SYNOPSYS®

WHITE PAPER

Digital Signal Processing for Frequency-Modulated Continuous Wave RADARs

Author

Dmitry Utyansky
Senior Staff Engineer,
Synopsys

Introduction

RADAR is an established technology, but interest has been stimulated recently by demands of driver assistance systems and emerging self-driving cars for applications including proximity warning, blind spot detection, adaptive cruise control, and emergency braking.

Advanced driver-assistance systems (ADAS) and autonomous driving (AD) systems typically combine several types of sensors, such as cameras, RADARs, and LiDARs. Different types of sensors have their strengths, and effectively complement each other. Cameras and the appropriate machine vision algorithms can "see" lane marking and recognize traffic signals and signs. LiDAR can offer high (cm-level) resolution and high density of collected data points. RADAR technology is indispensable in ADAS applications because of its robustness to a variety of environmental conditions, like rain, fog, snow, and its ability to directly and precisely measure range and velocity. Basic use cases can rely on RADAR sensors only, enabling cost-effective solutions. RADAR can be used to augment, cross-check or 'fuse' situational models derived using advanced computer vision algorithms.

Automotive applications add specific requirements, such as:

- Safety and security requirements.
- Life-cycle management: cars can have a 20+ year life cycle, but the domain evolves, both in terms of regulations and algorithms. Therefore, a system implemented today must allow the ability to upgrade and, ideally, some reserves for future growth of computational complexity, especially in higher-end applications.
- Electrical power available to the vehicle is usually limited: this affects the design of the car, costs, and mileage. Lower power consumption also means lower system temperatures and decreased probability of failures.
- As the number of RADAR and other sensors in a car grows, system costs become important. Higher integration and efficient system-on-chip (SoC) designs enable cost reduction, hence the growing need for highly integrated silicon.

To meet these requirements car manufacturers are looking for a more programmable solution that will increase flexibility and enable a future upgrade path.

RADAR implementation complexity and the respective algorithm computational requirements vary depending on application and system requirements. This paper provides an overview of digital signal processing algorithms typically used in frequency-modulated continuous wave (FMCW) RADARs, reviews system tradeoffs, offers a way to approximately estimate computational complexity with a few numerical examples, and finally presents implementation options and solutions available from Synopsys.

RADAR Signal Processing

In a typical ADAS application, RADAR is used to detect and track objects around the car. Typically, several systems cover various sectors of the environment. RADAR estimates distance to a target, velocity of a target, or both distance and velocity. Depending on the system configuration, RADAR can also determine the direction to the target. The underlying physics is straightforward: a RADAR unit transmits a radio signal, the signal is reflected off the target, and the reflected signal is received and analyzed. One approach to estimate the range to the target (R) is to measure the signal round-trip time, which directly translates to the distance to the target:

$$R = \frac{\Delta t\, c}{2} \qquad (1)$$

where Δt is the roundtrip time and c is the speed of light.

For RADAR to be able to measure the round-trip time, the signal transmitted must have some form of time-varying modulation. The two most often used RADAR types are pulse modulation and frequency modulation of a continuous signal. FMCW systems are more widely used in automotive applications. FMCW systems perform well for precise measurement of short time intervals, and experience few or no issues with receive and transmit (RX and TX) interference. On the other hand, unambiguous range measured by FMCW systems is inherently limited due to the signal structure. Range typically varies from 200-300 m for "long range" FMCW RADARs to within a few dozen meters for "short range" FMCW RADARs. See DaimlerChrysler AG review[10] for a few examples of existing systems.

Basic Math and Design Tradeoffs

FMCW RADARs commonly use some form of linear frequency modulation, e.g., sawtooth or triangular. Reviewing a simple example with the associated math helps understand design tradeoffs.

Figure 1 depicts a typical FMCW signal, where the transmitted signal is organized into packets, and each packet consists of a series of linearly "sawtooth"-like frequency modulated fragments ("*chirps*"). Each chirp signal swipes the available radio frequency bandwidth *B* linearly. It is assumed that target range and velocity do not change (or change little) for the duration of one packet transmission. This structure helps in resolving range/velocity ambiguity. The time interval of packet transmission is often referred to as the *Coherent Processing Interval* (CPI).

