Optimization Bottleneck Analysis in GPU-Based Aiming at SAR Imaging

Wang Shi-Yu(✉), Zhang Sheng-Bing, An Jian-Feng, Huang Xiao-Ping, and Wang Dang-Hui

School of Computer Science,
Northwestern Polytechnical University, Xi'an 710129, China
onion0709@mail.nwpu.edu.cn

Abstract. Application Defect induced by GPU Aiming at SAR Imaging are studied. It is the first time the issue of application defect induced by GPU is addressed in SAR field. In GPU-based SAR imaging system, application defect induced by resources competition can significantly decrease the granularity of parallelism. To solve this problem, the GPU-based SAR imaging system with CUDA is firstly modeled. Secondly, conditions of parallel granularity loss rate by using CUDA are obtained based on time output feedback scheme. Thirdly, more importantly, find the difficulties and bottlenecks in the optimization of SAR imaging operation is proposed according to the measured conditions of parallel granularity loss rate. Finally, optimization bottleneck analysis through FFT function and linear matrix interpolation scheme, and numerical simulations are made to demonstrate the effectiveness of the proposed scheme.

Keywords: GPU · SAR imaging · Parallel computing · Optimization bottleneck analysis

1 Introduction

SAR (Synthetic Aperture Radar) remote sensing data, with its unique advantages is gradually used in the field of earth observation. The amount of data contained in a SAR image is usually very large. Considering that the traditional serial processing method has a certain lag, in practical application, operation rate is one of the key factors of SAR image processing.

The current SAR imaging processing algorithms mainly include RDA algorithm, CS (Scaling Chirp) algorithm, and the ωKA algorithm. The three algorithm flowcharts are shown in Fig. 1.

At present, most of the SAR imaging systems based on the CPU of the personal computer, workstation or large computing server, are carried out on such hardware in the system. Besides, in order to make the CPU-based imaging system be fully applied and further accelerate the speed of SAR echo data processing, the design of the software architecture also needs a lot of human input.

CUDA (Compute Unified Device Architecture) providing a parallel programming model and software environment will fully mobilize the powerful parallel computing

F. Chen and Y. Luo (Eds.): Industrial IoT 2017, LNICST 202, pp. 43–52, 2017.
DOI: 10.1007/978-3-319-60753-5_5

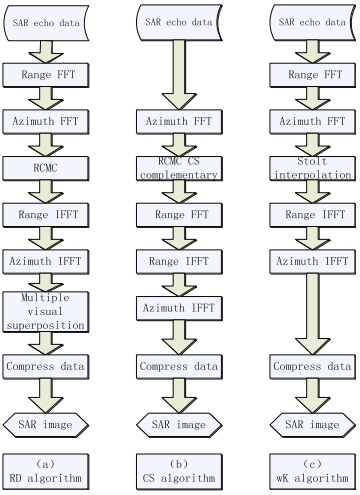

Fig. 1. Three kinds of SAR algorithm flow

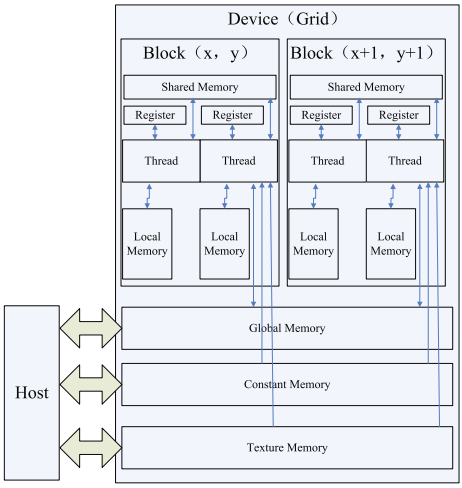

Fig. 2. CUDA storage model

power of GPU, so that GPU can play its inherent advantages in solving complex computational problems. Hierarchical management is shown in Fig. 2.

With its rapid development, GPU is widely used to solve the problem of the calculation of massive remote sensing data processing [3]. However, using GPUs for scientific computing has been mostly dominated by those with needs for a large number of tightly-coupled floating-point operations such as the n-body problem [4].

At present, the research on the acceleration of SAR imaging is mainly reflected in the improvement of the SAR algorithm [6] and the efficient application of the CUDA architecture [7–9]. In this paper, the CUDA architecture is used to accelerate the SAR imaging, identify the bottlenecks in the process of acceleration, and point out the focus of the work for future SAR imaging.

