



Article

Constrained Optimization of FPGA Design for Spaceborne InSAR Processing

Jiahao Li, Ming Xu, Yizhuang Xie * and He Chen

Beijing Key Laboratory of Embedded Real-Time Information Processing Technology, Beijing Institute of Technology, Beijing 100081, China

* Correspondence: xyz551_bit@bit.edu.cn

Abstract: With the development of spaceborne processing technologies, the demand for on-board processing has risen sharply. Against this background, spaceborne Interferometric Synthetic Aperture Radar (InSAR) processing has become an important research area. In many cases, high processing capacity is required during on-board InSAR processing, yet Field-Programmable Gate Array (FPGA) resources on the satellites are limited. To improve the performance of spaceborne remote sensing processing, this paper designs a high-performing FPGA system for the coarse registration and interferogram generation process of InSAR. Moreover, to address this dual-constraint problem of resource and processing capacity, the paper proposes an FPGA design method based on the gradient descent theory, which can identify the optimum trade-off scheme between two such constraints. Finally, the proposed system design and method are implemented in FPGA. Experiments showed that the FPGA system outperformed the NVIDIA (Santa Clara, CA, USA) GTX Titan Black Graphics Processing Unit (GPU), and the optimum trade-off scheme only increases the entire time by 1.1% but reduces the FPGA BRAM usage by 8.7%. The experimental results proved the effectiveness and validity of the proposed system and method.

Keywords: FPGA; InSAR; double nonlinear constraints; gradient descent; spaceborne processing



Citation: Li, J.; Xu, M.; Xie, Y.; Chen, H. Constrained Optimization of FPGA Design for Spaceborne InSAR Processing. *Remote Sens.* **2022**, *14*, 4713. <https://doi.org/10.3390/rs14194713>

Academic Editor: Deodato Tapete

Received: 21 July 2022

Accepted: 16 September 2022

Published: 21 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Synthetic aperture radar (SAR) is an all-weather and all-time system that can obtain high-resolution images. Interferometric Synthetic Aperture Radar (InSAR) is a new interferometry technology introduced into the field of SAR due to its high precision. After obtaining two or more SAR images of the same scene from slightly different angles, InSAR could detect minor topographical changes by processing the interferometric phase of those SAR images. In recent years, InSAR has played a vital role in geological research, including earthquakes [1–3], volcanoes [1–6], landslides [7–9], ground subsidence [10–12], etc. Moreover, with the ability to acquire three-dimensional elevation information, InSAR also demonstrates potential in forest height estimation [13,14], space planetary exploration [1–17], and ocean observation [18–20].

On-board processing, among other research areas of InSAR, attracts much research attention as it can reduce the volume of mission data through satellite–ground communication and improve the proportion of valid data [21]. Owing to the advancement of rad-hardened/tolerant FPGAs, SAR on-board processing has been realized [22], which provides a solid foundation for implementing InSAR. InSAR on-board processing is advantageous in fields with real-time requirements since it could reduce output data rates and downlink volume. For example, it has been employed in many oceanic studies, including ocean topology and surface water topography, which require the processing of a large amount of ocean observation data [23,24]. As for fields that require high accuracy instead of real-time performance, such as planetary missions, on-board InSAR processing is not the ideal choice compared with the ground processing system at the current

stage. Therefore, this paper focuses on applying InSAR on-board processing technology in oceanic observation.

Existing research on InSAR processing can be loosely grouped into two categories: the first type looks at the optimization of the InSAR algorithm under specific scenarios from theoretical perspectives [25–28]; the second type focuses on the implementation and optimization of the InSAR engineering system [29–34]. This paper belongs to the second type and works for the betterment of InSAR engineering.

As the processing speed of the Central Processing Unit (CPU) is relatively low, GPU and Digital Signal Processor (DSP) are used as the mainstream processors in InSAR engineering. Since 2015, GPU has been applied to InSAR processing. The processing time of GPU is only one-tenth that of CPU, demonstrating its fast-processing capacity [29–33]. Some scholars implement the InSAR algorithm with DSP as the central processor. In 2016, Wang et al. [34] built an InSAR system based on TMS320C6678 DSP. However, the above research was conducted with in-ground or airborne scenarios with no power constraints and low throughput. Due to the strict conditions on satellites, the volume power consumption is limited, and the image data volume is large, so it is challenging to meet the requirements of spaceborne real-time processing using DSP or GPU.

Compared with DSP, FPGAs have the advantage of high processing capacity. Compared with GPU, FPGAs consume lower power consumption. Thus, FPGA has excellent potential in the field of spaceborne real-time processing. In addition, with the development of FPGA technologies, especially the advancement of rad-hardened FPGAs, such as Xilinx (San Jose, CA, USA) Virtex-5QV and Zynq UltraScale+, on-board SAR processing has become a reality [21,35], which prepares conditions for InSAR on-board processing. Yet, as on-board InSAR processing based on FPGA is a relatively new research area, this paper attempts to conduct exploratory research.