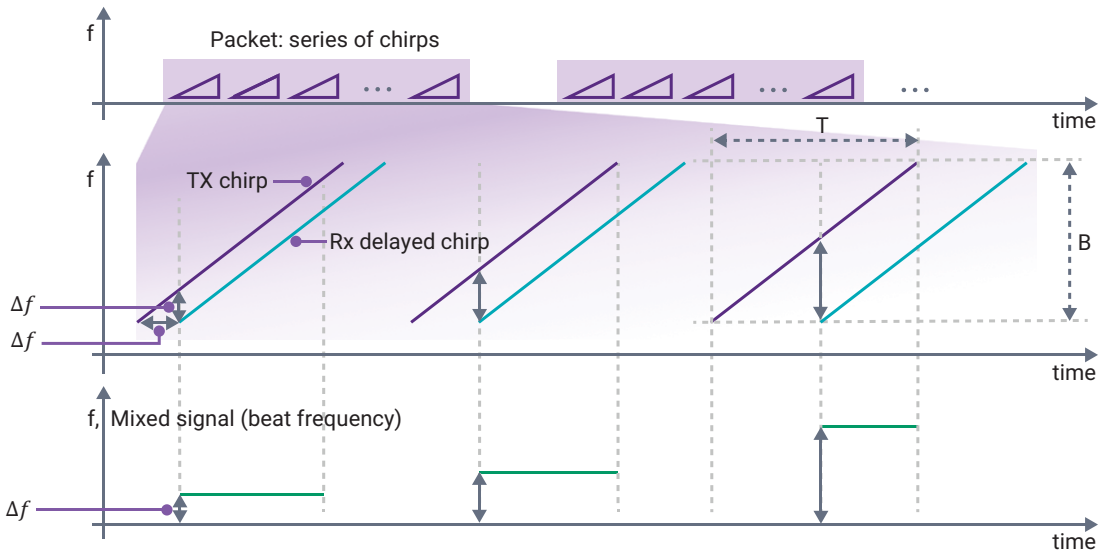

Figure 1: Sawtooth FMCW signal

The chirp is transmitted, gets reflected off the target, and the reflected signal is received by the RADAR. This occurs while the chirp is still being transmitted, delayed by the round-trip time. FMCW RADAR does not measure this round-trip time directly, instead the delayed received signal is mixed (multiplied) with the signal being transmitted. The two signals have a small difference in frequency. The exact math differs slightly for real and complex signals, but in either case from the mixed signal one can easily obtain a signal in which the frequency is equal to the difference of the transmitted and received signals, Δf. This is also known as *beat frequency*.

Note that the interference from the transmitted signal is easily filtered out with low-pass filtering. The frequency of this difference signal, Δf, is proportional to the round-trip time, Δt:

$$\frac{\Delta f}{\Delta t} = \frac{B}{T} \qquad (2)$$

where T is transmission time and B is chirp bandwidth. And so, given (1), the range is ultimately proportional to Δf

$$R = \frac{cT\Delta f}{2B} \qquad (3)$$

One observation from (3) is that since the resolution of frequency identification is inversely proportional to the time of observation, T, the range resolution is inversely proportional to the bandwidth used (Table 1):

$$\Delta R = \frac{c}{2\,B} \qquad (4)$$

Bandwidth, MHz	Range resolution, m
10 MHz	15
250 MHz	0.6
500 MHz	0.3
4 GHz	0.04

Table 1: Range resolution vs. bandwidth

High-resolution requirements of automotive applications have been one of the driving forces of migration to 76-81 GHz bands from the legacy 24 GHz band. The accuracy also depends on linearity of RX and TX channels and signal-to-noise ratio (SNR). The following are a few examples of tradeoffs involved in waveform design:

- Wider bandwidth and faster frequency ramp improve range resolution, increasing Δf
- Maximum unambiguous range is limited by the required overlap of the TX and RX signals: T should be large enough
- T cannot grow too much; if the range substantially changes during the chirp, so does the "beat frequency", introducing ambiguity. There is also ambiguity between range and velocity detection.
- Processing requirements depend primarily on the data rate of the sampled beat signal and the overall system time resolution requirements—how many estimations per second must be made (also referred to as *frame rate*).

Current systems typically utilize a sweep bandwidth of a few hundred MHz but can go above 1 GHz for ultra-short range. The typical beat frequency is around 10-20 MHz. Figure 2 shows a simplified diagram of a typical one-channel FMCW radar.

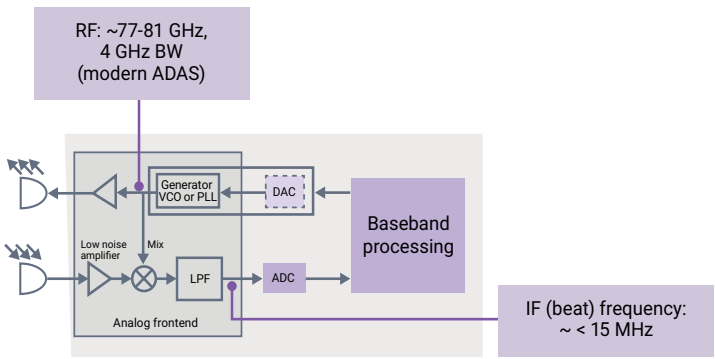

Figure 2: Basic system diagram for 1 RX, 1 TX channel

The system includes an analog frontend, with a waveform generator for generating chirp signals, controlled by a processor, transmit and receive amplifiers, mixer, low-pass filter, and analog-to-digital converter. There are multiple ways to implement a waveform generator—with a frequency synthesizer using programmable PLL, typically fractional PLL, or with explicit digital-to-analog converter. The digital processing section can be implemented on one or more commercial DSPs or application-specific instruction set processors (ASIPs).

