On-board Processing Architecture of DLR's DBFSAR / V-SAR System

Maron Schlemon, Rolf Scheiber, Stefan Baumgartner, Sushil Kumar Joshi, Marc Jäger, and Sebastian Pasch
German Aerospace Center (DLR), Microwaves and Radar Institute, Münchener Str. 20, 82334 Oberpfaffenhofen

Abstract

For real-time Synthetic Aperture Radar applications, data must be processed and sent to the ground station efficiently. This paper describes the processing architecture of DLR's DBFSAR system with the aim of presenting recent developments of on-board radar processing. It explains how the low level optimizations were conducted and under which conditions their integration in the SAR imaging process and maritime moving target indication leads to real-time capability.

1 Introduction

Images generated by a Synthetic Aperture Radar (SAR) involve a significant amount of handling processing and transfer, leading to a bottleneck for real-time applications. In areas such as surveillance or environmental monitoring, however, it is of great interest to retrieve and interpret SAR images rapidly. To achieve this, SAR data first must be efficiently processed on-board and then sent to the ground station. Present on-board processing approaches show a variety of approaches using Field Programmable Gate Arrays (FPGAs) or Graphics Processing Units (GPUs) in order to accelerate or parallelize SAR routines [1] [2]. However, DLR's digital beamforming SAR (DBFSAR) system described in Section 3 is based on Central Processing Units (CPUs). Since the airborne hardware cannot be easily replaced or extended due to tedious and complex certification steps, only CPU-based optimizations are considered in the sections below. Solutions based on GPUs are not entirely developed yet but will be a major focus in the future.

For the proposed developments, real-time capability exists when SAR products can be produced between two recordings as shown in Figure 1. This time window exists because the aircraft needs to be realigned and readjusted for the next data take. It varies and can be extended to a certain degree, however, for the given DBFSAR system a time of 10 minutes was derived from historical data.

Figure 1 Real-time definition for the DBFSAR system

Section 2 introduces the DBFSAR system, while Section 3 explains the on-board processing hardware in detail as well as the products generated by the system. Section 4 first identifies the performance critical functions of the processing pipeline, then it deals with the techniques used to accelerate SAR routines. Finally, Section 5 shows the results of implementing the fast kernels and clarifies whether the real-time condition is met.

2 The DBFSAR/V-SAR system

As described in detail in [3], DLR developed a new advanced high-resolution airborne SAR system with digital beamforming capabilities, the so-called DBFSAR. It is operating at X-band and features 12 simultaneous receive and 4 sequential transmit channels with 1.8 GHz bandwidth each, flexible DBF antenna setups and is equipped with a high-precision navigation and positioning unit. The system is certified for operation on the Do-228 aircraft of DLR. When operated for moving target indication (MTI) purposes, a dedicated antenna system consisting of 6 receive antennas is used and the system is termed V-SAR. It acquires multi-channel SAR data in support of algorithmic research in the domain of traffic monitoring and maritime security. For this purpose it also includes CPU boards for on-board processing, as described in detail in the next section, and a LTE mobile network modem to downlink the on-board processed data products to ground, the LTE antenna being mounted below the aircraft fuselage. Together with the on-board dual-channel AIS receiver, the system is thus capable of quasi real-time demonstration of maritime surveillance.

3 Architecture of the On-Board System and Data Products

Figure 2 illustrates the on-board system architecture. It consists of three real-time processor boards (RTPs) each with an Intel central processing unit (CPU) of four cores and 16 GB of random access memory (RAM). Details about the hardware are shown in Table 1. A network attached storage (NAS) is directly connected to each RTP using the Network File System (NFS) protocol. All RTPs communicate through a 10 Gbit/s switch allowing fast data access with an average throughput of 750 MB/s. During the data acquisition step, the two dimensional raw data

of the antennas (X1-X6) is written through the receive channels (R5-R12) to the NAS drives. Figure 1 shows that, e.g., the raw data of antenna X1 and X2 is written to NAS drive 2. The wiring is fixed and cannot be changed.

