Architecture of a SAR Range-Doppler processor with real time requirements using a general purpose computer

Sérgio Henrique Trofino [1,2]
Roberto d'Amore [2]
David Fernandes [2]

[1] Mectron - Engenharia, Indústria e Comércio S.A.
Av. Brigadeiro Faria Lima, 1399 - 12227-000 - São José dos Campos - SP, Brasil
sergio.trofino@yahoo.com.br

[2] Instituto Tecnológico de Aeronáutica - ITA
Praça Marechal Eduardo Gomes, 50 - 12228-900 - São José dos Campos - SP, Brasil
{damore, david}@ita.br

Abstract. Synthetic aperture radars (SAR) have many applications and the increasing demand for real-time processing has motivated this development. There are no reports in the literature about the use of low cost platforms for SAR images processing with real time requirements using, such as general purpose computers (PC) and that don't use auxiliary processors, such as graphics cards processors. It was implemented a real time SAR image processing system based on the Range-Doppler algorithm which can run in a lap-top computer. Two scenarios were used to study the proposed system, one is a simulated scenario containing three ideal point reflectors and the other one is a real scenario obtained from ERS-1 SAR. The implemented image processor was evaluated by the achieved resolution of ideal point reflectors after the image synthesis, the time of the processing from one and ten frames of each scene, and the notion of continuity in displaying frames. The ideal point reflectors resolution in the SAR image was within the expected limits, and the processing time allows a real time operation without data loss. The available time for processing indicates the possibility of the implementation of other features to the SAR processor, such as motion compensation or tracking or target recognition, especially if the platform is expanded, for example, using graphics coprocessors.

Key words: synthetic aperture radar, real time SAR images processing, SAR images processing using PC.

1. Introduction

The use of imaging radars, such as synthetic aperture radar - SAR, can be found on orbital and aerial platforms (e.g.: unmanned aerial vehicles). The demand for fast information acquisition and real time image evaluation take to the challenge of real time SAR image processing.

Real time SAR processing was discussed in Guarita (2011), Mura et al. (2011), Malanowski et al. (2013), Song et al. (2013), Chiaratti and Fernandes (2013), Trofino et al. (2012) and Wu et al. (2012), among other works. In Guarita (2011), Song et al. (2013) and Chiaratti and Fernandes (2013) the SAR processing considered the performance for real time operation. In these works, however, it was not discussed the interfaces (raw data input and processed image output), the tasks processing time and its integration to the processor. In Malanowski et al. (2013), it was presented a SAR processor operating in real time, running in a GPU's notebook (Graphics Processing Unit), having the telemetry link of a SAR system onboard an aircraft as input data. In Wu et al. (2012), it was presented a complete system, which was tested using data from an onboard SAR in a satellite and it was chosen the Chirp Scaling algorithm for data processing, due to being considered more appropriate to run on a GPU. Because of the great demand in processing in mathematics operations, reports are not found on the literature describing the use of low cost platforms for SAR images processing such as general purpose computers (PC).

This paper proposes the use of a PC to process and display a focused SAR image using the Range-Doppler algorithm without motion compensation. The tasks consider the data inflow and outflow in a way that it does not accumulate non-processed data. A test system

which displays focused SAR images from raw data was also developed. The raw data can either be originated from another PC, simulating operating radar, or can be generated in the same computer where the processing happens.

The processing operations and real-time visualization of the developed system were evaluated using input data representing ideal point reflectors and using real data of the ERS-1 (European Remote Sensing Satellite).

2. Proposed architecture and test scenarios

A Range-Doppler processor focuses an image from the received echo signal performing the following steps: signal compression in range, resolution cell migration correction (RCMC) and signal compression in azimuth. The signal compression, both in range and in azimuth, is the convolution of the received signal with the echo function representative of the spreading of a point target filtered through a Hamming window. The use of the Hamming window causes a spread of 30% in the original resolution, as seen in Elarrat (2012), however, it attenuates the secondary side lobes of the point spread function in -32dB in relation to the main lobe. The diagram representing the implemented Range-Doppler algorithm is shown in Figure 1.