Signal Processing Flow and Computational Requirements

A nice feature of RADARs and, in particular FMCW RADAR, is that both the range and velocity of the target can be obtained directly. Figure 3 shows the typical signal processing for each packet. The first step is identification of tones in the "beats" signal, which then correspond to the ranges of the targets. The straightforward approach is to use Fast Fourier Transform (FFT) algorithms. The tones identified correspond to target candidates. The next step is to find velocities, assuming they are needed. The distance to the target changes slightly from chirp to chirp within the packet, and this distance change results in change of phases in the tones identified in the "range" step. With known phase changes, the velocities of the targets can be computed. This can be considered as the second set of tone identification problems, slicing across the frequency lines of the results of the first layer of FFTs. This second stage of FFTs is often referred to as *velocity or Doppler* FFTs.

After these two stages of FFTs, there is a 2D map of velocity/range points, with higher amplitude values corresponding to target candidates. Then further processing should be applied to identify real targets. Usually this involves some version of thresholding operations, like Constant False Alarm Rate (*CFAR*) schemes. CFAR algorithms consider noise variance in the neighborhood of each value being tested, and adjust thresholds accordingly. The approach was originally proposed in the 1960s by Finn and Johnson [11], but now there is a fairly broad family of algorithms, with different variations of threshold computations, using averages, min/max, ordered statistics, etc. Reference [12] for a review.

Subsequently clustering and tracking/filtering algorithms can be applied, depending on how the results are to be used.

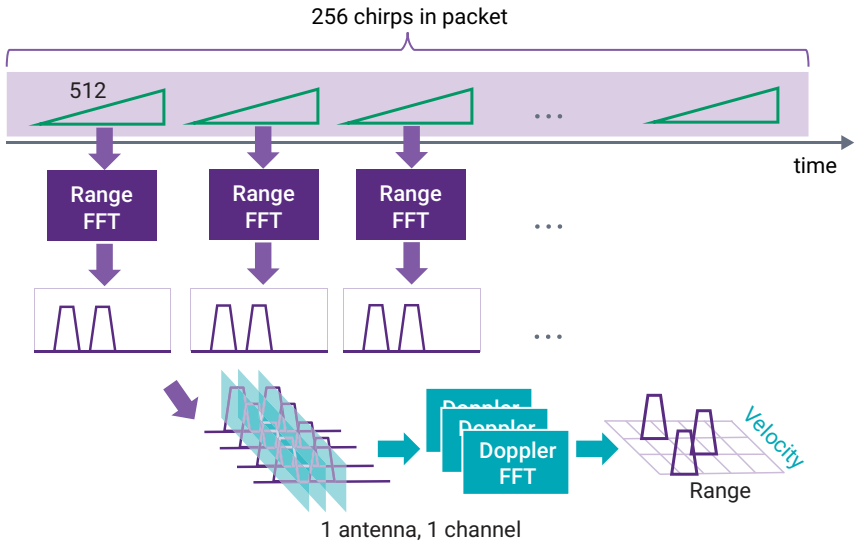

Figure 3: Range/velocity processing of one channel of FMCW RADAR

The range and velocity "2D FFT" usually makes up the bulk of signal processing computations. Subsequent processing steps are done at lower data rates. Computational complexity for most FFT algorithms is $O(N\log(N))$, where N is the size of the data block and the base of the logarithm depends on the specific algorithm (two for radix-2 factorization, four or higher for higher radices). The linear coefficient depends on fine details and hardware used for the implementation. Conventional DSP processors usually have instructions and architecture features that accelerate the inner kernel of the algorithms, so that FFT-typical "butterfly" operations are done very efficiently. It is not uncommon for one radix-2 butterfly to execute at one cycle per point rate, which makes a 256-point radix-2 FFT implementation run in about 256*log2(256)=2048 cycles plus overheads (eight passes of 256 cycles each). Specialized ASIP accelerators consuming and producing one sample per clock can be implemented, so that pipelined execution of a 256-point FFT takes 256 cycles.

To get some feel for the numbers, consider the example of chirps and packets in Figure 3, and suppose that the beat signal is limited to below 16 MHz. That means that it has to be sampled using a sampling rate that is at least twice as high (Nyquist frequency). Usually the signal is oversampled at even higher rates to simplify low-pass filters in front of the ADC. This paper does not go into

details of analog frontend design, but there are tradeoffs there as well, and often the way to decrease analog part complexity and power dissipation is to move more processing to the digital part, essentially doing more filtering as DSP. This has to be factored into computational requirements. Another factor affecting design of the analog frontend and processing of the sampled signal is whether the mixing functionality is implemented using real or complex — I/Q quadrature, also known as the Zero-IF approach. Complex sampling allows a sampling rate that is two times lower, but at the expense of complex symbols, each containing twice as much information.