The paper is structured as follows: The second part analyzes the specific algorithm of SAR imaging and the details of the formula. And the parallelism of the SAR imaging arithmetic unit is analyzed and designed in the third part, while the fourth part presents the experimental data and the parallel bottleneck analysis. The conclusion is given in the fifth part.

2 The Analysis of SAR Imaging Arithmetic Operator

From the previous description, SAR imaging steps can be summed up, although the imaging algorithms have different characteristics and advantages, but each SAR imaging algorithm basically contains FFT (IFFT), phase multiplication and interpolation of these three kinds of computing components. Most of the computing cost also comes from the three arithmetic units (Calculation Sample: 1024 $\times$ 1024 SAR image data matrix).

2.1 FFT (IFFT) Arithmetic Operator

N finite length sequence of X (n) DFT and IDFT operation process is as follows (Table 1):

$$X(k) = DFT[x(n)] = \frac{1}{N}\sum_{n=0}^{N-1} x(n) W_N^{nk} \quad \mathrm{k} = 0, \ldots, \mathrm{N} - 1 \tag{1}$$

$$x(n) = IDFT[X(k)] = \frac{1}{N}\sum_{n=0}^{N-1} X(k) W_N^{-nk} \quad \mathrm{n} = 0, \ldots, \mathrm{N} - 1 \tag{2}$$

$$W_N^{nk} = \exp(-2nk\pi i/N) \tag{3}$$

Table 1. DFT calculating amount

Amount of computation	N-point DFT	Complex multiplication	Complex addition
DFT[x(n)]	N-point X (k)	N^2	N (N – 1)

2.2 The Interpolation of Arithmetic Operator

SAR image processing achieves the sample position mainly by the interpolation. In accuracy and computation requirements, 8 point sinc interpolation is generally used. What's more, Sinc interpolation method concerns the use of the original function y in the value of the other points, weights sinc function, and gets the value of Y (x) at X.

The interpolation operation procedure is as follows:

$$y(x) = \sum_{i=-\infty}^{\infty} y(i) \sin c(x - i) \tag{4}$$

$$\sin c(x) = \frac{\sin(\pi x)}{\pi x} \tag{5}$$

Therefore, it is necessary for sinc interpolation truncation, the P sinc interpolation, and interpolation formula for range curvature correction (Table 2):

$$s'(m, n) = s(m, n + \Delta n) = \sum_{i=-p/2}^{p/2-1} s(m, n + n' + i) \sin c(franc - i). \tag{6}$$

Table 2. Interpolation calculating amount

Amount of computation	N-point Y (X)	Sine calculation	Complex multiplication
Y (x)	8-pointX (k)	8 N	8 N

2.3 Phase Multiplication Arithmetic Operator

In the process of the CS imaging algorithm, the three phase multiplication algorithm is included. The signal in the form of Doppler domain is as follows:

$$\begin{aligned} S_1 &= (f_t, \tau, r) \\ &= CG(-\frac{r\lambda f_1}{2v^2})m(-\frac{2R_f(f_t,r)}{c})\exp\{-j\pi K_m(f_t,r)\cdot \\ &[\tau - \frac{2R_f(f_t,r)}{c}]^2\}\cdot \exp\{-j\frac{4\pi r}{\lambda}\gamma(f_t)\} \end{aligned} \tag{7}$$

The first phase is multiplied in the azimuth FFT, the completion of the CS is realized by the RCMC operation:

$$S_2(f_\tau, \tau, r) = S_1(f_\tau, \tau, r)\cdot H_1(f_\tau, \tau, r). \tag{8}$$

In Formula (8):

$$H_1(f_t, \tau, r) = \exp\{-j\pi K_m(f_t, r_{ref})\cdot C_s(\tau - \tau_{ref})^2\}. \tag{9}$$

$$\tau_{ref} = \frac{2}{c} r_{ref}[1 + C_s(f_t)] \tag{10}$$

τ_{ref} is generally used as the center of the imaging and mapping band as well as a fixed reference distance.