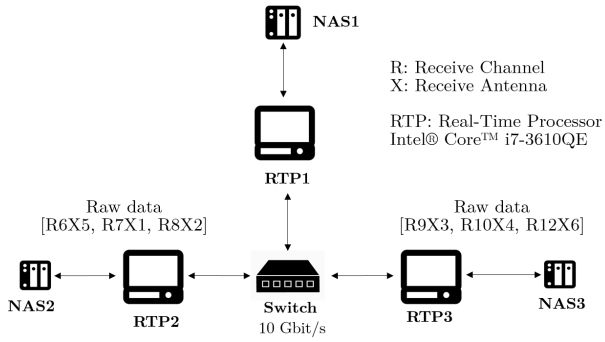

Figure 2 Sketch of the On-Board Processing System

Table 1 Central Hardware of the Processing Boards

Processor	
Name	Intel® Core™ i7-3610QE
Code Name	Ivy Bridge
Max TDP	45 W
Cores	4
Hyper Threading	Off
Base frequency	2.3 GHz
Memory	
Type	DDR3
Capacity	16384 (2x8192) GB
Operation	Dual Channel
Frequency	1333 MHz
Operating System	
Distribution	Linux / openSUSE Tumbleweed
Version	20210606 (64 bit)

Optimally, all 12 CPU cores should be used for the SAR processing, however 1 core per RTP is reserved for tasks on system level. The NAS systems also store the OmniStar HP enhanced GNSS/INS navigation solution for each recorded data take and make it available for subsequent on-board processing. The goal is to perform all processing steps for both SAR imaging and MTI on-board as described in the following sections.

3.1 SAR Image Products

The on-board processing software presently supports processing of SAR data acquired in stripmap mode up to the full resolution of the DBFSAR/V-SAR sensor. Typically, a swath width of 2-3 km is acquired and processed, depending on flight altitude. Data take duration is variable and can reach several tenths of km, which the on-board processor (OBP) can support. The resolution is up to 0.5 m in ground range, depending on the transmit chirp bandwidth. The data can be multilooked for downlink, however, geocoding is presently not performed on-board, but can be performed on ground and/or included in future. Due to the limited accuracy of the on-board navigation data and because of the approximations in motion compensation in the software, repeat-pass SAR interferometry (e.g. for coherent change detection) is not foreseen to be supported by the OBP.

3.2 MMTI Products

With the OBP also maritime moving target indication (MMTI) products are generated. For each detected ship a track [7] and a single inverse SAR (ISAR) image or, if required by the application, even an ISAR image sequence (video) is generated [6]. Tracks are composed of several track points, whereas for each track point the following information is provided:

- UTC time of the corresponding ship detection.
- Geographical position of the vessel's center in WGS84 reference system,
- Absolute speed and moving direction of the ship,
- Auxiliary information like range, Doppler frequency, bounding box size, etc.
- Link to the corresponding ISAR image.

Currently, the tracks are written into a SQLite database table. From this table all the required information can be easily extracted and, for instance, KML files for visualization with Google Earth can be generated (see example in **Figure 8**). The ISAR images are stored as PNG files and also as complex-valued arrays.

4 Algorithms and Optimizations

The constantly growing data throughput of SAR applications requires faster data handling, processing, and transfer. Therefore, the demand for on-board generation of final image products and/or compression techniques keeps increasing. Present on-board processing solutions show constraints regarding computational performance, size, and transfer speeds.

Thus, the acceleration of processing routines for on-board systems for different algorithms such as the (Extended-) Omega-k (EOK) (Figure 3) or the Maritime Moving Target Indication (MMTI) algorithm was investigated. According to Figure 3, FFTs, interpolations and algebra functions are the limiting factors for SAR image formation processing, since they are repeatedly used on large amounts of data. In order to prove that and further evaluate the developments in a performance-critical context, we used the Intel VTune Profiler to first measure the achieved performance and second to identify all relevant perfromance bottlenecks. For real-time systems it is therefore required to optimize them and thus to eliminate or reduce performance barriers. Furthermore, it is important to consider the memory bottlenecks, which are particularly significant in the case