Many technical challenges need to be addressed to build an FPGA-based on-board InSAR processing system, including the matrix transpose structure [36] and the trade-off between processing capacity and resources.

The matrix transpose module realizes the mapping to the Double-Data-Rate Three Synchronous Dynamic Random-Access Memory (DDR3 SDRAM) from the partitioned matrix, including two-dimensional mapping [36–40] and three-dimensional mapping [41–44]. With the increasing complexity of spaceborne processing functions, the software–hardware co-design method has been adopted to shorten the development and maintenance cycle of the System on Programmable Chip (SoPC). The SoPC system is based on the Advanced eXtensible Interface (AXI). However, the existing matrix transposition scheme is unsuitable for the system under the condition of AXI for the following reasons. Firstly, most of the existing transposition schemes use native interfaces, and the address space of the native interface differs from that of the AXI. To combine the two modes, additional steps are needed to change the address between the two modes, thus occupying more FPGA resources. Secondly, the existing method of matrix transposition of the native interface improves the storing efficiency by analyzing the difference in storage efficiency of data in the row, column, and bank, which requires a three-dimensional storage space. However, the storage space of the AXI is performed as a two-dimensional storage space, which is not compatible with the three-dimensional mapping structure of the native interface. Therefore, to address the compatibility issue, it is necessary to study the matrix transpose structure under AXI conditions.

In on-board processing, given the limited resources and high-performance requirements, the FPGA design is faced with the contradiction between processing capacity and resource usage. Researchers have developed many FPGA design techniques to improve processing capacity and reduce resource consumption, which could be classified into three categories.

First, utilize FPGA resources fully. Fu et al. [45] implemented two int8 multiplications with one DSP48E2 in a new module. The DSP resources were fully used to improve the system processing capacity. However, although such a method can increase processing

capacity, the field of application is limited. For example, it cannot work in high-bit data widths, such as 32 bit.

Second, integrate heterogeneous processors perfectly. To take advantage of different processors, some researchers [43,46,47] moved the operations that consume many FPGA resources to the DSP so that the FPGA resources are significantly saved with processing capacity improved. However, this method increases the complexity of the system. The scale of the system increases the difficulty of control. Moreover, the communication rate between heterogeneous processors probably hampers the system processing capacity.

Third, modify algorithm flow correspondently, such as quantization and compression. Many researchers [48,49] used the quantization method to reduce the data bit width and the resources by converting the floating-point data into low-bit data. By pruning algorithm steps, Han et al. [50] used the compression method to reduce the overall computation, thus reducing the resources of FPGA. This method can fundamentally lessen the computation in the FPGA. However, it may lead to the loss of algorithm accuracy and not directly address the issue between resources and processing capacity.

In summary, the existing technical solutions do not directly address the contradiction between processing capacity and resources. Little research has explored the trade-off relationship between processing capacity and resources based on the FPGA system without modifying the algorithm and structure. Therefore, faced with the dual constraints between resources and processing capacity, this paper proposes an FPGA design method based on the gradient descent theory. Unlike the existing technical solutions, this method works with FPGA as the only processor without designing a new module or modifying the algorithm flow. This method solves the dual-constraint problem by balancing the relationship among the modules to find the optimum trade-off between resources and processing capacity.

Focusing on the dual-constraint problem in on-board processing, we designed an efficient FPGA system for the InSAR coarse registration and interferogram generation process. To improve the processing capacity, we studied the data reading and writing rules of matrix transposition based on AXI conditions. Moreover, to balance the relationship between resources and processing capacity, we proposed an FPGA design method based on gradient descent theory, which could help identify the optimum trade-off scheme. Finally, the structure and method of this paper were successfully implemented on FPGA. The contributions of this paper are summarized as follows:

- To realize on-orbit InSAR processing, we design an efficient FPGA system for coarse registration and interferogram generation flow of InSAR.
- To perform matrix transposition under the AXI protocol, we analyze the data rule of matrix transposition based on the AXI protocol.
- To solve the dual-constraint problem of processing capacity and resources, we propose a design method of FPGA based on the gradient descent theory. We use the InSAR design in this paper to illustrate how to find the optimum trade-off between processing capacity and resources.
- We implement the efficient structure and the design method on FPGA. The results show that the efficient structure outperforms the NVIDIA GTX Titan Black GPU. The design method can not only guide the engineering design but also evaluate the optimized structure academically.