Suppose for simplicity that after this pre-filtering the signal is down-sampled to the critical rate of 2*16 MSPS, or 16 MSPS with complex samples. That yields 512 complex samples per chirp, 256 chirps per packet, and packet duration of about 10 ms at this rate. Then the processing will include:

- 256 “Range” FFTs, 512 points each
- 512 “Doppler” FFTs, 256 points each
- -> 256*512*2 = 262 k points processed per 10 ms, or 26 Mpoints per second

This processing load can be handled by a 1-cycle-per-point accelerator running at 26 MHz, or by a scalar DSP running at about an order of magnitude higher clock rate.

This example considers only single-channel processing. To estimate direction to the target, multiple antennas and multiple channels are required. This multiplies the compute requirements by the number of channels (e.g., eight times, see the 'Angle of Arrival Estimation' section below), thus quickly scaling up processing requirements beyond the capabilities of a typical scalar DSP processor.

Accuracy Requirements and Fixed Point

Required accuracy and signal dynamic range have to be taken into account when selecting data types for storing and processing signal. The usual way to increase computational efficiency and reduce energy consumption is to use fixed point arithmetic for the bulk of signal processing computations. Usually 16- or 32-bit data types, also referred to as Q15 and Q31, are used for that (see Reference [5] for a more complete programming model discussion). Most DSP processors offer instruction sets optimized for these data types, supporting efficient scaling shifts and saturation. Usually 16-bit operations are 2x more efficient cycle-wise than 32-bit. See Reference [1] for ISA details and References [3, 4] for DSP Library functions benchmark data.

Depending on dynamic range and accuracy requirements, a 16- or 32-bit fixed-point implementation can be selected.

Usually high-rate ADC used in RADAR systems have limited resolution (below 16 bits), therefore it is not required to use more than 16 bits for incoming samples. Subsequent processing might require wider data types though. For larger FFTs some form of scaling will be required to compensate for potential data range growth. FFT of size N can result in N times larger values on output, although this data growth depends on the input signal characteristics. If this does not fit into a selected data type, output values have to be downscaled N times; for a 256-point FFT that effectively means that eight bits must be reserved for potential growth. So out of 16 bits of Q15 fixed point only eight bits remain for meaningful signal, limiting the resulting signal-to-noise ratio (SNR) to about 42dB instead of 90+. In many cases the first layer of FFTs (range) can be implemented in 16-bit fixed point or block floating point, while the second layer needs to run in 32-bit fixed point or floating point.

For any fixed-point design, data range analysis and selection of appropriate scaling is very important. If high values happen relatively rarely—a reasonable approach could be to allow signal to saturate in such cases. This will slightly deteriorate SNR for a few rare cases, but will improve performance for the most common ones.

It is important to have a working simulation for all candidate data formats for the entire processing chain, and representative input data (test vectors), to be able to analyze the impact of choices on system performance before finalizing system architecture. It is crucially important to create a good base of test vectors, covering all data ranges and corner cases, early in the development cycle. They can be subsequently used throughout the development for validation and verification as well as performance tuning.

Angle of Arrival Estimation

To obtain the direction to the target, some form of parameterizable directional sensitivity must be introduced to the system. This can be done with a physically rotated directional antenna or with an array of fixed antennas and additional signal processing to synthesize the desired “direction”. Most compact modern systems, including ADAS applications, use the latter approach.

Figure 4 illustrates a basic example of a uniform linear array with five antennas and a signal arriving from direction θ. We use the far-field assumption, with planar wave front, to simplify the example. In this case the underlying physics and math are straightforward. If

θ were 0, the wave would hit all antennas at the exact same time, with the exact same phase. For a non-0 angle of arrival, there is a difference in the distance the signal has to travel to reach each antenna, d sin(θ), where d is the distance between antennas.

There are different ways to look at this. One is that if we delay the signal from each antenna by the time it takes for the wave to travel this $\mathrm{d}\ \sin\ (\theta)$ distance and add such aligned signals, we will end up with the combined "antenna" with its direction tuned to θ. This view can be useful if we just want to have a directional antenna, looking in the direction of θ, which we can turn "virtually". This is often referred as "delay-and-sum" beamformer in the time domain.

