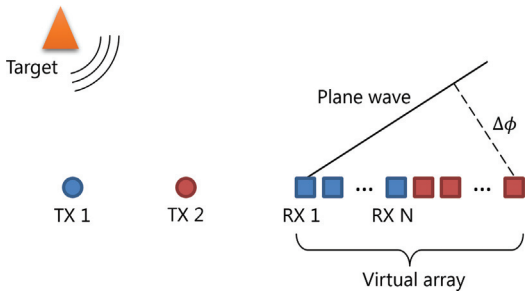

Figure 7.3 Possible MIMO antenna array design: The physical receiver array (blue) is extended by several virtual antennas (red squares) due to the second transmitter TX 2.

7.2.2 Two-Dimensional Spectrum Analysis for Range and Velocity Estimation

Multi-target scenarios are usually encountered in automotive radar applications. Especially static targets are often present in the field of view arising from roadside structures, e.g. guardrails and reflector posts. Furthermore, with increased resolution, multiple scattering centers are visible from single objects, e.g. the shape of car bodies is seen as a large cloud consisting of many reflections [4].

In order to resolve and separate proximate targets, a good range resolution and thus frequency resolution is required. One widely used technique providing a fast and robust frequency estimation is the fast Fourier transform (FFT). For a further increase in range resolution, advanced frequency estimation algorithms like autoregressive (AR) models or multiple signal classification (MUSIC) can be employed [5, 6]. Beside the higher computational requirements, they suffer from the fact that the number of detections needs to be known prior to the estimation. For this reason, the presented system relies on the more convenient FFT-based spectrum analysis.

The Doppler frequency estimation is carried out by a second FFT. Instead of the raw time signals, the frequency bins of the first FFT are used as input signal. In other words, the second FFT measures the ramp-to-ramp phase offset for each target. This offset depends solely on the Doppler shift of the target, because the radar system ensures a coherent sampling of the transmitted frequency chirps. Only if the target is moving relatively to the sensor, the measured phase value will vary between the consecutive chirp ramps.

As depicted in Figure 7.2, targets with different ranges and different velocities are separated after this step. In contrast to many other FMCW modulation forms, a matching step to find corresponding ranges and velocities is no longer required, because the values are directly obtained from the two-dimensional indices. Furthermore, the computational effort stays constant and is thus independent from the number of prevailing targets. This property plays a key role in scenarios with many scattering points as often encountered with high resolution automotive radar sensors.

Another benefit of the two-dimensional spectral processing is the higher sensitivity. Particularly small targets with a low radar cross-section (RCS) can be masked by the noise floor of the first FFT. These targets become visible only by the help of the additional processing gain of the second FFT. Thus, each output bin of the first FFT shall be taken into account and the full 2D matrix should be evaluated before any target detection takes place.

7.2.3 Thresholding and Target Detection

A crucial point in the signal processing chain is the separation of different target reflections in the two-dimensional power spectrum. With the help of this step, data of relevant objects will be isolated from the random noise components. This leads to a significant reduction of data rate and thus lowers the computational performance requirements for the downstream signal processing steps.

The target detection is carried out with the help of an adaptive threshold, reducing the effects of local noise and clutter components. With the means of a constant false alarm rate (CFAR) processing, the probability of false alarm remains constant, irrespective of varying operational and environmental conditions.

Different types of CFAR processors can be used for noise level estimation. Two variants are presented in this section, the cell-averaging (CA-CFAR) and the ordered-statistic (OS-CFAR), two of the most extensively used variants.

Cell-Averaging CFAR (CA-CFAR)

The basic task of a CFAR detector is to provide an adaptive threshold, which is then used for the subsequent detection step, i.e. the decision if a specific cell contains a present target or just irrelevant noise components. In contrast to a fixed threshold, an estimate of the local background noise level is used as threshold, which has to be obtained automatically and separately for each cell under test (CUT). Many different methods exist to provide such an estimate, each leading to different classes and variants of CFAR detectors.

A simple yet powerful approach is the mean value of a number of window cells in proximity to the CUT (see Figure 7.4). This variant is known as cell

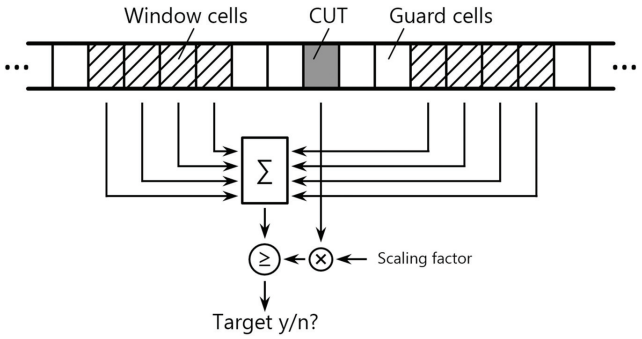

Figure 7.4 CA-CFAR sliding window implementation.

averaging CFAR, or CA-CFAR. The assumption made in this case is that all window cells contain only noise components and thus the mean value is a good estimate of the noise variance. In the case of white Gaussian noise, the value is corresponding to the maximum likelihood estimator. However, for many radar systems the assumption of normal distributed noise turns out to be inaccurate [7].