The second phase is multiplied by the distance to FFT, as well as the completion of RCMC, SRC and distance compression:

$$S_3(f_t,f_\tau) = S_2(f_t,f_\tau)\cdot H_2(f_t,f_\tau). \tag{11}$$

In Formula (11):.

$$H_2(f_t,f_\tau) = \exp\{-j\pi\frac{f_\tau^2}{K_m(f_t, r_{ref})[1 + C_s(f_t)]}\}\exp\{j\frac{4\pi}{c} r_{ref} C_s(f_t) f_\tau\}. \tag{12}$$

Third phase multiplication in the distance to IFFT after the completion of the azimuth compression and phase correction:

$$S_4(f_t, \tau) = S_3(f_t, \tau)\cdot H_3(f_t, \tau). \tag{13}$$

In Formula (13) (Table 3):

$$H_3(f_t, \tau) = \exp\{j\frac{2\pi}{\lambda} c\tau\cdot\gamma(f_t) + j\Delta\}. \tag{14}$$

Table 3. Phase multiplication calculating amount

Amount of computation	N-point Y (X)	Complex multiplication
Y (x)	N-point X (k)	N

$$\Delta = \pi K_m \frac{C_s(\tau - \tau_{ref})^2}{1 + C_s} \tag{15}$$

3 Parallel Architecture for Computing Components

Multiple FFT parallel computing, data exist before and after dependence [12, 13]. Therefore, in the calculation process, FFT calculation process in each block there is a pipeline to wait for the calculation process [14]. As shown in Fig. 3.

Interpolation and phase multiplication calculation no data dependencies. Therefore, Parallel computation of multi channel block, multiple threads can be calculated in parallel. As shown in Fig. 4.

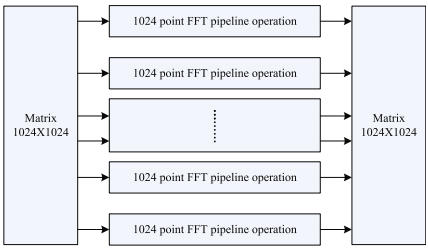

Fig. 3. FFT multi path parallel computing architecture

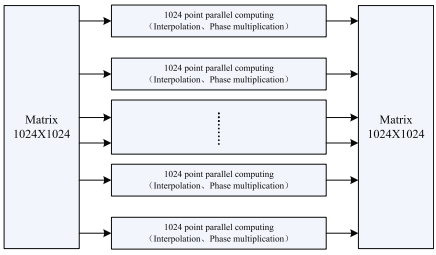

Fig. 4. Interpolation, Phase multiplication parallel computing architecture

4 Experimental Results and Analysis

4.1 Experimental Computing Platform

(1) Hardware Platform:

CPU Model: Intel(R) Core(TM) i5-3230M;
CPU Dominant Frequency: 2.6 GHz;
System Memory: 8.00 GB;
GPU Model: NVIDIA GeForce 840M;
GPU Dominant Frequency: 1.12 GHz;
The Number of SM: 3;
The Number of SP: 3128;
Memory Interface: 64-bit;

Memory Bandwidth: 14.4 GB/s;
Memory Dominant Frequency: 900 MHz;

(2) Software Platform

Operating System: Microsoft Windows 8.1;
Test Platform: Visual Studio 2013;
CUDA v7.5;
Calculation Sample: 1024 × 1024 SAR image data matrix

4.2 Computational Optimization Ratio

See (Table 4).

Table 4. GPU/CPU speedup ratio

Arithmetic unit	CUDA mode	Sine calculation	CPU mode
FFT	7.7 ms	2.665 s	346.1
Interpolation	7.9 ms	1.8 s	227.8
Phase multiplication	15 ms	0.055 s	3.67

4.3 CUDA Parallel Acceleration Bottleneck Analysis

If the data matrix is at one end of GPU, data first entered the memory global, when CUDA starts the CPU computing core. And then, the data is written to memory share, and GPU operations on the data in share memory. The data migration generated a corresponding time, but GPU inside the SM did not appear to wait for the phenomenon of hunger, and the full load of work is always kept. Therefore, the data migration in the chip fails to affect the calculation of the acceleration optimization. However, there are only three SMs in the GPU.