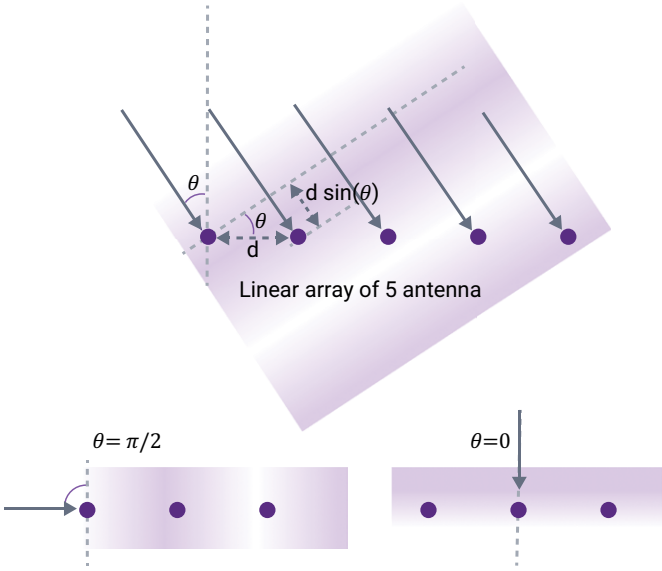

Figure 4: Uniform linear array of antennas

Alternatively, we can look at the chunk of data received from all antennas in this example (or some series of such chunks), and observe that in effect we are sampling the wave in space at different points defined by the antennas' positions. So if the wavelength is λ (and the spatial frequency along the signal travel direction is 1/λ, respectively), at these points the observed sampled discrete spatial frequency of ω radians would be:

$$\omega = \frac{2\pi d \sin\theta}{\lambda} \qquad (5)$$

Once this frequency is identified using any available method, the direction to the signal source will be

$$\theta = \arcsin\frac{\omega\,\lambda}{2\pi d} \qquad (6)$$

Note that since the system is linear, the same reasoning applies to the case of multiple target candidates. Using the same set of data, signals from multiple targets coming from different directions can be identified. The antenna does not need to rotate to scan the entire field of view sequentially.

The direction of arrival (DoA) problem is thus reduced to the problem of identifying the frequencies of tones in the signal again, similar to the case with range and velocity identification, this time "across antennas".

The sampled frequency ω is lowest (0) when the signal comes from the frontal direction, θ=0. This is also where angular resolution of such linear array is at its best. Differentiating ω with respect to the θ, we get

$$\Delta\omega = \frac{2\pi d \cos\theta\ \Delta\theta}{\lambda} \qquad (7)$$

and so

$$\Delta\theta = \frac{\Delta\omega\,\lambda}{2\pi d\cos\theta} \qquad (8)$$

Near the point θ=0, where cosine is close to 1, we get:

$$\Delta\omega = \frac{2\pi d\Delta\theta}{\lambda} \qquad (9)$$

$$\Delta\theta = \frac{\Delta\omega\,\lambda}{2\pi d} \qquad (10)$$

So the wider antennas are placed (the larger d is), the better angular resolution is.

On the other hand, there is a limit to d. If the RADAR must cover the entire ±90 degrees range, then the sampled frequency ω is maximum when the signal comes from the side, $\theta=\pi/2$, and the critical sampling is attained when $d_{max} = \frac{\lambda}{2}$.

For $d = d_{max} = \frac{\lambda}{2}$ near $\theta = 0$, equations (9) and (10) become

$$\Delta\omega = \pi\Delta\theta \qquad (11)$$

$$\Delta\theta = \frac{\Delta\omega}{\pi} \qquad (12)$$

Using N-point FFT for frequency estimation, the resolution $\Delta\omega$ is $\frac{2\pi}{N}$; substituting this into (12) we get the best discernable $\Delta\theta$ to be around 2/N radians. For example, for eight antennas we get only about 15 degrees resolution near the field of view center. If the field of view can be narrowed down, antennas can be spaced farther apart with the respective improvement in the resolution. This is one motivation behind splitting RADAR subsystems into short- and long-range, with “long” range systems' field of view limited to a small angle, e.g. 10-20 degrees total.

With range, velocity, and direction coming together, we get to the often-used notion of a *data cube*, where the complete data set looks like a cube with three dimensions:

- Range
- Velocity
- Direction of arrival

Computing data points for this cube requires significant signal processing, comprised of FFTs or more complex estimation algorithms. It is important to carefully analyze system requirements and design all aspects of data flow and required computations. Careful design can save a lot of resources. For instance, computations along some of the “cube” dimensions can be saved by recognizing that the direction of arrival can be estimated not for all points, but rather for a small subset corresponding to the plausible target candidates identified on the range/velocity plane.

Another interesting design tradeoff is between hardware resources, required resolution and computational complexity of algorithms.

Increasing Resolution

As was briefly discussed earlier, FFT-based estimation of angle of arrival has limited resolution, especially with few antennas. Estimation based on a small sized FFT suffers from biasing due to FFT "binning" effects. In the example of eight antennas and 8-point FFT, resolution is inherently limited. There is wide main lobe or high leakage (high side lobes), or both. Interpolation or zero padding with larger FFT can be to avoid binning bias. Windowing before FFT can help tuning tradeoffs between main lobe width and side lobes. But the overall estimation accuracy is still limited.