When designing a CFAR detector, an important parameter is the window size around the CUT. On the one hand, a larger window size reduces the statistical estimation error; on the other hand, local differences in the noise level can be blurred by a large window. A tradeoff has to be made between the deviation from the requested false alarm rate due to the estimation error and the local sensitivity of the adaptive threshold which results from smoothing. Furthermore, the computational effort becomes more relevant with increasing window sizes.

Ordered-Statistic CFAR (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 sortation of the values inside the window, similar to a median filter.

The algorithm performs the following steps for each cell under test (CUT):

- Sort all cells inside the window by their absolute square value
- Take out the k-th value of the sorted list. This value serves as an estimate for the local noise level
- Apply a scaling factor to the noise estimate
- Compare the scaled estimated noise value against the CUT
- Decide whether the CUT is a valid target

Especially in the field of high-resolution radar, big window sizes are required, because large and widespread targets will easily occupy multiple window cells. The complete sortation of the whole window is not a very efficient solution. Only a single value of the sorted list is of interest, while all other values are discarded. Furthermore, when evaluating neighboring CUTs, the previously sorted list can be used as starting point.

Several optimizations of the algorithm aim at these specific sortation 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 [8]. Another efficient realization uses a sliding window approach which keeps a sorted list in memory [9]. 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 sortation of the list can be bypassed and the detection step can be performed in a "rank-only" manner [10]. Therefore, the inverse threshold is applied to the CUT and the result is compared to each cell inside the window. The 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 rank is greater than k, the CUT is considered as valid detection. This approach is depicted in Figure 7.5.

In contrast to a complete sortation, this 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 unknown in the rank-only case. This value can serve as an estimate for the local noise level and can be required by subsequent signal processing blocks. A supplementary estimation of this value can be considered, e.g. the mean value of all cells which have been classified as noise.

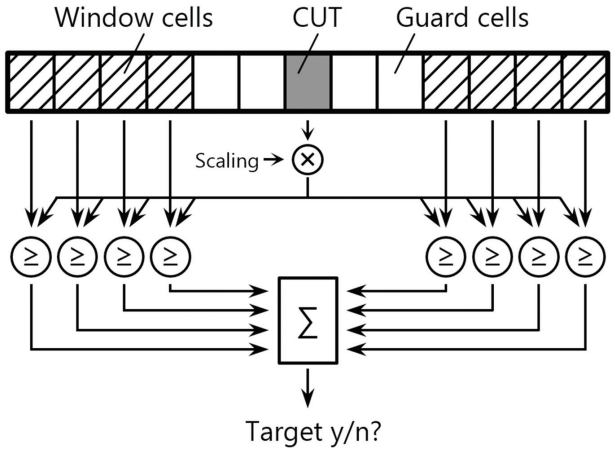

Figure 7.5 Rank-only OS-CFAR implementation.

Non-Coherent Integration (NCI)

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 of all channels prior to the detection step turns out to be beneficial, assuming that the noise components are independent and identically distributed (i.i.d.). However, the phase relationship of the signals between adjacent channels is not known prior to the angle estimation and can take any value. When summing up the complex values of each channel, 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).

In the following, 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 received signal s[t].

It can be shown, that both the real and imaginary parts of the noise components follow a zero-mean normal distribution after transformation into the frequency space [11]. The variance of N[k] depends on the input variance as well as on the length of the input signal, i.e. the length of the FFT. When taking longer signal sequences, the signal-to-noise ratio can be improved (so-called processing gain).

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

$$\downarrow$$

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

The power spectrum can be calculated by summing up the squared values of real and imaginary part. As a sum of two squared, i.i.d. Gaussian variables, it results a chi-squared distribution $\chi^2(n)$ with $n = 2$ degrees of freedom for the squared magnitude $|N[k]|^2$:

$$|N[k]|^2 = N_{Re}[k]^2 + N_{Im}[k]^2$$
$$|N[k]|^2 \sim \chi^2(2)$$

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

$$N_{NCI}[k] = \sum_{i=1}^{m} |N_i[k]|^2 \sim \chi^2(2m)$$

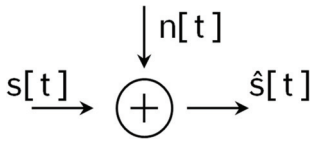

Figure 7.6 Additive white Gaussian noise model.