The remainder of this article is organized as follows. Section 2 introduces the targeted InSAR algorithm, including coarse registration and interferogram generation, and briefly introduces the gradient descent theory. Section 3 describes the architecture, processing, and storage structure of the FPGA system proposed by this paper. Notably, the matrix transpose storage structure can be adapted to the SoPC system under the AXI conditions, ensuring processing efficiency. Section 4 illustrates how to seek the balance between resources and processing capacity of FPGA using the gradient descent theory. In Section 5, this paper shows experimental results measuring the performance of the FPGA system and compares them with that of GPU and DSP. Results show that the balanced design of the FPGA proposed by this paper could achieve high processing performance while

saving resources. With the design proposed by this paper, FPGA demonstrates comparable processing capacity to that of NVIDIA GTX Titan Black GPU, revealing the great potential of FPGA in the field of spaceborne real-time processing. The paper ends with conclusions in Section 6.

2. Theoretical Background

2.1. Coarse Registration and Interferogram Generation of InSAR

The InSAR processing flow includes 6 steps [51]: image registration, interferogram generation, interferogram flattening, interferogram phase filtering, interferogram phase unwrapping, phase to height conversion and geocoding. Existing research shows that the interferogram generation process is the most time-consuming step, occupying about 32% of the time in the process of InSAR (without fine registration) [34]. Therefore, this paper focused on the first two steps of InSAR processing: image registration and interferogram generation, to shorten the processing time of InSAR.

Before calculating the interference phase and coherence coefficient, the two images need to be registered to obtain accurate phase information. Image registration can be divided into coarse and fine registration, according to accuracy. Pixel-level registration is called coarse registration, while sub-pixel registration (i.e., less than one pixel) is fine registration.

The specific case used in this paper was the observation of the variation of ocean depth in the vast ocean scene through satellite altimetry. Since this paper did not focus on any particular area, fine registration resampling applied in GPU [52,53] did not fit our needs. Thus, a coarse registration algorithm, based on the correlation function, was adopted, as shown in Formula (1). $M_1(i,j)$ and $M_2(i,j)$ are the amplitude of the two complex images, and $R(u,v)$ is the correlation function of the two images. The correlation process could be transformed from the time domain into the frequency domain and quickly calculated by Fast Fourier Transform (FFT), as Formula (2) shows, which also suits pipeline operation in FPGA. $M_1(i,j)$ and $M_2(i,j)$ in the time domain correspond to $S_1(m,n)$ and $S_2(m,n)$ in the frequency domain, and R_{FFT} is the frequency domain result of the correlation function $R(u,v)$. The symbol * denotes the conjugate operation:

$$R(u,v) = \frac{\sum_i \sum_j M_1(i,j) M_2(i+u, j+v)}{\sqrt{\sum_i \sum_j M_1^2(i,j)} \cdot \sqrt{\sum_i \sum_j M_2^2(i+u, j+v)}} \quad (1)$$

$$R_{FFT} = S_1(m,n) \cdot S_2^*(m,n) \quad (2)$$

Cross-correlation results of the two images could be obtained by performing Inverse Fast Fourier Transform (IFFT) on the R_{FFT} . The offset coordinate (x -axis, y -axis) of registration could be calculated using the position of the maximum value in cross-correlation results.

What followed was interferogram generation. After the two images were aligned, according to the offset coordinates, the phase and coherence coefficient could be calculated using Formula (3):

$$\gamma = \frac{E\{M_1 \cdot M_2^*\}}{\sqrt{E\{|M_1|^2\}} \cdot \sqrt{E\{|M_2|^2\}}} = |\gamma| e^{j\theta} \quad (3)$$

In this formula, M_1 and M_2 represent two complex images; $E\{\cdot\}$ means the mathematical expectation; * is the conjugate; γ is the complex coherence, and $|\gamma|$ is the amplitude of the complex coherence; θ is the phase.

The FPGA calculation flow of coarse registration and interferogram generation is shown in Figure 1. Firstly, the sub-image (i.e., 512×512) was from the center of the original image (i.e., $16,384 \times 4096$). Secondly, the two sub-images were sent to the two-dimensional FFT, including range FFT, matrix transposition, and azimuth FFT. Thirdly, after the two-dimensional FFT and IFFT, the offset was obtained from the maximum value. Finally, the

two images' coherence coefficient and interference phase could be calculated, according to the offset coordinate (x -axis, y -axis).