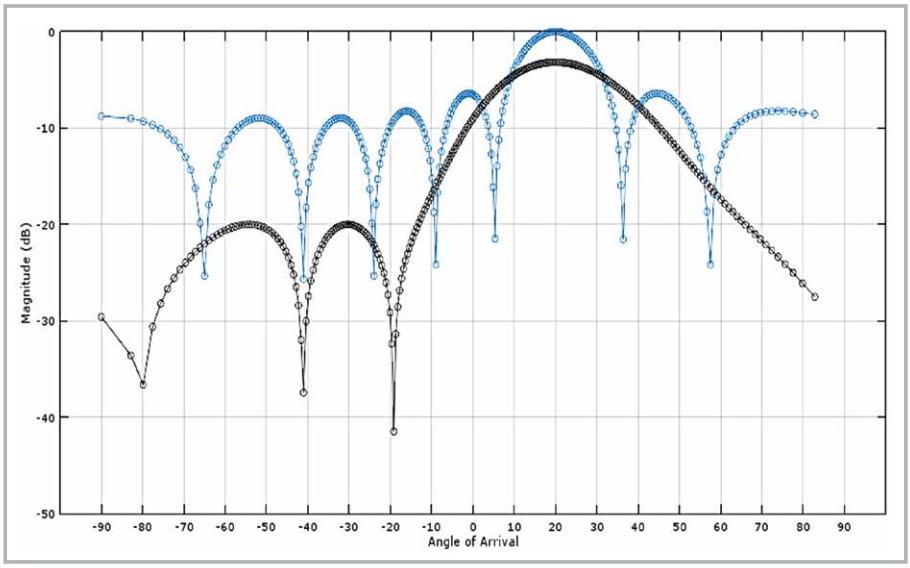

Figure 5: FFT estimation of a single signal with different windowing

Figure 5 shows a typical simulation result for a single signal coming from +20 degree direction, using eight antennas, under ideal noise conditions. Antennas are separated by half-wavelength, enabling +/-90 degrees field of view. Here 256-point FFT is used instead of 8-point one to have more points interpolated. The blue graph uses no windowing, the black graph uses the Hamming window before FFT. This illustrates how leakage to side lobes can be decreased at the expense of increasing the main lobe width (thus decreasing resolution further). Multiple target candidates will ultimately interfere, and close signals cannot be resolved. Figure 6 shows a simulation for the two signals separated by 15 degrees as another example.

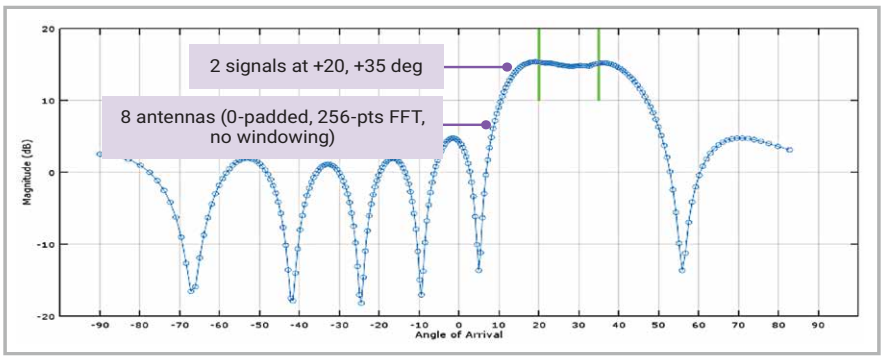

Figure 6: FFT estimation of two signals separated by 15 degrees

The problem can be addressed with larger antenna arrays, increasing system costs. An alternative approach could be to use high-resolution methods like *MUSIC* (Multiple Signals Classification)[8] and *ESPRIT* (Estimation of Signal Parameters via Rotational Invariance Techniques)[9]. Figure 7 shows simulated FFT and MUSIC spectrums for the case of two close signals, using FFT-based estimation with eight antennas (blue graph), FFT with 128 antennas (green graph), and using MUSIC algorithm using data from eight antennas (red graph).