In contrast to the FFT, the mean value of the noise power scales linearly with the number of channels in the same way the signal power does. Therefore, the signal to noise ratio is not improved. However, the variance is decreasing which has an effect on the possibility of false alarm. An example measurement is depicted in Figure 7.7, comparing the noise distribution of one channel and the distribution after the integration of 32 channels. It can be observed that for the same threshold level, a lower probability of false alarm can be achieved due to the lower variance of the blue histogram. The other way round, for the same probability of false alarm, a lower threshold level can be used, which increases the detection rate.

7.2.4 Angle Estimation

In Subsection 7.2.1 the measurement principle of antenna arrays has been introduced briefly. In general, the angle estimation is based on the measured phase offset ϕ_n between different antenna positions (cf. Figure 7.8).

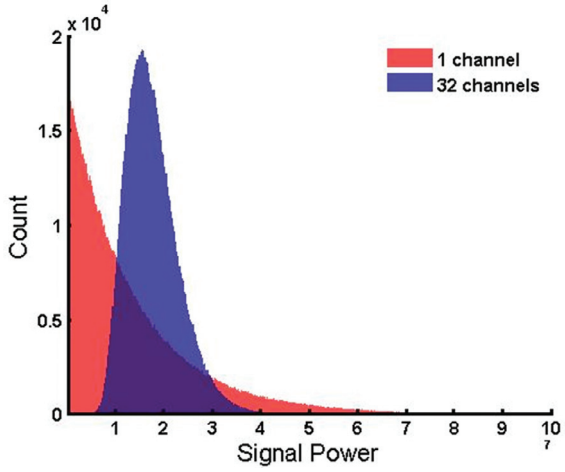

Figure 7.7 Histogram of a noise measurement showing the chi-squared distribution before and after NCI.

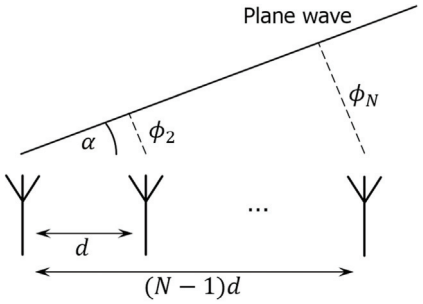

Figure 7.8 Uniform linear antenna array with spacing d and resulting steering vector $\boldsymbol{v}(\alpha)$.

Since the antenna positions are known, a conclusion may be drawn on the direction of arrival. For this purpose, the introduction of a steering vector $\boldsymbol{v}(\alpha)$ can be useful. This vector contains the expected phase offsets, equivalent to an ideal incident signal from a certain angle α:

$$\boldsymbol{v}(\alpha) = [e^{j\phi_1(\alpha)} \quad e^{j\phi_2(\alpha)} \quad e^{j\phi_3(\alpha)} \quad \dots \quad e^{j\phi_N(\alpha)}]$$

In the case of a linear array with N elements, the steering vector is simply constructed from the distance d between two antenna elements, the wavelength λ and the incident angle α. The phase of the first element is normalized to zero and the amplitudes are assumed to be all equal one:

$$\boldsymbol{v}(\alpha) = [1 \quad e^{j2\pi \cdot d \, \sin\alpha/\lambda} \quad e^{j2\pi \cdot 2d \, \sin\alpha/\lambda} \quad \dots \quad e^{j2\pi \cdot (N-1)d \, \sin\alpha/\lambda}]$$

Similar to the spectral estimation, different classes of algorithms can be identified. Some procedures like the Bartlett beamformer just calculate a weighted sum of the received signal vector $\boldsymbol{x}$. This is done for each possible DOA and results in an angular spectrum:

$$P(\alpha) = |\boldsymbol{x}^T \boldsymbol{v}(\alpha)|^2$$

The magnitude P represents the correlation between the received signal and the steering vector. A subsequent maximum search extracts the estimated target angle. The separation of two targets is also possible by simply extracting the two largest peaks, however attention has to be paid to the occurrence of sidelobes. Furthermore, the width of the mainlobe determines the separability which is often not satisfactory.

More sophisticated methods to mention are the Capon beamformer, also known as minimum variance estimator, which achieves a better angular separability. Another important class is known as subspace based methods, incorporating MUSIC and ESPRIT as the most prominent examples. Finally, maximum-likelihood estimators exist, which need to know the model order in advance, i.e. the number of targets. However, if the targets have already been separated by different ranges and velocities, the estimation of the model order is feasible because only few targets will be present, in most of the cases only one. A comprehensive overview of existing methods and algorithms is given in [2].

7.3 Hardware Accelerators for MIMO Radar Systems

7.3.1 Basic Structure of a Streaming Hardware Accelerator

Figure 7.9 shows the overview of a hardware-accelerator for high-resolution MIMO radar sensors. Obviously, a high degree of parallelism can be observed, due to the pair wise independence of the receiving channels. Up to the NCI step, each data stream is processed for its own.