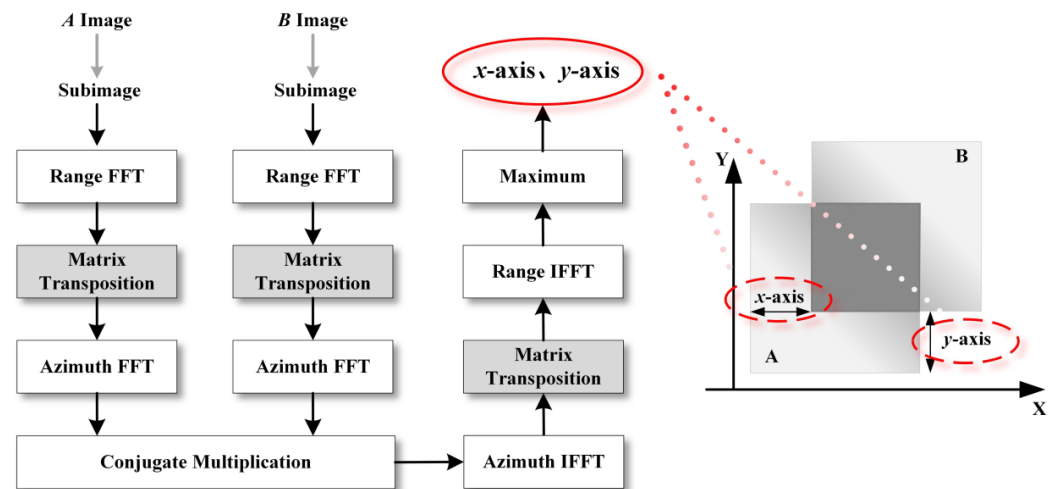


Figure 1. FPGA calculation flow of the algorithm. “FFT” refers to “Fast Fourier Transform”. “IFFT” represents “Inverse Fast Fourier Transform”.

The operation of matrix transposition is essential in the process of two-dimensional FFT. The step of matrix transposition is highlighted in grey to illustrate that a specialized FPGA structure was required.

2.2. Gradient Descent Method

In on-board processing, the design of an FPGA system is faced with various strict nonlinear constraints, such as resources, processing capacity, power consumption, volume, weight, etc. Therefore, it was essential to research an FPGA system's design under multiple constraints.

Mathematical optimization theory can shed light on solving nonlinear multi-constrained problems. These can be expressed by the objective function and constraints, as shown in Formula (4):

$$\begin{aligned} \text{Objective functions : } & \quad \text{MIN } F(X_1, X_2, \dots, X_n) \\ \text{Constraints} & \quad \begin{aligned} & g_i(X_1, X_2, \dots, X_n) \geq 0 \quad i = 1, 2, \dots, m \\ & X_j \geq 0 \quad j = 1, 2, \dots, n \end{aligned} \end{aligned} \quad (4)$$

If the objective function F can be expressed as a formula, then the nonlinear optimization problem could be solved mathematically by obtaining the optimum value X_j . Since there are many types of FPGA logic resources and the resource values are discrete, neither the objective function nor constraints can be expressed as a function, thus making it very difficult to apply the optimization theory to solve the problem directly.

That said, we introduced the gradient descent method in optimization theory into the design of the FPGA system to solve the dual-constraint problem between resources and processing capacity. The basic principles of the gradient descent method are described below.

The gradient descent method was proposed by Cauchy in 1847 and has become the basis of many constrained and unconstrained optimization methods [54]. The gradient descent method sets the negative gradient direction of the objective function as the searching direction of each iteration so that the value of the objective function decreases the fastest in the first few steps. For a nonlinear function f with a first-order continuous partial derivative, X contains n variables x_n , and the gradient is defined by Formula (5):

$$\nabla f(X) = \frac{\partial f}{\partial X} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right] \quad (5)$$

We used the gradient descent method to map each FPGA scheme's resources and processing capacities in two-dimensional coordinates. Then we analyzed the relationship between resources and processing capacity through the gradient between the schemes. After that, we identified the balanced scheme. The detailed steps are described in Section 5.

3. Design of System and Structure

This paper built a SoPC system based on the AXI to achieve high efficiency in communicating, storing, and processing, which is described in detail in the following sections.

3.1. System Architecture and Data Communicating

As the on-board processing algorithm becomes increasingly complex, the debugging cycle of traditional HDL finite state machines becomes longer, which increases the cost of maintenance. As an alternative, the solution SoPC has been widely used. As shown in Figure 2, the SoPC system has four parts: controlling, computing, arbitrating, and storing.

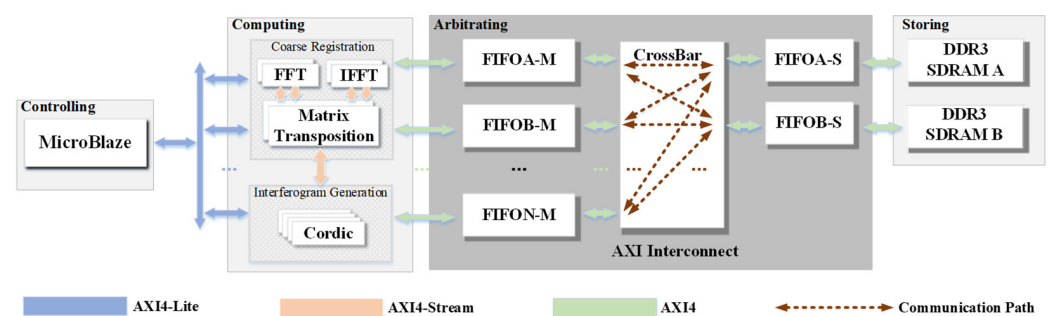


Figure 2. The architecture of the system. In “FIFOA-M” and “FIFOA-S”, “FIFO” means “First In First Out”, “N” is the number of FIFO, “M” and “S” denotes Master and Slave respectively. “DDR” represents “Double Data Rate”.