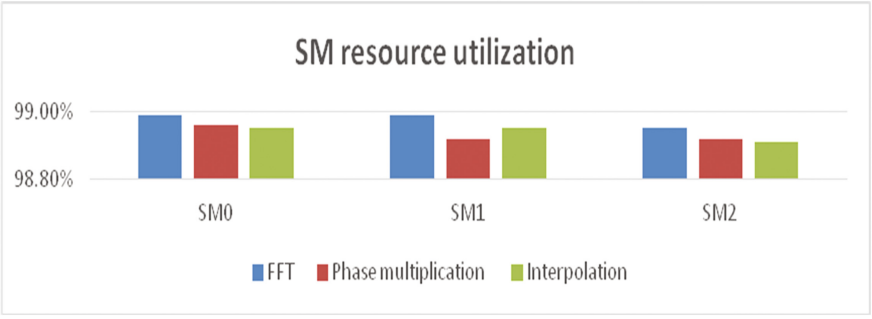

Fig. 5. SM resource utilization

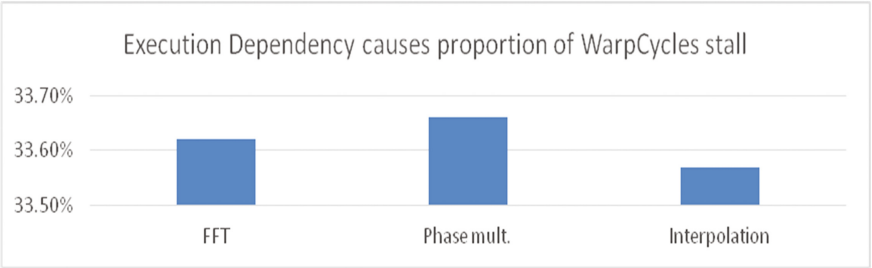

Fig. 6. Execution dependency causes proportion of WarpCycles stall

The internal SM of GPU is in full load working state, which is shown in Fig. 5. The utilization rate of SM has reached 100%, and there is no waiting for data. Therefore, the data migration in the chip does not affect the optimization efficiency of the operation (Fig. 6).

Throughout the clock cycle, more than 80% of the clock cycle is executed in kernel, so there is no reason for warps to block the next instruction, which is shown in Fig. 7.

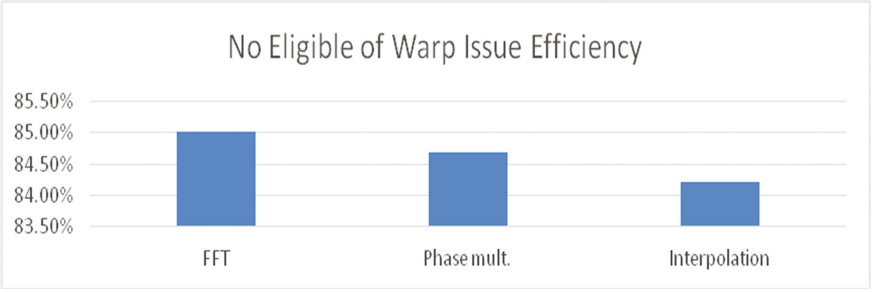

Fig. 7. No eligible of warp issue efficiency

Both the SAR imaging algorithm and parallel computing have been developed towards a mature phase, and plenty of research results have been obtained in these fields [23, 24]. CUDA program contains three kinds of executive unit, namely thread, block and grid. In CUDA programming, a grid is divided into a number of blocks, and then, a block is divided into multiple threads. The division is based on the task characteristics and the hardware characteristics of GPU itself. The division of tasks will also affect the final implementation of the results (Figs. 8 and 9).

In GPU hardware resources, SP (streaming processor) is the most basic processing unit of GPU, and all the operations will be implemented on the SP. Besides, GPU in parallel computing will be ultimately reflected in a number of SP parallel computing. Any SP with some allocation of resources, including storage, memory and register, consists of a SM (streaming multiprocessor). The current GPU contains a limited