The spectral analysis is carried out with the help of a FFT, whose efficient implementation in streaming applications is well understood. A critical step in the design process of this block is the specification of the maximum FFT lengths, as this parameter determines essentially resource usage. Furthermore, when using fixed-point arithmetic, the word length and data scaling behavior can have major effects on performance and efficiency. This aspect is investigated in Subsection 7.3.2.

Regarding the two-dimensional FFT, a concept for data storage and transfer has to be developed. The storage of a complete chirp sequence, i.e. a set of K ramps is required in order to perform the second dimension FFT processing. This dictates mainly the size of the memory, which grows rapidly due to the influence of further key parameters. In general, increasing the resolution in range, in velocity or in the angular domain, also increases the required memory size. It turns out, that this size exceeds rapidly several

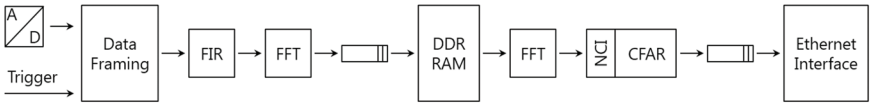

Figure 7.9 Architecture of a streaming hardware accelerator.

MBytes. Thus, the usage of large DRAMs becomes necessary since the size of an on-chip SRAM cache memory is not sufficient anymore. An analysis for different modulation and system parameters can be found in [12].

Regarding the throughput of the memory, the addressing scheme affects heavily the performance in the case of a DRAM. The row opening and closing delays, as well as the read and write transfers can be completely hidden due to the streaming nature of the application. The problem of transforming large two-dimensional matrices with the help of DRAMs has been investigated in [13]. An addressing scheme suitable for the application to chirp-sequence processing has been derived in [14].

Depending on the type of threshold estimation, the calculation can be a simple mean value in the case of CA-CFAR, but it can also become very costly in the case of a sorted list (OS-CFAR). Subsection 7.3.3 presents an efficient architecture based on the rank-only OS-CFAR which avoids a complete sorting of the values inside the window.

7.3.2 Pipelined FFT Accelerator

For streaming applications, pipelined FFT architectures provide a very high throughput. The usage of dedicated hardware accelerators is especially useful for real-time applications, where a high degree of capacity utilization can be achieved. Many different implementation forms have been reported in the past decades. One important parameter is the used butterfly architecture, which can be based on a Radix-2, Radix-4 or Split-Radix decomposition, just to mention a few. In practice, multiple butterflies are cascaded to achieve longer transform lengths. Another important design decision is the use of a single-path vs. multi-path implementation.

A straight forward implementation of the Cooley and Tukey FFT algorithm is shown in Figure 7.10 [15]. It is realized with Radix-2 butterflies which are combined in a decimate-in-frequency (DIF) decomposition. This architecture can process one sample per clock cycle and needs $\log N - 1$ multipliers.

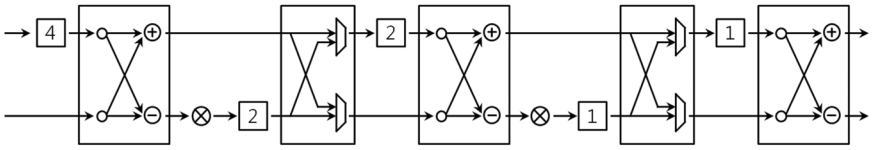

Figure 7.10 Radix-2 FFT implementation based on a multi-path delay commutator (MDC) pipeline.

Furthermore, several buffer memories are required which have the total size $3N/2$.

When analyzing the data flow, it turns out that the butterflies and the multipliers are only used half of the time. Furthermore, only half of the memories store valid data at the same time. Several optimizations have been proposed in order to increase the utilization of the multipliers and memories. 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 (cf. Figure 7.11) [16].

When using Radix-4 butterflies, the number of multipliers can be reduced as well, at the cost of more complicated butterflies requiring more dedicated adders.

Another FFT algorithm for pipelined implementations has been proposed by He and Torkelson [17] 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, all multiplications after the first 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. Table 7.1 compares different implementations.

In the case of multiple parallel data streams, the utilization of the complex adders and multipliers can be further increased to 100% by using a modified MDC architecture with a proper scheduling of the different data streams [18].

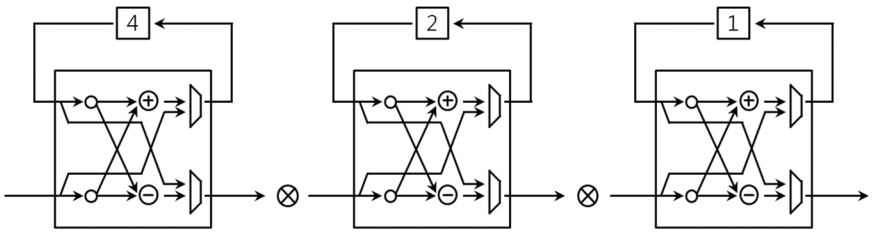

Figure 7.11 Radix-2 FFT implementation based on a SDF pipeline.