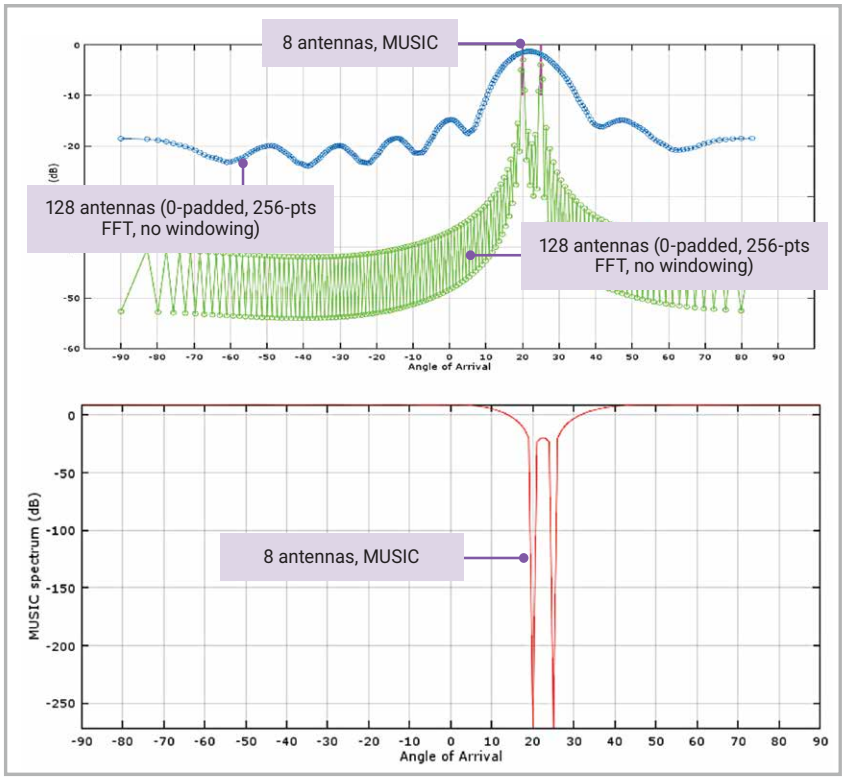

Figure 7: Identification of two close signals, using FFT and MUSIC

The figure shows FFT-based estimation based on eight antennas cannot resolve the two signals, while MUSIC "spectrum" clearly shows them. The reason is that the algorithm can efficiently utilize signal model assumptions, combining several data snapshots to increase accuracy. The underlying mathematics is beyond the scope of this whitepaper, but common features of high-resolution methods, also referred to as "super-resolution methods" (or "subspace methods", of which MUSIC, in particular, is representative) are:

- High resolution, even independent of number of antennas, given enough uncorrelated observations and high enough SNR.
- Usually the number of targets shall be below the number of antennas, or a similar condition exists depending on algorithm details.
- The algorithms have much higher computational costs than the FFT, typically $O(N^3)$ where N is the number of antennas, whereas FFT is $O(N\log(N))$. Usually they involve eigenvalue decompositions of antenna array data covariance matrix or singular value decomposition of the antenna array data.
- There are numerical stability challenges, sensitivity to SNR and correlation of signals (e.g. due to multipath). Floating point implementation is necessary, especially for higher N.

Such algorithms are especially suitable for direction of arrival estimation, with relatively low antenna count and given the ability to intelligently prune the number of computations down to a few reasonable target candidates.

Radar SoC Architectures

System-on-chip (SoC) design for ADAS is a complex task. It involves many application-specific considerations, and requires thorough analysis of performance, implementation constraints, algorithms and data flow patterns. Depending on the application there is a wide range of processing requirements.

One way to estimate the amount of DSP processing required is to take the input sample rate (ADC data rate, for all channels combined), sizes of range and velocity FFTs, the required number of frames per second, and multiply that by the number of channels. That results in the number of FFTs of given sizes that must be performed per second. Since FFT is a "staple" function in signal processing, usually it is included in the libraries supplied with the processors, and the exact cycles required for the given size FFT and data type are published (see References [3, 4]). This can be used to estimate FFT performance requirements. On a cautionary note, typically the published benchmark data assumes ideal conditions, such as no overheads for data access, single-cycle closely coupled memory, data available in local memory, pre-filled caches etc. The actual environment to be used in the SoC, in particular the memory

subsystem, needs to be taken into account. Additional preprocessing, such as windowing, must be factored in as well. Note also that data transfers can create bottlenecks comparable to or worse than the actual computations, therefore this must be considered from the early design stages. If the system relies on any resource sharing, the combined requirements for the supported use-cases must be estimated for processor cycles, memory buses, memory sizes, etc. ADAS applications usually have strict real-time requirements, therefore worst case conditions are of most interest.

For example, consider the case of 8-channel RADAR processing, with each channel sampled at 16 MSPS, and range and velocity FFTs sizes like those used earlier in the *Signal Processing Flow and Computational Requirements* section. The combined sample rate of 16*8=128 MSPS fits nicely into the range that can be covered with pipelined one sample per cycle FFT accelerators with 128 MHz clock rate, offloading constant heavy-duty FFT processing from the processor core. It might make sense to do FFT processing "on-the fly" getting data from the ADC without writing to memory. There is a fairly wide tradeoff space with such accelerator design; since the eight channels are independent of each other, and FFTs within the packet are independent of each other, the amount of parallelism within the accelerator and its clock rate can be traded off (e.g., going down to 16 MHz rate for each of eight parallel units). The choices also affect buffering requirements and system latency. Direction of arrival computation for an 8-channel case, if done using FFT, can be handled using a scalar DSP core such as Synopsys' DesignWare® ARC® HS47D Processor. The same core can take care of higher-level lower-bandwidth processing tasks, like CFAR, thresholding, higher-level filtering operations and further decision-making. Figure 8 shows such a system, highlighting components available from Synopsys.