For the controlling part, compared with the traditional HDL finite state machine, the Microblaze embedded processor soft core replaces the conventional finite state machine, software programming supersedes hardware programming, and software testing replaces hardware debugging. For a SoPC system, control signals are exchanged through AXI4-Lite, and data is processed through AXI4-Stream and visited by the storage space through the AXI4 [55].

The computing part realizes InSAR signal processing, containing the major operation units FFT and Cordic. FFT is used to transform data between the time domain and frequency domain, while Cordic is used to calculate the coherence and phase. The Matrix Transposition module is designed to improve the efficiency of accessing range and azimuth data and increase processing bandwidth.

The arbitration part based on AXI Interconnect consists of CrossBar, which can arbitrate reading/writing data operations between the FIFOs on both sides. The arbitration structure solves the problem caused by the algorithm function and the dual-channel storage structure. In the process of interferogram generation, either chip of DDR3 SDRAM includes both reading and writing operations simultaneously, which cannot be achieved without the arbitration structure. Therefore, CrossBar implements arbitration of reading/writing operations through the Shared-Address Multiple-Data (SAMD) topology to realize equivalent pipeline processing [56].

3.2. Hardware Structure of Graduated Registration

Before the interferogram generation, the images must be aligned in the range and azimuth directions. To improve the processing capacity, this paper proposed a specialized structure based on FPGA called “graduated registration.” It fits the FPGA pipeline, thus improving the processing capacity and reducing the algorithm’s complexity. The analysis is as follows.

The offset coordinates (x -axis, y -axis) are obtained when the registration process is completed. Before entering the interferogram generation process, the two images must be registered by the offset coordinates. The coordinate offset reflects the relative position of the main and auxiliary images. The offset coordinates (x -axis, y -axis) can be positive or negative; the auxiliary image can be moved either positively or negatively along the X or y -axis. x -axis > 0 means that Image A moves in the positive X direction relative to Image B; similarly, y -axis > 0 means Image A moves in the positive Y direction relative to Image B. Therefore, there are four cases of offset in the four quadrants, as shown in Figure 3.

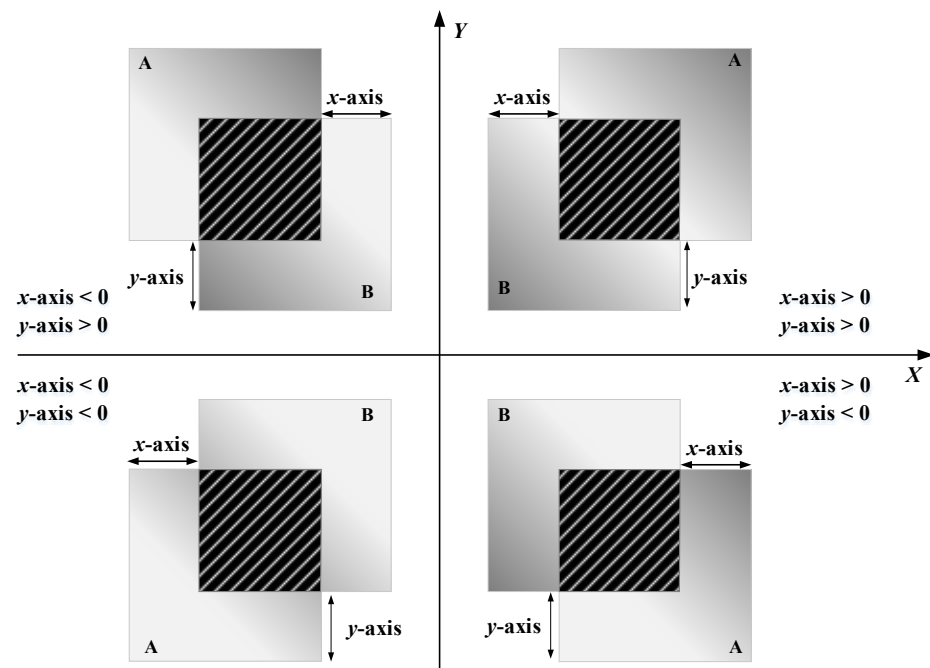


Figure 3. The relative position of the images.

If the offset of X and Y works in the DDR3 SDRAM, like 2D_Address, shown in Figure 4, the algorithm's complexity is $O(n^2)$ because the offset coordinate works in two loops. More importantly, when filtering data in two-dimensional space, the corresponding DDR3 SDRAM addresses are discontinuous, which affects storage efficiency terribly. To improve efficiency and fully take advantage of the pipeline processing capacity of FPGA, we designed a structure called graduated registration, as shown in Figure 4. The x -axis was moved from the DDR3 SDRAM into the FIFO, and only the y -axis worked in the DDR3 SDRAM. In this way, the algorithm's complexity was reduced to $O(n)$ without the loss of efficiency caused by discontinuous addresses, like 1D_Address.