Table 7.1 Resource usage of different pipelined FFT implementations [17]

	No. of Multipliers	No. of Adders	Memory Size
Radix-2 MDC	$2(\log_4 N - 1)$	$4 \log_4 N$	$3N/2 - 1$
Radix-2 SDF	$2(\log_4 N - 1)$	$4 \log_4 N$	$N - 1$
Radix-4 SDF	$\log_4 N - 1$	$8 \log_4 N$	$N - 1$
Radix-2^2 SDF	$\log_4 N - 1$	$4 \log_4 N$	$N - 1$

In the case of MIMO systems this approach outperforms the Radix-2^2 SDF implementations which seem to be superior in single channel applications.

Even though not optimal in terms of butterfly utilization, a Radix-2 based architecture provided by the Xilinx IP Core is used for the presented MIMO radar system [19]. The principal reason is the faster implementation and integration time. The efficiency in terms of resource usage can be improved in future work.

Fixed-Point Noise

In digital signal processing systems, all computations are carried out with discrete values. The majority of arithmetic units use fixed word lengths which always have a limited accuracy. Consequently some amount of quantization noise is added for each rounding operation. Often floating-point values are used, because they work very well in most environments, regardless of the input signal characteristics. However, if the dynamic range of the input signal is known to a certain extent, fixed-point arithmetic can considerably reduce the resource usage. Many FFT accelerators use integer operations and various models for the engendered quantization noise have been developed.

The quantization noise due to truncation or rounding after a multiplication is often modeled as additive white noise source with a uniform distribution. Even though not accurate under all circumstances, this model is appropriate if the input signal has a sufficiently large bandwidth and amplitude [20]. It can thus be applied to a radar system, due to the wide bandwidth background noise, which is always visible.

The quantization noise variance σ^2 in the case of a uniform distribution can be derived for a simple truncation [21]. The least significant bit (LSB) after the truncation is denoted by $q = 2^{-b}$, where b is the resulting integer word length and k the number of truncated bits:

$$\sigma^2 = \frac{q^2}{12}(1 - 2^{-2k})$$

During the computation of the FFT, the variables grow with each butterfly stage, resulting from the addition inside the butterflies. The complex multiplication does not scale up the intermediate values, because they perform just a rotation in the complex plane and the twiddle factors are all normalized. Thus, the resulting word length of the FFT depends on the input data and grows by 1 bit with each stage. In order to maintain a certain word length, the values can be scaled after each stage at the cost of additional quantization noise.

A complete scaling of the input signal is disadvantageous and engenders an even higher level of quantization noise [22].

Furthermore, a quantization error is introduced after the multiplication, because the resulting word length is cut down by half and also the twiddle factors are represented with limited accuracy. However, it turns out that the coefficient errors are less severe than the round-off errors if the same word length is used for both the coefficients and signals [22].

The following analysis is based on [22], and only the most severe round-off errors are considered. The used noise model applies to a Radix-2 decimation-in-frequency butterfly, which is used by the presented system. Furthermore, the signals are not scaled directly after the addition, but only after the multiplication. Therefore, only one noise source is present for each butterfly output. For the sake of simplicity, the error variance for both outputs is considered equal, even though only one output is the result of a multiplication. This approximation acts as an upper bound because the real output variance after the addition and the truncation will be slightly lower.

The variance of the quantization error σ_e^2 after the multiplication is derived by decomposing the complex operation into four real multiplications, each truncated individually. In this case, the uniform noise model is applied and the number of truncated bits k is assumed to be sufficiently large:

$$\sigma_e^2 = 4\frac{q^2}{12} = \frac{q^2}{3}$$

The total output variance is then calculated by adding all error variances contributing to the respective output. When observing the butterfly graph, a tree-like structure leads to each output, incorporating $N-1$ butterfly nodes. However, if the signal is scaled after each stage, the accumulated noise decreases just as well. In this case, the total noise variance σ_N^2 equals to:

$$\sigma_N^2 = \sigma_e^2 + 2\frac{\sigma_e^2}{4} + 4\frac{\sigma_e^2}{16} + \cdots + \frac{N}{2}\frac{\sigma_e^2}{(N/2)^2} =$$

$$\sigma_N^2 = \left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{N/2}\right)\sigma_e^2 \approx 2\sigma_e^2$$

Remarkably, the total noise variance is independent of the length of the FFT. However, when examining the signal-to-noise ratio (SNR) at the output, it turns out that the SNR is decreasing for longer FFT lengths, because the output is a scaled version of the FFT. Considering a random input signal,