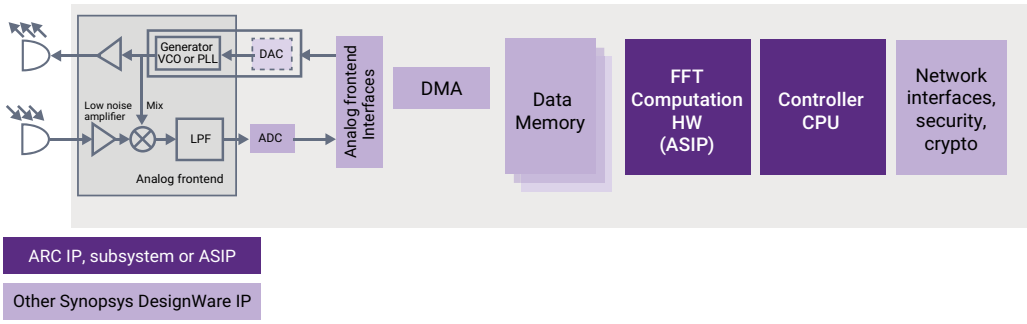

Figure 8: Example RADAR DSP components

Higher-end applications might require a vector processor, like those in the Synopsys' DesignWare ARC EV6x Processor IP family instead of, or in addition to ASIP FFT accelerators and scalar processor cores. Compared to a dedicated accelerator, a vector processor could handle more varied tasks. One example is using high resolution methods to estimate signal parameters. This involves intensive linear algebra on matrices, which is typically vectorizable. A dedicated accelerator for that might be too complex and inflexible. The tradeoff is compute resources (area and power of vector processor and memories) versus estimation accuracy. In the case of angle of arrival estimation, it could be worth trading compute resources versus RX and TX circuitry and power, reducing overall system power consumption with a more complex processor core and algorithms.

Since higher-end ADAS applications benefit from integration of vision and RADAR data, it makes sense to do such high-bandwidth fusion using programmable vector processors.

Due to the automotive life-cycle requirements and the current pace of technology, any ADAS system must be designed with enough flexibility and upgradeability in mind, which strongly suggests programmable versus hard-coded RTL solutions.

Conclusion

This paper reviewed algorithms used in FMCW RADARs, including approaches used to estimate their computational requirements and some of the system design considerations. Depending on the application, the required processing requirements can vary greatly. Synopsys offers a wide range of fully-programmable control, DSP, and combined control/DSP ARC processor IP, as well as other IP components for peripherals, interfaces, memories, and more. For specialized tasks or higher compute loads, designers can create application-specific accelerators with Synopsys' ASIP Designer tool. The combination of processing solutions available from Synopsys enable design teams to create highly-efficient and differentiated SoCs for RADAR/LiDAR applications.

References

1. DesignWare ARCv2 ISA Programmers Reference Manual for ARC HS Processors. Synopsys.
2. MetaWare DSP Programming Guide. Synopsys.
3. DSP Library Performance Databook for ARC HS Processors. Synopsys.
4. DSP Library Performance Databook for ARC EM Processors. Synopsys.
5. MetaWare Fixed-Point Reference for ARC EM and ARC HS. Synopsys.
6. ASIP Designer. Overview of the Manuals.
7. Example processor models. ASIP Designer. Synopsys.
8. R. O. Schmidt, “Multiple emitter location and signal parameter estimation,” IEEE Trans. Antennas Propagation, vol. AP-34, pp. 276–280, Mar. 1986.
9. R. Roy and T. Kailath, “ESPRIT—Estimation of signal parameters via rotational invariance techniques,” IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-37, pp. 984–995, 1989.
10. J. Wenger, “Automotive radar—Status and perspectives,” in Proc. IEEE Compound Semicond. Integr. Circuit Symp., Oct. 2005, pp.21–25.
11. Finn, H. M. and R. S. Johnson, "Adaptive detection mode with threshold control as a function of spatially sampled clutter level estimates," RCA Review, Vol. 29, 414-464, 1968.
12. H. Rohling: Radar CFAR Thresholding in Clutter and Multiple Target Situations, IEEE Trans. Aerosp. Electron. Syst. 19 No. 4, 1983, pp. 608-621.

11/12/18.CS285881601_RADAR WP.

Pub: November 2018