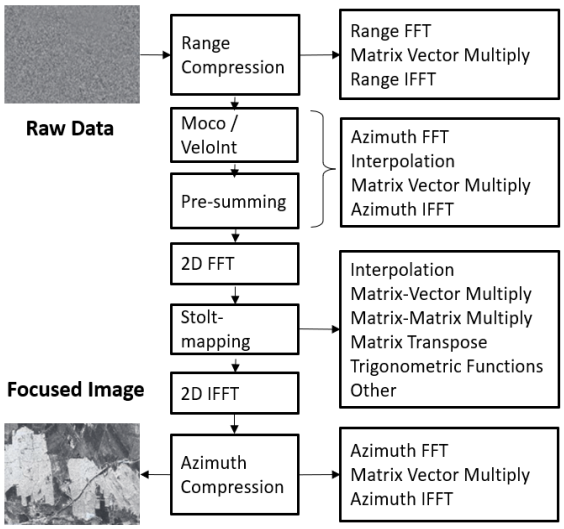

Figure 3 Block diagram of the SAR image formation with indication of elementary operations for each processing step

of non-sequential data accesses, and to minimize them by suitable data structures and processing pipelines.

Single instruction multiple data (SIMD) instructions are implemented on Intel's general purpose processors in order to enhance performance. Only a single instruction is fetched and applied to a set of data (vectors). Intel's Instruction Set Library provides the convenient solution to implement SIMD instructions [12] [13]. Therefore, it was used to vectorize the critical functions and thus create fast single core kernels (components). The SIMD optimizations for the on-board system are based on the Streaming SIMD Extensions (SSE) and on the Advanced Vector Extensions (AVX) [11].

Once the kernel performance was satisfactory, the algorithms were further parallelized and accelerated by distributing the kernels over many cores and finally over the three RTPs. In the following section, the range compression, being required for both, SAR image formation and also for MMTI, will be used as an example to present these optimizations and to provide important performance metrics.

4.1 Accelerating the Range Compression

As shown in Figure 3, the range compression consists of three processing steps: a FFT of the raw data in range direction, an element-wise complex matrix vector multiply by the reference function and an inverse range FFT in order to transfer the data back to the time domain. The FFT is based on the radix 2/4 algorithm and it has also been extensively SIMD optimized [10]. These three steps represent a matched filter which is a commonly utilized tool for data compression. However, the data has to be provided efficiently. After the data acquisition step the real-valued raw data is stored in bytes on a NAS drive. Since the kernels are based on single precision SIMD instructions, the data first needs to be read from file and loaded to the memory, then converted to floating point values and finally stored in a aligned memory space efficiently. For this purpose, a memory mapped and vectorized raw reader was developed that additionally handles the block processing by shifting every block to the next kernel. As shown in Listing 1, it first broadcasts 64 bytes into a 256 byte AVX register, then every byte is converted into an 4 byte integer and finally converted into 4 byte single precision floating point value which results in a *_m256* vector of eight floating point values. Ultimately, the generated vectors are stored in a 32 byte aligned memory which is then available to the following kernels of the processing pipeline. Tests have shown that, in addition to the vectorized reading, memory mapping is a significant part as a result of the data being projected directly into memory and thus providing faster access.

```
_mm256_cvtepi32_ps(_mm256_cvtepi8_epi32(
    _mm_broadcastq_epi64()))
```

Listing 1 Converting 8 bytes into a ymm (256 bit) vector

The processing units (Intel® Core™ i7-3610QE) of the on-board system described in Section 3 do not support the more advanced AVX2 and Fused Multiply Add (FMA) technologies which, according to **Figure 4**, leads to a performance deficit. The overall processing time for a block size of 32,768 range and 4,096 azimuth samples is for the AVX case 1.69 s and for the AVX2 case 0.91 s. The underlying data take of the Ammersee campaign (see section 5) has 285.456 samples in azimuth which translates into 70 processing blocks.