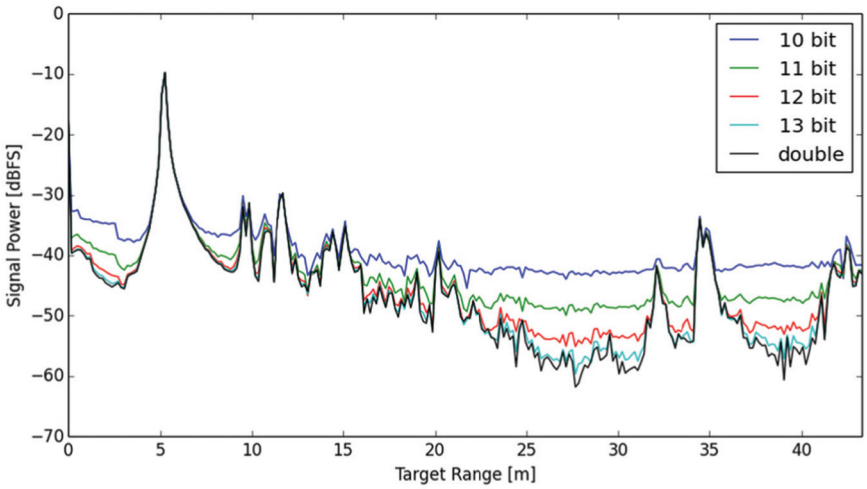

Figure 7.12 Effects of different word lengths on the amount of quantization noise.

with all values i.i.d. and a variance σ_s^2, then the variance for each output of the FFT is scaled by $\frac{1}{N^2}$:

$$\sigma_{s,fft}^2 = \frac{1}{N^2}(N\sigma_s^2) = \frac{\sigma_s^2}{N}$$

Composing the signal-to-noise ratio at the output leads to the expected result:

$$\text{SNR} = \frac{\sigma_{s,\,fft}^2}{\sigma_N^2} = \frac{\sigma_s^2}{2\sigma_e^2 N} = \frac{3\sigma_s^2}{2Nq^2}$$

Consequently, if the FFT length N is doubled, the word length has to be increased by half a bit also in order to maintain a constant signal-to-quantization-noise ratio (SQNR). To illustrate the influence of the word length, an exemplary radar measurement, processed with a scaled fixed-point FFT is shown in Figure 7.12.

Different word lengths have been used in order to illustrate the effect of the introduced quantization noise. The FFT is implemented in a Radix-2 DIF decomposition. The values are rounded and scaled after each stage. The black curve has been processed with double precision floating-point and acts as reference.

It can be observed that the fixed point versions lie all above the reference. The reason is that the quantization noise power is added to the signal

and the amount of quantization noise should be the lowest for the floating point version. Furthermore, it can be observed that the noise floor increases significantly in regions with low signal power. The difference for 1bit word length is about 6dB, which correlates with the derived noise model in the case of a truncation or rounding operation. In regions with more signal power, for instance around 5m target range, the quantization noise effect is less severe, due to the higher SQNR.

For a 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. Noise components arising from fixed-point computations should be clearly below the system noise floor in any case. It is important to consider the processing gain when designing an optimal word length, because the noise level drops for larger FFT lengths. Thus, the maximum possible FFT length can be considered as worst-case scenario when designing the word length of the FFT.

7.3.3 Rank-Only OS-CFAR Accelerator

The CFAR processing step requires the use of a local window for threshold calculation. For a streaming application, a sliding window exploits the locality of the data and can be used easily without excessive memory transfers. It is implemented with the help of a shift register. Current FPGA devices offer several different building blocks for this purpose, 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 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.

As described in subsection 7.2.3, the rank-only detection step depends on N comparisons, a binary sum and a comparison for the decision. Each register of the sliding window is routed to a dedicated comparator, whose second input is fed by the CUT with a threshold value applied. The comparison result is routed to a binary adder with N inputs. Several LUTs are cascaded for this step, which can impose an upper limit to the clock frequency. In order to maximize performance, it is implemented in two steps, i.e. the lower and the upper half of the window is summed up separately before the final rank is computed.

The described architecture has been implemented on a Virtex-7 FPGA and the engendered resource usage has been analyzed. For window sizes up to 128, an operating frequency of 250 MHz could be achieved by this