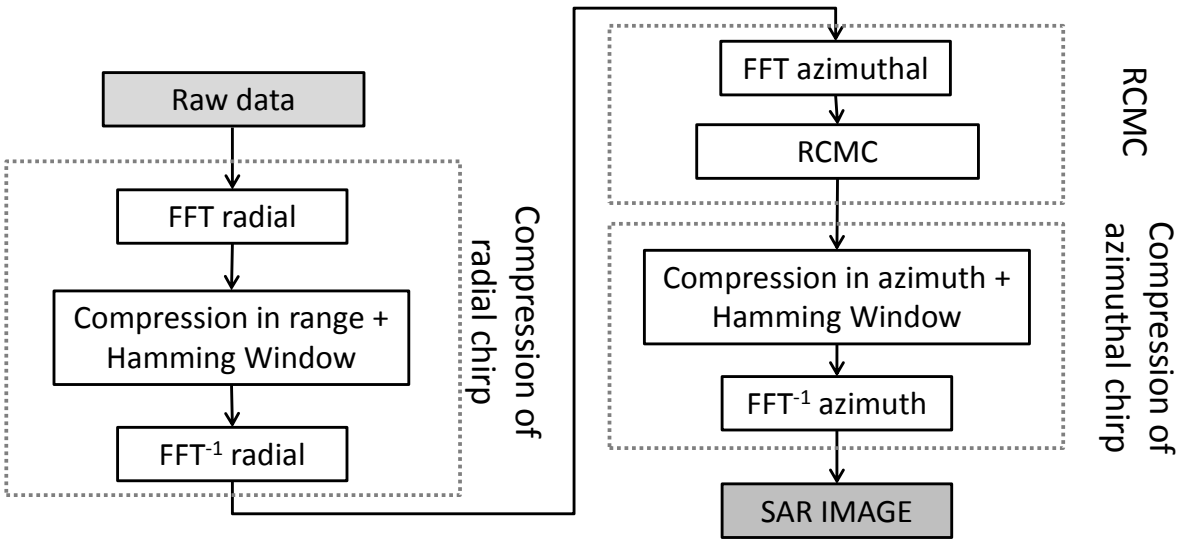

Figure 1. Diagram of the Range-Doppler algorithm implemented.

The proposed processor architecture contains two main applications: SimSAR and ProcSAR. The SimSAR simulates the SAR sensor, sending raw data to ProcSAR through a network communication port (socket). The data sent by SimSAR can be originated from the Scenario Generator application or a file containing a recorded scenario. Both applications use a file containing the parameters of the SAR scenario (platform speed, altitude, antenna aperture, etc.) and parameters of the processing system (size of the preview window, the origin of the scenario - simulated or file, etc). This architecture is shown in Figure 2.