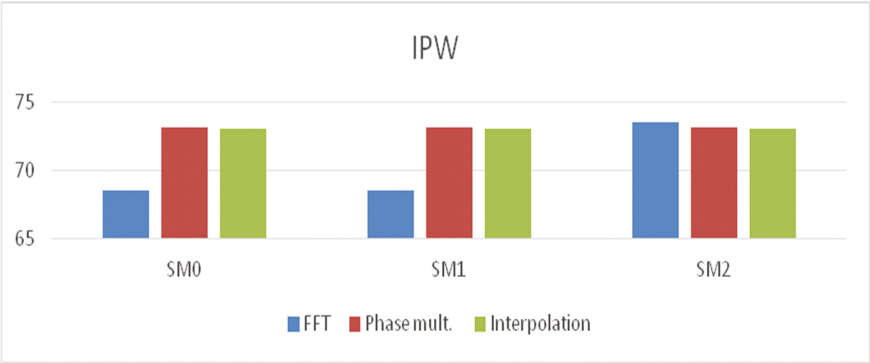

Fig. 8. Instructions per warp

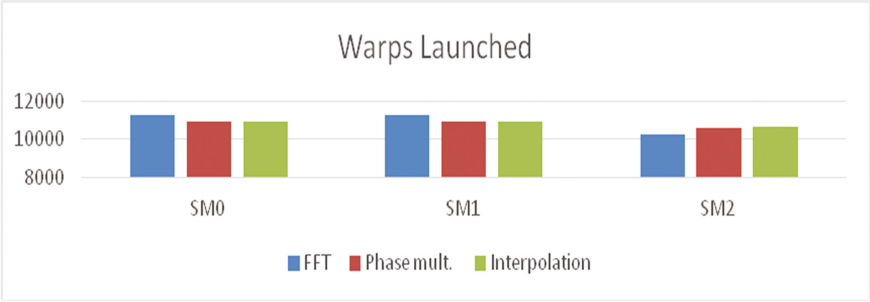

Fig. 9. Warps launched

number of SM and SP. When taking GPU as an example in this experiment, the GPU contains three SMs, each of which contains 128 SPs.

In the process of executing the CUDA program, a SM performs a block task, and at the same time, a SP performs a thread task. However, in the execution program of GPU, the warp is considered as the unit, and a warp is a thread group that contains 32 threads.

In the parallel processing SAR image data matrix and the operation process of three kinds of computing components, the task is divided into 1024 blocks, and the operation of each point is assigned to 1024 threads. When taking the FFT operation of the SAR image matrix as an example, each line of data is assigned to a block which consists of 1024 threads, and each thread performs the FFT operation of the corresponding point in the row data.

The compute-resource allocation process of the parallel algorithm in GPU is analyzed, and at the same time, there will be 1024 blocks, and among them, each block contains 1024 threads to be executed. As above, the parallel algorithm requires 1024 SM processing corresponding to the 1024 blocks, Each SM requires 1024 SPs corresponding to each of the 1024 threads in the block. Only to meet the needs of these

hardware resources, the algorithm can be truly parallel computing. However, from the GPU resource list, SM and SP resources in GPU are relatively limited, and the experiment with the GPU contains 3 SMs, and each SM contains 128 SPs. Therefore, in the implementation of the CUDA program, only three blocks can be executed in parallel at one time. The CUDA program contains 1024 blocks, such as the 3 SMs to be seen as a computing unit, and besides, every 3 block is considered as a unit. If you need to complete 1024 blocks, the 3 SMs sequence needs to be conducted for 342 times. 1024 blocks are actually calculated in a serial manner 342 times. Each block contains 1024 threads, while each SM in the SP executes a thread, but each SM contains 128 SPs. Similarly, with the 128 SPs as a computing unit and 128 threads as a pending computing unit, the 1024 threads need to be calculated in a serial mode eight times. Therefore, due to the limitation of GPU's own hardware resources, the parallel optimization efficiency of the SAR image matrix processing program is of great limitation.

SFU, as a special function of SM can achieve six kinds of transcendental functions, including common sine, cosine, logarithm, exponential, reciprocal and square root. However, when each thread needs to use SFU, each of the four threads requires a serial use of SFU. it seriously affects the parallel optimization efficiency between threads. For example, in the process of interpolation, as shown in Table 5, the sine and the reciprocal operation are involved. In this case, each of the four interpolations of the thread must be done after the serial operation of the sine and the countdown to complete the entire interpolation algorithm. What's more, this process turns the parallel operation between threads into the serial operation between threads, and the advantage of CUDA parallel computing is lost.