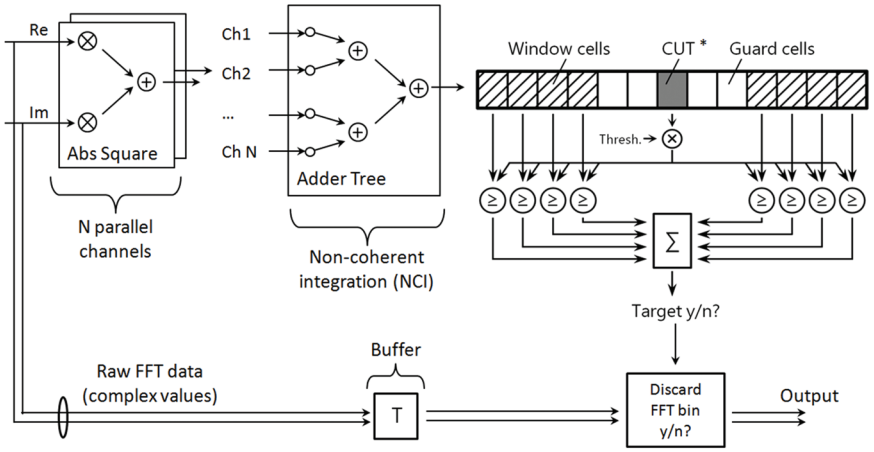

Figure 7.13 Architecture of the rank-only OS-CFAR accelerator.

implementation. The LUT usage depending on the number of channels is depicted in Figure 7.14.

As expected the CFAR-processing part (greenish blue color in Figure 7.14) is practically independent from the number of channels, because the NCI step is performed in advance. The NCI step by itself scales approximately with $\log N$, which is a result of the used tree structure. For a number of channels above 32, the raw data buffer which compensates the pipeline delay consumes more LUTs than the CFAR processing part. It grows linearly with the number of channels and is thus the dominating part for large channel numbers. The

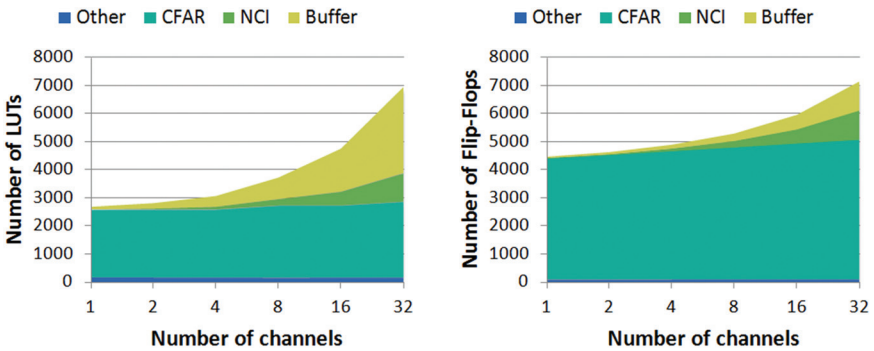

Figure 7.14 Resource usage against number of channels for a constant window size (128 cells).

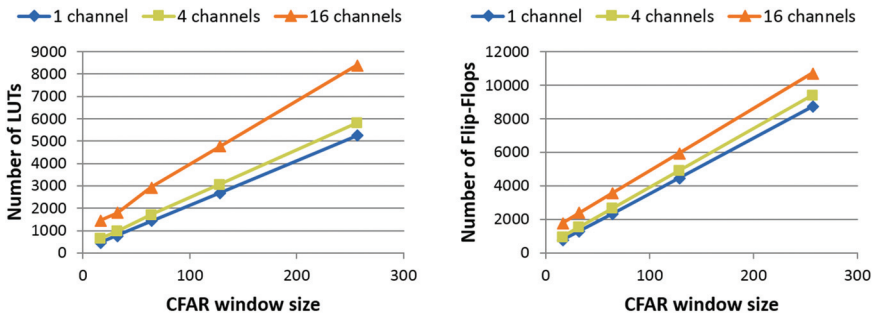

Figure 7.15 Resource usage against window size for different number of channels.

usage of a dedicated Block RAM can be considered if the number of LUTs is scarce.

The scaling behavior in relation to the window size turns out to be nearly linear (cf. Figure 7.15). It is clearly dominated by the N comparators as well as the data buffer equalizing the pipeline delay. The number of channels has a much lower effect on LUT resource usage as the window size. For instance, the resource usage is within the same order of magnitude when comparing one and 32 channels. The architecture can be considered as very efficient for large channel numbers and is thus suitable for MIMO systems. It can be concluded that the usage of NCI before the actual CFAR processing is beneficial in two ways. It improves detection performance and reduces resource requirements at the same time.

7.4 Conclusion

A data processing architecture for future automotive MIMO radar systems has been presented in this chapter. Beside the algorithmic background information, a focus has been set on the target detection with the help of CFAR processing. Attention has been paid to real-time requirements as well as resource usage. The step between the target detection and the subsequent angular processing could be identified as a good data interface between different processing units, each optimized for different requirements on control flow complexity and data throughput.