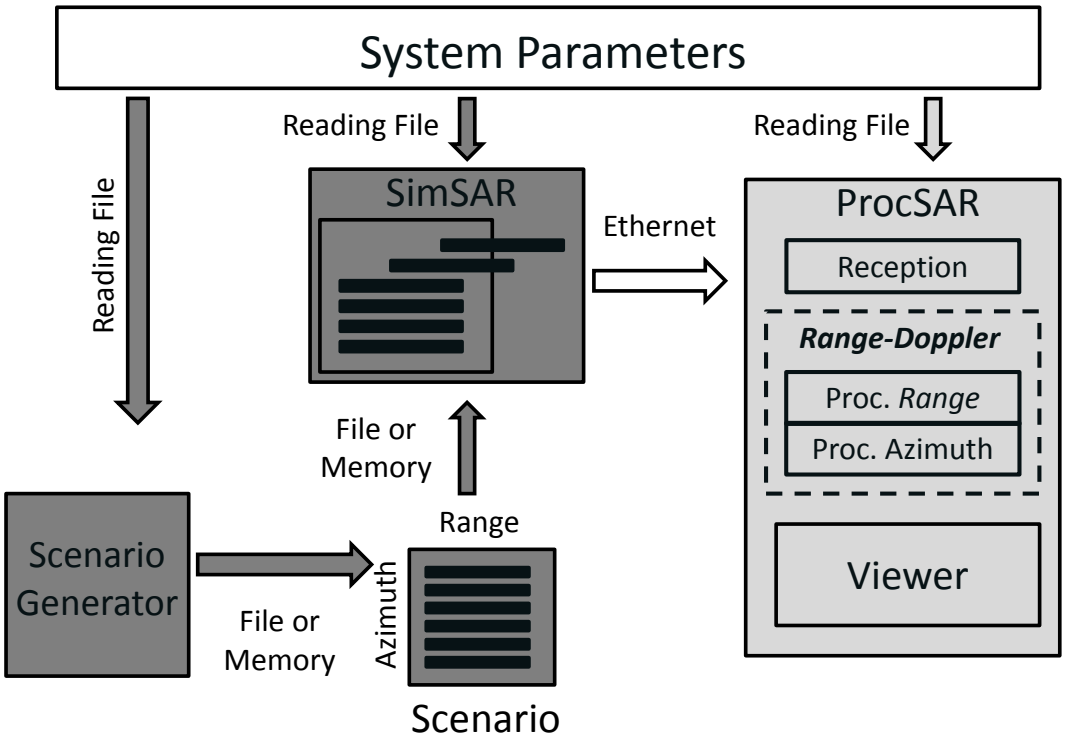

Figure 2. System architecture.

SimSAR and ProcSAR may run in two separate computers. In this condition, ProcSAR operates as a real system, where data is provided by a SAR system. This is possible due to the use of socket that makes the communication among processes via the network card. This configuration allows an easy replacement of SimSAR by a real acquisition system. This alternative was tested. However, it was more convenient to run both applications on the same computer, due to the possibility of performing all operations on a single hardware without affecting the processing.

The SimSAR block operates in two ways. In the first option, a simulated scenario is generated, stored in the PC memory and sent cyclically to ProcSAR. In the second option, a scene stored in a file is read by SimSAR, transferred to the PC memory, and sent cyclically to ProcSAR. The cyclic way to send the raw data is due to the limited scenario's size used. When the SimSAR reaches the last sample of a scenario, it starts to send the same scenario again. This process is repeated according to a parameter defined in the parameter file.

The ProcSAR block implements the Range-Doppler algorithm and the operation, which allows the visualization of the processed SAR image. The algorithm coding followed the diagram shown in Figure 3. The structure of the Range-Doppler algorithm allows to parallelize the compression in range with the RCMC and the compression in azimuth. Moreover, the image displaying can also be done in parallel with other processor's tasks.

To distribute processing tasks efficiently in the computer's cores, ProcSAR was divided into four sub-application (processing threads): a sub-process that deals with the reception of the raw data coming from SimSAR; a sub-process that deals with the compression in range; a third sub-process that takes care of the RCMC and compression in azimuth; and a last sub-process that handles with the image display. This process division is illustrated in Figure 3.

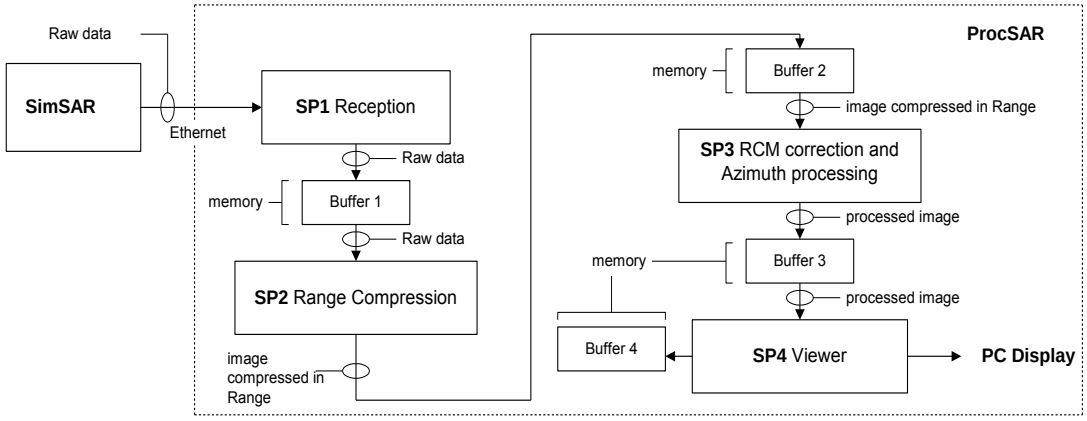

Figure 3. Diagram depicting the structure of ProcSAR.

The FFTs (Fast Fourier Transforms) calculation of the Range-Doppler algorithm employed a FFT library called FFTW, described by Frigo and Johnson (2005). This library allows the hardware test to evaluate the execution time of the FFTs' algorithms and thus select the most efficient one.