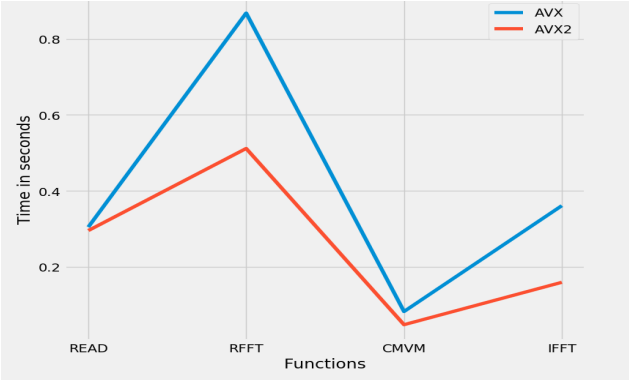

Figure 4 Comparison between AVX and AVX2 for Range Compression of (4096, 32768) Block Size

$$t_{\text{AVX}}^{\text{block}} = 1.69s \Rightarrow t_{\text{AVX}}^{\text{Total}} = 118.3s$$
$$t_{\text{AVX2}}^{\text{block}} = 0.91s \Rightarrow t_{\text{AVX2}}^{\text{Total}} = 63.7s$$

Comparing these technologies, it turns out that the AVX2 and FMA technologies lead to an improved processing time by a factor of nearly two which becomes particularly important when real-time applications need to be supported.

4.2 SAR Image Processing

The block diagram of the adopted SAR image formation processor is shown in **Figure 3**. It inherits and adapts a pre-developed Python pipeline and kernels from DLR's new offline processor presently being developed in Python. The main kernels of the processor are range compression, motion compensation, antenna pattern correction and a wavenumber domain $\omega - K$ focusing kernel. In order to reduce the processing time, the sinc-interpolations used in the velocity interpolation and motion compensation were replaced by nearest neighbour estimates. Furthermore, no topography and aperture dependent motion compensation is performed. This is reasonably accepted, as it does not cause sensible degradation of image resolution or contrast and interferometric applications must not be supported. **Figure 3** also shows the elementary blocks of the different kernels, which currently are accelerated/optimized individually using the approach already described for the range compression step. The achievable gain in processing time is shown exemplary for the azimuth re-sampling step in **Figure 5**, where the „CMVMx" stands the element-wise complex matrix vector multiplies. The blue line represent the initial Python (Numpy) solution which was served as a reference. The red line (SARCOMP) shows the performance of the same processing pipeline but using proposed optimized AVX/AVX2 kernels. Like the range compression, the velocity interpolation also shows significantly improved performance.

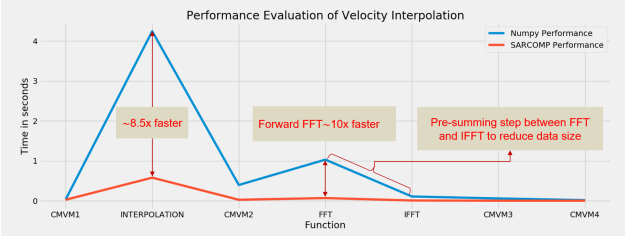

Figure 5 Optimization of the velocity interpolation (azimuth resampling) using fast kernels

4.3 MMTI Processing

A simplified block diagram of the implemented MMTI processor is shown in **Figure 6**. As input, pre-calibrated range-compressed multi-channel data is required. In the "Preprocessing" block an amplitude equalization between the channels is carried out and the remaining phase offsets are corrected. Afterwards ship detection, tracking and parameter estimation (i.e., geographical position, speed and moving direction) are carried out. In the "Radar Imaging" block ISAR images are generated which are then used for estimating the ship dimensions in the "Ship Classification" block. The results are fused with the automatic identification system (AIS) data received independently with the dual-channel on-board AIS receiver. Finally, the results are passed to the "Data Transmission" block and, thus, to the on-board LTE router which transmits the result to ground. More details regarding the processing blocks and the used algorithms are given in [6].