Furthermore, a FPGA based implementation of the raw data preprocessing chain has been presented and investigated. As crucial points in the design procedure, several parameters could be identified. Especially, the maximum length of the FFTs and the expected dynamic range of the signals determine

basically the resource usage in terms of logic elements and memory size. These parameters have a strong dependency on the used modulation waveform, which is why the design of the signal processing architecture has to be integrated into the overall radar system design process. With the help of model-based design space exploration methods, the estimation of resource requirements is feasible, even in an early development stage. The derivation of appropriate models from the realized hardware implementation will be part of future work.

The used design methodology which evolved from the DESERVE project turned out to be very efficient in terms of performance and development time. The usage of heterogeneous platforms, even in an early prototype system, made it possible to handle the tremendous amount of data in real-time. Thanks to the integration with established tools like ADTF and Matlab, the system is ready to be integrated into a test vehicle with a multiplicity of sensors devices. Finally, the early availability of such high resolution automotive radar sensors can be an important step on the way towards automated driving.

References

[1] V. Winkler. "Range Doppler detection for automotive FMCW radars." *IEEE 37th European Microwave Conference (EuMC),* Munich, Germany, 2007.

[2] H. L. van Trees. "Optimum Array Processing (Part IV of Detection, Estimation, and Modulation Theory)" *John Wiley & Sons,* 2004.

[3] U. Nickel. "Angular superresolution with phased array radar: a review of algorithms and operational constraints." *IEE Proceedings F: Communications, Radar and Signal Processing* 134.1 (1987): 53–59.

[4] D. Kellner, M. Barjenbruch, J. Klappstein, J. Dickmann and K. Dietmayer. "Wheel extraction based on micro doppler distribution using high-resolution radar." *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM),* Heidelberg, Germany, 2015.

[5] M. Bouchard, D. Gingras, Y. De Villers and D. Potvin. "High resolution spectrum estimation of FMCW radar signals." *IEEE 7th SP Workshop on Statistical Signal and Array Processing,* Québec, Canada, 1994.

[6] M. A. Abou-Khousa, D. L. Simms, S. Kharkovsky and R. Zoughi. "High-resolution short-range wideband FMCW radar measurements based on MUSIC algorithm." *IEEE Instrumentation and Measurement Technology Conference (I2MTC),* Singapore, 2009.

[7] J. B. Billingsley et al. "Statistical analyses of measured radar ground clutter data." *IEEE Transactions on Aerospace and Electronic Systems* 35.2 (1999): 579–593.

[8] B. Magaz and M. L. Bencheikh. "An efficient FPGA implementation of the OS-CFAR processor." *IEEE International Radar Symposium (IRS),* Wroclaw, Poland, 2008.

[9] 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* 20.6 (2010): 1733–1747.

[10] M. R. Bales, T. Benson, R. Dickerson, D. Campbell, R. Hersey and E. Culpepper. "Real-time implementations of ordered-statistic CFAR." *IEEE Radar Conference (RadarCon),* Atlanta, USA, 2012.

[11] M. A. Richards. "The discrete-time Fourier transform and discrete Fourier transform of windowed stationary white noise." *Georgia Institute of Technology, Tech. Rep,* 2007.

[12] 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." *IEEE International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS),* Samos, Greece, 2014.

[13] S. Langemeyer, P. Pirsch and H. Blume. "Using SDRAMs for two-dimensional accesses of long $2^n \times 2^m$-point FFTs and transposing." *IEEE International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS),* Samos, Greece, 2011.

[14] F. Meinl, E. Schubert, M. Kunert and H. Blume. "Realtime FPGA-based processing unit for a high-resolution automotive MIMO radar platform." *IEEE 12th European Radar Conference (EuRAD),* Paris, France, 2015.

[15] L. R. Rabiner and B. Gold. "Theory and application of digital signal processing." *Prentice-Hall, Inc.,* 1975.

[16] E. H. Wold and A. M. Despain. "Pipeline and parallel-pipeline FFT processors for VLSI implementations." *IEEE Transactions on Computers* 100.5 (1984): 414–426.

[17] S. He and M. Torkelson. "A new approach to pipeline FFT processor." *IEEE 10th International Parallel Processing Symposium (IPPS),* Honolulu, USA, 1996.

[18] 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* 21.4 (2013): 720–731.

[19] Xilinx Inc. "LogiCORE IP FFT." *PG109 v9.0,* October 2014.

[20] C. W. Barnes, B. N. Tran and S. H. Leung. "On the statistics of fixed-point roundoff error." *IEEE Transactions on Acoustics, Speech and Signal Processing* 33.3 (1985): 595–606.

[21] D. Menard, D. Novo, R. Rocher, F. Catthoor and O. Sentieys. "Quantization mode opportunities in fixed-point system design." *IEEE 18th European Signal Processing Conference (EUSIPCO),* Aalborg, Denmark, 2010.

[22] C. J. Weinstein. "Quantization effects in digital filters." *Lincoln Laboratory, Massachusetts Institute of Technology, Tech. Rep. No. TR-468,* 1969.