The data flow works as follows. First, the initial address of azimuth data is obtained according to y -axis. Then, a particular row of the original data is read from the DDR3 SDRAM. Finally, two lines of data from two DDR3 SDRAM chips are buffered in S1_PRE and S2_PRE to filter range data, thus realizing the registration process of two-dimensional data. As the offset, y -axis and x -axis work in DDR3 SDRAM and FIFO, respectively, to perform two-dimensional registration. We called the proposed structure "graduated registration".

According to the x -axis, both the S1_PRE and S2_PRE modules have two cases, as shown in Figure 4. It can be seen that data in the modules could be divided into three segments: the C segment was for the coherence coefficient and phase, the A segment was discarded, and the B segment was zeroed. In this structure, the unavailable data was zeroed or discarded, so the depth of FIFO stayed unchanged (i.e., 4096).

The graduated registration structure proposed in this paper has two advantages. On the one hand, it ensures a high data bandwidth reading from DDR3 SDRAM with lower algorithm complexity and efficiency loss. On the other hand, the length of data buffered in this structure is constant by filling zeros, which can easily be controlled and debugged.

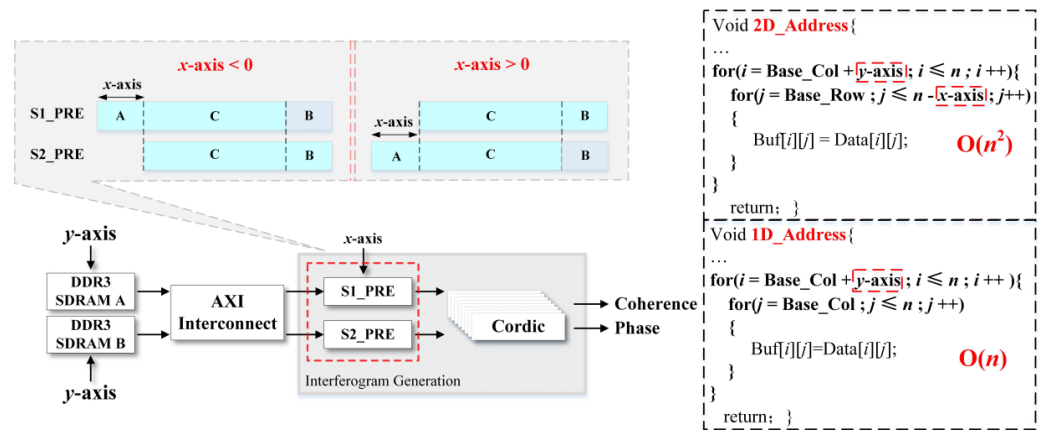


Figure 4. The graduated registration. “Base_Row” represents the initial row address in DDR. “Base_Col” means the initial column address in DDR. Both “Base_Row” and “Base_Col” are base addresses. “y-axis” and “x-axis” denote the offset addresses.

3.3. Matrix Transposition Based on AXI

Matrix transposition is an essential step in the 2D FFT calculation of coarse registration. The efficiency of matrix transposition significantly affects the total time of coarse registration. The burst length of the native interface in DDR3 SDRAM is 8 [36], yet the burst length of the AXI can reach 1024 data. The schemes [36–44] of existing research on matrix transposition are built on the native interface, which could not be used in the SoPC system directly. However, the existing matrix transposition schemes could shed light on the research on the AXI matrix transposition. Inspired by a previous study [37], we found out that the key to efficient matrix transposition data access lay in the balance of two-dimensional data: the number of range and azimuth data were the same. Therefore, the azimuth and range data in the design structure occupied half of the continuous data. The matrix block structure, including the reading and writing process, is shown in Figure 5.