After the processing of the SAR image, due to the edge effect of the convolution, it is necessary to discard a certain amount of image data in range (M_d) and in azimuth (N_d) (Chiaratti and Fernandes, 2013).

Considering the operation of the SAR sensor, where a set of M samples in range is obtained in a pulse repetition period (T_0), in order to process the SAR image in real time, the compression in range of each set of M samples of the scene must be performed in a time period equal or lower than T_0. Furthermore, the RCMC and the compression in azimuth require the formation of a block with $N \times M$ data samples, where N is the number of echoes signals. This way, besides the processing of M data in a period of time less or equal to T_0, it must be processed $N \times M$ data in a time equal or less to $N \times T_0$. In this work, it was choose to process all M range data samples, disconsidering the discard of range data, since there is no division in the data block in this dimension. This way, the timeout for performing compression in range remains T_0. In azimuth, to present a continuous image by joining the processed frames, each data block with size of $N - N_d$ uses N_d samples of the data block processed in the previous iteration, which implies the acquisition of $N - N_d$ new data to generate a new data block, as explained by Trofino (2014). This way, the time limit for performing RCMC and azimuth compression becomes $(N - N_d) \times T_0$, which is the time required to form the new data block, corresponding to an acquisition of $N - N_d$ new samples (Chiaratti and Fernandes 2013).

Two scenarios were used for testing and validation of the developed system: a simulated scenario and a real scenario obtained from ERS-1 orbital platform. The simulated scenario is a non-reflective scene containing three ideal point reflectors according to the requirements of a operation mode of a real time simulated SAR system developed at ITA with support from FINEP (AEROSAR project agreement 01.10.058.00). The raw image (real part) of the simulated scenario is shown in Figure 4a) and Figure 4b) shows the raw data (real part) of the ERS-1 scenario.

Figure 4. Raw images of the test scenarios: a) real part of the raw data of the simulated scenario and b) real part of the raw data of ERS-1 scenario.

Table 1 shows the parameters of the SAR processing system considered for the test scenarios.

Table 1. Parameters for generation and / or processing of test scenarios.

Symbols	Description	Simulated scenario	ERS-1 scenario
Radar			
λ	Wavelength	0,03 m	0,056666 m
T_0	Period of repetition of pulse	625 μs	595 μs
T_p	Pulse duration	667,13 ns	37,12 μs
f_s	Sample frequency	300 MHz	18,962468 MHz
ϕ	*Squint angle*	1,0 °	0,112 °
γ_R	*Chirp rate*	449,3 THz/s	23,2464 THz/s
L_a	Effective length of antenna	0,25 m	12 m
M	Number of range samples	2048 pixels	2048 pixels
N	Number of azimuth samples	8192 pixels	2048 pixels
N_d	Number of discarded azimuth samples	323 samples	780 samples
δ_R	Range resolution	3 m	10 m
δ_A	Azimuth resolution	3 m	10 m
Platform			
h	Altitude	4 km	780 km
v_p	Speed	200 m/s	7120,7327 m/s
θ_a	Azimuthal aperture of antenna	6,88 °	0,271 °
Scenario			
Azimuth	Azimuth imaged strip	2 km	8,68 km
Range	Range imaged strip	1,2 km	16,2 km

3. System evaluation methods and results

The evaluation of the implemented system analyzed: the frame processing time; the processing time of 10 frames; the ideal point reflectors image resolution; and the continuity of the displayed image frames.

The measurement of the tasks execution time used the omp_get_wtime() function, available in OpenMP software library described by Mattson (2011).

The resolution analysis (energy dispersion) of the ideal point reflectors considered a the range between 70% of the point target peak amplitude. This dispersion is the resolution of the image listed in the parameter's scenario (Table 1).

Also, it was analyzed the continuity of the displayed image by the system with the images sent cyclically by SimSAR.

The hardware platform configuration and applications used for the tests are listed in Table 2.

Table 2. Features Hardware and Software used.

Processor	Intel Core i7 - 2630M	Linux API	Cigwin 64 bits v2.831
RAM Memory	6 Gigabytes	C compuiler	GCC 64 bits v4.8.2
Video card	GT550M - 1 Gigabyte	Mathematical SW	IDL v7.1.1
O.S.	Windows 7 64 bits SP1	Graphics API	OpenGL
FFT API	FFTW v3.3	File API	libxml2 v2.9.1

In Figure 5 it is depicted the images obtained for the three ideal point reflectors and for ERS-1. The ERS-1 scenario is a lake in Austria (darker region) and towns (bright spots).