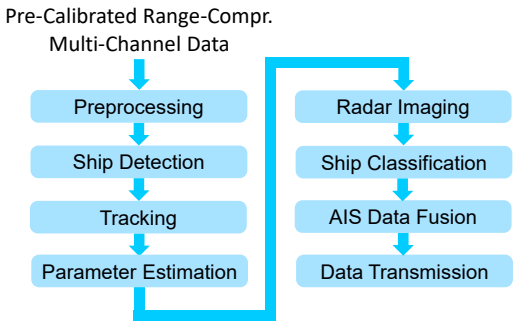

Figure 6 Simplified block diagram of the implemented MMTI processor.

5 On-board Processing Results

Although not yet fully optimized, the developed on-board processing software was implemented to a DBFSAR/V-SAR data set acquired in August 2020 across lake of Ammersee. The experiment was conducted in order to support and demonstrate the development of moving target detection algorithms.

5.1 SAR Imaging Result

Figure 7 presents an on-board processed SAR image. The data acquisition time was 1.5 minutes and the single-threaded, non-optimized computation took 12 minutes which was used as a reference time. The following multi-core processing time estimates are based on already conducted evaluation of the multi-core performance. According to them, the processing time for one receive antenna is 5 min using three cores of one RTP which meets the predefined real-time condition. However, it implies that the processing of three receive channels needs to be distributed over the three RTPs allowing the generation of three SAR images/polarizations within time slot between successive data takes. Ongoing optimizations for vectorized computations are expected to further reduce the processing time to 2-3 minutes, comparable to the data acquisition time. After this time, SAR image data can be transferred to ground via the LTE modem available in the aircraft [9].

5.2 Exemplary MMTI Results

In **Figure 8** some exemplary MMTI results obtained over the lake Ammersee are shown. The boats labelled with 1 to 5 are small electrical boats (size $\approx$ $1.5\,\mathrm{x}\,3.5\,\mathrm{m}^2$) and boat 6 is a sailing boat ($\approx$ $2\,\mathrm{x}\,5\,\mathrm{m}^2$). All of them were equipped with a handheld GPS tracker. By comparing the GPS tracks with the geocoded boat tracks obtained from the MMTI processor, it turned out that the absolute geographical position estimation accuracy is in the order of 10 m, which is a very good value for such small boats with low radar cross section. Details on the used geocoding technique can be found in [8] and ISAR image examples obtained with the implemented MMTI processor are shown in [3] and [5].

en

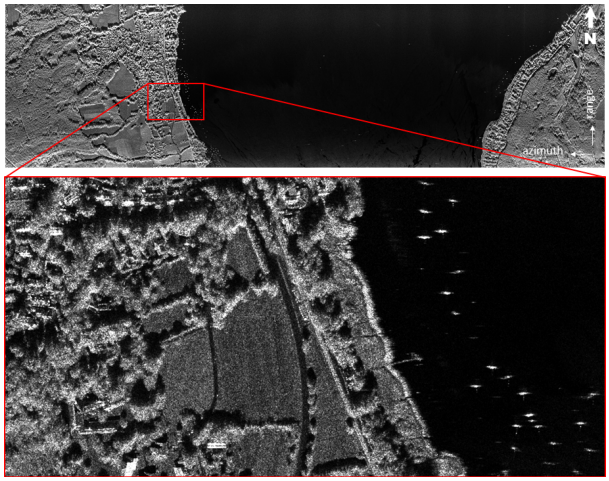

Figure 7 On-board processed SAR image result across lake Ammersee, Bavaria, X-band, VV polarisation, image dimensions are 7.5 km by 2.5 km, with zoom on sailing vessels, ground range resolution 0.5 m by 1 m.