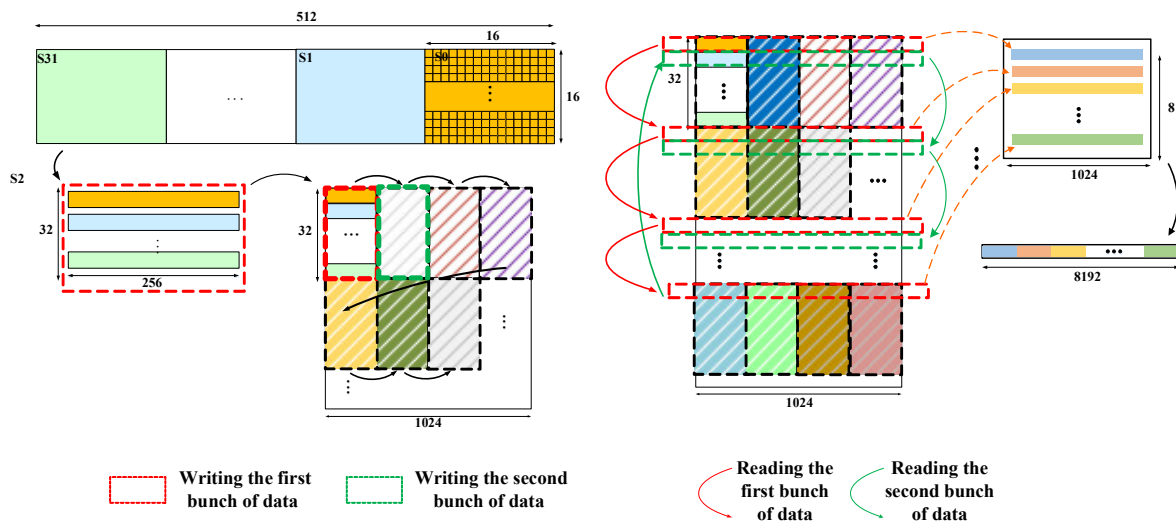


Figure 5. Reading and writing process of matrix transposition.

We took the transfer of 256 data of 64 bit at a time as an example to illustrate the reading and writing process. For the 512×512 -point data to be registered, 16 lines of data were buffered. After that, we rearranged the data to take 16 range data in each row, then put the 256 data from 16 rows into a line. Then, the new array containing 32 lines was placed into DDR3 SDRAM in the direction of the arrows in Figure 5, until the writing process was completed. When data was read from the storage space, data was read from

the entire row of the DDR3 SDRAM according to different row addresses to form an array. The array corresponded to 16 columns that were the transposed data.

When the burst length of AXI changed, the efficiency would be different, and the matrix transposition structure would be different. Among the achievable solutions, the burst length could be selected as 1024, 256, 64, and 16; correspondingly, 32, 16, 8, and 4 pieces of data needed to be buffered and transposed. It could be seen that as the burst length increased, the efficiency gradually increased, and the corresponding storage resources increased. In this paper, the coarse registration dealt with 512×512 floating-point complex images. If the most efficient transfer was achieved, the burst length was 1024, with 32 lines of image data buffered in the RAM. Considering both the reading and writing process, the storage space was at least $2 \times 512 \times 32 \times 64 \text{ bit} = 2 \text{ Mb}$.

Under the condition that the image size remained unchanged, the higher transmission efficiency led to more BRAM resources being occupied correspondingly. This also reflected the contradiction between the processing capacity and resources in the FPGA design. The limited amounts of resources would inevitably limit the processing capacity. Faced with this conflict, it was necessary to make a balance and trade-off from the system design perspective so that the used resources could maximize the processing performance. To analyze and solve this problem, the following section examines the two-dimensional constraints of processing capacity and resources from a systematic perspective.

4. System Analysis under Double Constraints

4.1. Design Method Based on Gradient Descent Theory

In FPGA design, the high processing capacity can be achieved by increasing resources with the parallel structure. Yet, there are conflicts between resources and processing capacity. Therefore, it is necessary to solve the following optimization problem: when faced with different constraints, including requirement and environment restrictions, how can one build a balanced system that can meet the requirements of quick data processing and reduce resource usage to a great extent. Balance in this paper primarily referred to the trade-off between resources and processing capacity. The strength of balance is reflected by the quantitative relationship between resources and processing capacity. The dual-constraint problem is described below by analyzing the relationship between processing time and resources.

The relationship between processing capacity and resources is nonlinear—resources are allocated to different processes, including computing, controlling, communicating, etc., which means that only part of the resources could be used to improve processing capacity. According to the optimization model, as described by Formula (4), the relationship between resources and time could be described similarly as in Formula (6):

$$\begin{array}{ll}
 \text{Objective functions :} & \begin{array}{l} \min F(R_j, T_j) \\ g_i(R_j, T_j) \geq 0 \end{array} \\
 \text{Constraints} & \begin{array}{l} R_j \leq R_0 \quad i = 1, 2, \dots, m \\ T_j \leq T_0 \quad j = 1, 2, \dots, n \end{array}
 \end{array} \quad (6)$$

For structure j , R_j represents the used resources, and T_j represents the consumed time. In other words, different structures contain different R_j and T_j . The constraint of resources R_0 represents the total amount of resources, and the constraint of time T_0 comes from application requirements.