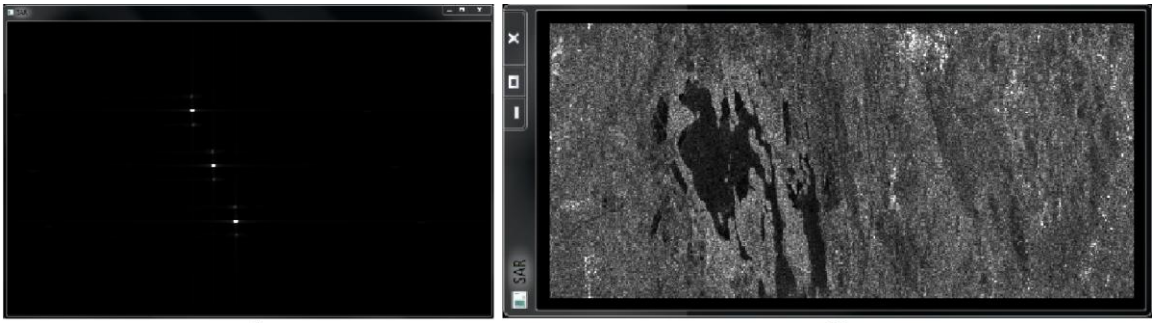

Figure 5. Processed images of the test scenarios: a) Image of the simulated scenario; b) Image of ERS-1 scenario.

The curves representing the spreading of the energy of the echo signal of a processed image, in the azimuth direction and in the radial direction, allow the resolution evaluation. Figure 6a and Figure 6b show these curves for the simulated scenario (Figure 5a). Considering as image resolution the signal scattering in a range up to 70% of the maximum amplitude of the echo signal. The obtained image has a resolution of approximately 4 m in both directions. This image focusing can be considered satisfactory because the use of the Hamming window increases in 30% the signal resolution, extending the ideal image resolution from 3m to approximately 4m.

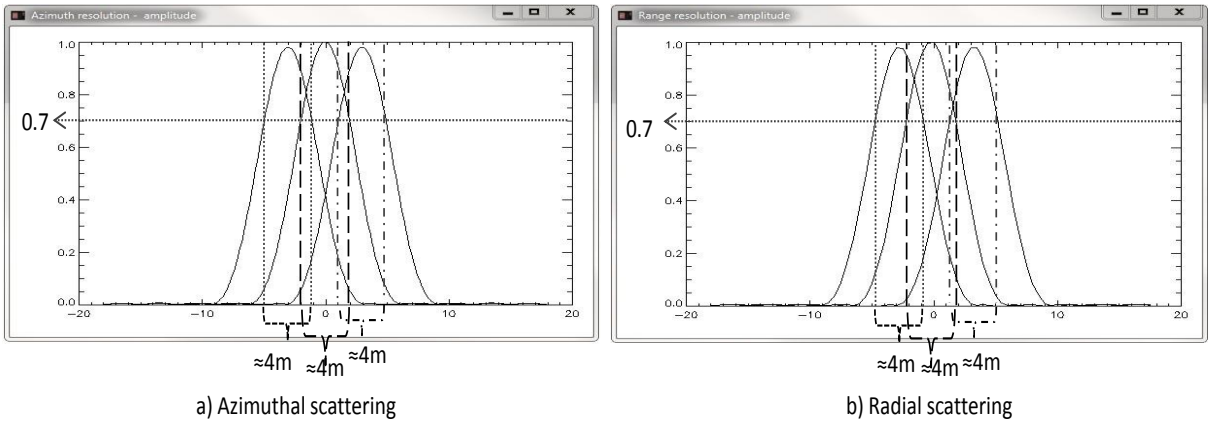

Figure 6. Scattering power of the targets of the simulated scenario: a) scattering in azimuth; b) scattering in range.

The scenarios processing time, and real-time operation timeouts are shown in Table 3. The timeout for range processing is equals the pulse repetition period (T_0) of each scenario. The azimuth processing time, is the time to perform the RCMC and compression in azimuth, which corresponds to $(N - N_d) \times T_0$ of each scenario. The values used to calculate these limits are listed in Table 1.

Table 3. Processing times and timeouts for scenarios processing.