Table 5. Interpolation operation time in different degree of parallelism

Degree of parallelism	1-thread	2-threads	4-threads	8-threads	1024-threads
Operation time	6.6 ms	6.7 ms	6.6 ms	6.6 ms	7.7 ms

5 Conclusion

In GPU-based SAR imaging system, application defect induced by resources competition can significantly decrease the whole system performance. To solve this problem, the SAR imaging system is firstly modeled on CPU and GPU in view of the idea that CPU rate can be replaced by GPU, and the speedup ratio of system with GPU-based is also given. Then, the principle of GPU-based that the parallel granularity loss rate is analyzed, and the deceleration ratio of SAR imaging with CUDA is also given. Finally, the bottleneck of GPU-based SAR imaging system is analyzed, and the bound on SAR imaging speedup ratio that GPU-based without increasing power consumption is proposed. The effectiveness of the proposed inference is demonstrated by numerical simulations. It is shown that GPU-based SAR imaging system with resource bottleneck operation rate can not be further improved.

Note that only three key operating cells in multiple SAR imaging algorithms in the paper, however, similar analysis can be made of other operating cell in SAR imaging

algorithm. The scheme proposed in this paper achieve heuristic result has some values of guidance in acceleration research of SAR Imaging.

How to increase the calculating speed in real-time SAR imaging is an open problem. In addition, application on GPU-based has little room for improving performance more. For future work, we plan to deeper investigation of arithmetic unit oriented to application of SAR imaging.

References

1. Larsen, E.S., Mc Allister, D.: Fast matrix multiplies using graphics hardware. In: 2001 ACM/IEEE Conference on Supercomputing (CDROM). ACM Press (2001)
2. General-Purpose Computation Using Graphics Hardware. http://www.gpgpu.org/
3. YANG, C., Wu, Q., Hu, H., Shi, Z., Chen, J., Tang, T.: Fast weighting method for plasma PIC simulation on GPU-accelerated heterogeneous systems. J. Cent. South Univ. **20**, 1527–1535 (2013)
4. NVIDIA CUDA: Compute Unified Device Architecture http://developer.nvidia.com/object/cuda.html
5. Li, Z., Wang, J., Liu, Q.H.: Frequency-domain backprojection algorithm for synthetic aperture radar imaging. IEEE Geosci. Remote Sens. Lett. **12**(4), 905–909 (2015)
6. Sheng, H., Wang, K., Liu, X., Li, J.: A fast raw data simulator for the strip map SAR based on CUDA via GPU. In: IGARSS, pp. 915–918 (2013)
7. Dąbrowski, R., Chodarcewicz, Ł., Kulczyński, T., Niedźwiedź, P., Przedniczek, A., Śmietanka, W.: Accelerating USG image reconstruction with SAR implementation on CUDA. In: Kim, T., Cho, H., Gervasi, O., Yau, S.S. (eds.) FGIT 2012. CCIS, vol. 351, pp. 316–329. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35600-1_47
8. Denham, M., Areta, J., Tinetti, F.G.: Synthetic aperture radar signal processing in parallel using GPGPU. J. Supercomput. **72**(2), 1–17 (2015)
9. Long, H.: Research and implementation of synthetic aperture radar parallel imaging algorithm. Master thesis, University of Electronic Science and Technology (2001)
10. Benson, T.M., Campbell, D.P., Cook, D.A.: Gigapixel spotlight synthetic aperture radar backprojection using clusters of GPUs and CUDA. In: 2012 IEEE Radar Conference (RADAR), pp. 0853–0858. IEEE (2012)
11. Kong, F., Zhao, J., Yue, B.: Research on parallel processing of SAR imaging algorithm. In: Proceedings of the 2nd Asian-Pacific Conference on Synthetic Aperture Radar, pp.784–787 (2009)
12. Swarztrauber, P.N.: FFT algorithms for vector computers. Parallel Comput. **1**, 45–63 (1984)
13. Cooley, J., Tukey, J.: An algorithm for the machine calculation of complex Fourier series. Math. Comput. **19**(90), 297–301 (1965)
14. Ogata, Y., Endo, T., Maruyama, N., Matsuoka, S.: An efficient, model-based CPU-GPU heterogeneous FFT library. In: Proceedings of the 17th International Heterogeneity in Computing Workshop (in conjunction with IPDPS 2008) (2008)
15. Solimene, R., Catapano, I., Gennarelli, G., Cuccaro, A.: SAR imaging algorithms and some unconventional applications: a unified mathematical overview. IEEE Signal Process. Mag. **31**(4), 90–98 (2014)
16. Capozzoli, A., Curcio, C., Liseno, A.: Fast GPU-based interpolation for SAR back projection. Progress Electromagn. Res. **133**, 259–283 (2013)