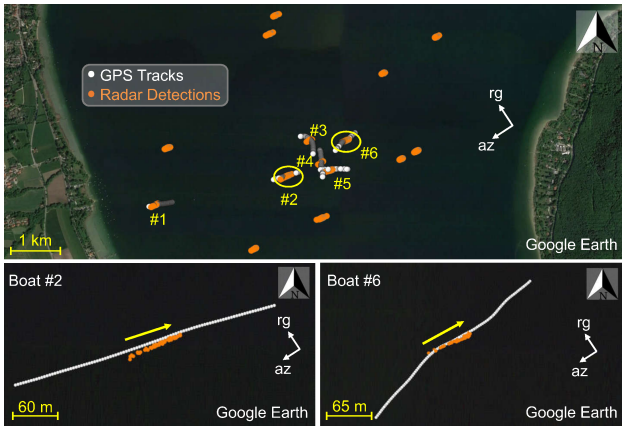

Figure 8 Detection and tracking of small boats with low radar cross section moving on the lake Ammersee (orange color: geocoded radar detections and tracks; white: GPS reference tracks).

The MMTI algorithm was not optimized yet. It runs on one RTP using a single core only. The total processing time in the tests conducted, was 5:50 min which already matches the real-time condition introduced in Section 1. The future work consists of optimizing the algorithm by utilizing multi cores as well as the other RTPs which will improve the run time.

6 Summary and Outlook

It has been shown that the low level optimizations accelerate SAR routines and thus the overall SAR application. However, future work consists of further improvement of the algorithms mainly by utilizing all hardware resources optimally. From the research that has been carried out, it is possible to conclude that by definition DLR's DBF-SAR system is capable of processing SAR data in real-time when using all the resources effectively. This will be verified in next years North Sea campaign.

7 Literature

[1] J. Doubleday and S. Chien and Y. Lou and D. Clark and R. Muellerschoen: Onboard Autonomous Response for UAVSAR as a Demonstration Platform for Future Space-Based Radar Missions. Acta Futura, vol. 9, pg. 31-39, 2014

[2] D. Romano and V. Mele and M. Lapegna: The Challenge of Onboard SAR Processing: A GPU Opportunity, Computational Science – ICCS 2020, Springer International Publishing: Cham p. 46-59

[3] A. Reigber et al: The High-Resolution Digital-Beamforming Airborne SAR System DBFSAR. MDPI Remote Sensing. vol. 12, no. 11, pg. 1710, 2020

[4] R. Que and O. Ponce, and St. Baumgartner, and R. Scheiber: Multi-mode Real-Time SAR On-Board Processing. Proceedings of EUSAR, 2016

[5] St. Baumgartner, and S. K. Joshi: Efficient Clutter and Noise Filtering for Improved ISAR Imaging. Proceedings of EUSAR, 2021

[6] S. Baumgartner and S. K. Joshi, "Onboard processing concept for maritime surveillance demonstrated with DLR's airborne radar sensors F-SAR and DBF-SAR", 13th European Conference on Synthetic Aperture Radar. Online: VDE, 2021, pp.1-6.

[7] S. K. Joshi, S. Baumgartner and G. Krieger, "Tracking and Track Management of Extended Targets in Range-Doppler Using Range-Compressed Airborne Radar Data," in IEEE Transactions on Geoscience andRemote Sensing, vol. 60, pp. 1-20, 2022.

[8] S. K. Joshi, S. Baumgartner, A. Barros Cardoso da Silva, and G. Krieger, Direction-of-arrival angle and position estimation for extended targets using multichannel airborne radar data, IEEE Geoscience and Remote Sensing Letters, vol. 19, pp. 1-5, 2022.

[9] S. Baumgartner, and A. Nottensteiner and D. Rosigkeit: Usability of LTE for Transmitting Radar Data from DLR's Research Aircraft DO228-212. Proceedings of 36th European Telemetry and Test Conference - etc2016, pp.181-187, Nuremberg, Germany, 2016.

[10] M. Schlemon and J. Naghmouchi, "FFT Optimizations and Performance Assessment Targeted towards Satellite and Airborne Radar Processing," 2020 IEEE 32nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2020, pp. 313-320, doi: 10.1109/SBAC-PAD49847.2020.00050.

[11] A. Fog, "Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD, and VIA CPUs," Technical University of Denmark

[12] Intel Corporation, "Intel® 64 and IA-32 Architectures Software Developer's Manual,"

[13] Intel Corporation, "Intrinsics Guide," https://www.intel.com/content/www/us/en/docs/intrinsics-guide/