	Processing time		Timeouts	
	Simulated scenario	ERS-1 scenario	Simulated scenario	ERS-1 scenario
Range processing	47.7 μs	44.9 μs	625 μs	595 μs
Azimuth processing	2.54 s	0.562 s	4.92 s	0.754 s
Time of processing of 10 *frames*	27.2 s	6.06 s	49.2 s	7.54 s

A continuous imaging was achieved with the system in operation. Figure 7 shows three frames of ERS-1 scenario in different time instants.

Figure 7. System implemented in three stages, illustrating the display of three frames of ERS-1 scene, forming a continuous image.

4 Conclusions

It is possible to use a PC as a platform to implement a SAR processor for real-time operation. Through this study, it was shown the processing of a simulated scenario based on a SAR (onboard an aircraft - Project FINEP AEROSAR), and a real scenario data from the ERS-1 SAR (sensor onboard an orbital platform). It was achieved a correct image focusing, and a processing interval below time limits. This margin of time allows the implementation of further implementations (such as motion compensation) or image processing (such as segmentation image, pattern detection, etc.).

The use of additional co-processors such as GPUs, can also be explored increasing the system processing availability and allowing the expansion of SAR processor features.

Acknowledgements

The authors acknowledge the support of the Project FINEP AEROSAR, 01.10.058.00 agreement, developing the ITA and having FUNDEP as a signatory.

References

Chiaratti, A.; Fernandes, D. Avaliação da influência de requisitos operacionais na geração de imagens SAR em tempo real. In. Simpósio de Aplicações Operacionais em Áreas de Defesa (SIGE), 15., 2013, São José dos Campos. **Anais...** São José dos Campos, DCTA, 2013.

Elarrat, I. F. **Detecção e estimação de velocidade de alvos móveis em imagens SLAR utilizando multicanais.** 2012. 122 f. Dissertação (Mestrado em Engenharia Eletrônica e Computação) - Instituto Tecnológico de Aeronáutica, São José dos Campos. 2012.

Frigo, M.; Johnson, S. G. The design and implementation of FFTW3. **Proceeding of the IEEE**, v. 93, n. 2, p. 216-231, Feb. 2005.

Guarita, F. C. **Avaliação da arquitetura CUDA para síntese de imagens SAR.** 2011. 90 f. Dissertação (Mestrado em Engenharia Aeronáutica) - Instituto Tecnológico de Aeronáutica, São José dos Campos. 2011.

Malanowski, M.; Krawczyk, G.; Samczyński, P.;Kulpa, K.; Gromek, D.. Real-time high-resolution SAR processor using CUDA technology. In: International Radar Symposium (IRS), 2013, Dresden. **Proceedings...** Piscataway: IEEE, 2013. v. 2, p. 673-678.

Mura, J. C.; Gama, F. F.; Bins, L. S.; Jorge, D. G. T.; Silveira, G. F.. Processador SAR compacto baseado em FPGA para monitoramento em tempo real. In: Simpósio Brasileiro de Sensoriamento (SBSR), 15., 2011, Curitiba. **Anais...** São José dos Campos: INPE, 2011. p. 7572-7579.

Song, M.; Liu, Y.; Zhao, F.; Wang, R.; Li, H. Processing of SAR data based on the heterogeneous architecture of GPU and CPU. In: IET International Radar Conference, 2013, Xi'an. **Proceedings...** Stevenage: IET, 2013. p. 1-5.

Trofino, S. H. **Implementação de um processador SAR Range-Doppler em um computador pessoal visando operação em tempo real.** 2014. 113 f. Dissertação (Mestrado em Engenharia Eletrônica e Computação) - Instituto Tecnológico de Aeronáutica, São José dos Campos. 2014.

Trofino, S.; Fernandes, D.; d'Amore, R. Avaliação do processador SAR Range-Doppler para operação em tempo real em um computador de uso geral. In: Simpósio de Aplicações Operacionais em Áreas de Defesa (SIGE), 14., 2012, São José dos Campos. **Anais...** São José dos Campos: DCTA, 2012.

Wu, Y.; Chen, J.; Zhang, H. A real-time SAR imaging system based on CPUGPU heterogeneous platform. In: International Conference on Signal Processing (ICSP), 11., 2012, Beijing. **Proceedings...** Piscataway: IEEE, 2012. v. 1, p. 461-464.