In engineering, it is difficult to illustrate the objective function in specific formulae. Therefore, we could not directly express it in mathematical formulae like Formula (5). However, the idea of gradient descent from optimization theory can be brought into system design. Supposing there are n schemes in FPGA design, the consumed time and critical resources are $T_1, R_1, T_2, R_2, \dots, T_n, R_n$, etc. R_0 is the resource in FPGA and T_0 is the system processing time. The FPGA schemes could be mapped into a two-dimensional space of R and T , where discrete points represent the schemes. The gradient between the points

could be defined as ∇G , expressed as Formula (7). ΔR represents the absolute difference in resources between schemes; ΔT represents the absolute difference in time; ∇G represents the difference in system effect between resources and processing capacity:

$$\nabla G = \frac{\Delta R/R_0}{\Delta T/T_0} \quad (7)$$

We defined γ as $\gamma = \Delta R/R_0$, which meant the impact of changing partial resources on the total resources. Similarly, we defined β as $\beta = \Delta T/T_0$, which represented the effect of the changing partial time on the entire system time. Since the FPGA contained various resources, the total amount of resource R_0 could represent different types of resources. However, the total system time T_0 was not a constant and would vary according to the different design.

To realize fast processing, the analysis process of the proposed method in the two-dimensional R - T space was as follows:

- (1) Given that the physical constraints and time constraints are met (i.e., $R_j \leq R_0$, $T_j \leq T_0$), m initial points (R_j, T_j) are obtained through synthesis and estimation, $1 \leq j \leq n$ and $j \in Z$.
- (2) Setting the point $T_k = \text{MIN}(T_j)$ ($1 \leq k \leq n$ and $k \in Z$) as the initial state and setting the T_j ($1 \leq j \leq n$, $j \in Z$ and $j \neq k$) as the next state, then ∇G between (R_k, T_k) and (R_j, T_j) is calculated.
- (3) Finding the maximum value of ∇G through the following operation: $\nabla G(k, l) = \text{MAX}\{\nabla G(k, j)\}$ ($1 \leq j \leq n$, $j \in Z$ and $j \neq k$).
- (4) When $\exists l \in Z$, $1 \leq l \leq n$, $l \neq k$ and $\nabla G(k, l) \geq G_{\text{gate}}$, (R_j, T_j) is defined as the balanced scheme. Otherwise, there is no balanced scheme.
- (5) If there is a balanced scheme in step (4), the point (R_k, T_k) is deleted to form a new feasible zone. The value of T_l replaces the original value of T_l (i.e., $T_k = T_l$), and steps (2), (3), and (4) are repeated until there is no balanced scheme.

G_{gate} is the threshold set by developers according to the requirement of the system. G_{gate} was set as 3 in this paper to achieve high system performance. The following section presents a detailed analysis of the InSAR system based on what we have proposed above.

4.2. Analysis of Threads in Interferogram Generation

This part analyzes the relationship between resources and processing time in interferogram generation. According to the algorithm introduced in Section 3, $16,384 \times 4096$ image points were to be calculated in the process of interferogram generation, and 512×512 points were to be calculated in the coarse registration process; thus, the former required much more time than the latter process. Therefore, the time interferogram generation took up was critical to the total system runtime.

If 64-bit float complex data is regarded as a thread, the number of threads could represent the degree of parallelism. Faced with the constraint of the 512-bit data width for one DDR3 SDRAM chip, the thread can be set as 1, 2, or 4. Here, we designed an IP in 4 threads to find the critical resource. The synthesis results of the IP are shown in Table 1. Due to nonlinear operations in the calculation process, LUT took up the most significant proportion among other resources, so LUT was the critical resource.

Table 1. Vivado synthesis results of the IP.

Logic (Amount)	LUT (433,200)	FF (866,400)	BRAM_36K (1470)	DSP (3600)
Results (percentage)	41,723 (9.63%)	25,506 (2.94%)	7 (0.48%)	112 (3.11%)

To increase the proportion of the coarse registration on the system processing time, the thread of coarse registration was set as 1. When the system clock worked at 100 MHz,

512 × 512 points were calculated in 262,144 clocks, 2.62 ms in theory. Therefore, the ideal coarse registration time T_c was 7.86 ms. Similarly, to calculate 16,384 × 4096 points, the ideal interferogram generation time T_g was 671.09 ms. When the thread of interferogram generation changed from 1 to 4, the results are shown in Table 2. T_i was the entire calculation time $T_i = T_c + T_g$.

Table 2. Resource usage and theoretical calculation time in interferogram generation with different threads.

Scheme	Thread	R_i	R_{LUT}	R_i/R_{LUT} (%)	T_c (ms)	T_g (ms)	T_i (ms)
<i>a</i>	1	10,477	433,200	2.42	7.86	671.09	678.95
<i>b</i>	2	20,914	433,200	4.83	7.86	335.54	343.40
<i>c</i>	4	41,723	433,200	9.63	7.86	167.77	175.63

The variation ΔR_{LUT} was obtained by subtraction between the schemes R_i , as shown in Table 3. When threads changed from 1 to 4 ∇G was less than 0.1, which indicated that relatively few resources were saved while the processing capacity was lowered sharply. As ∇G was too low to meet the requirement $\nabla G \geq \nabla G_{gate} = 3$, there was no balanced scheme. As for Δba (i.e., from scheme *b* to plan *a*), the resources were saved by 2.41%, but the time increased by 97.7%.

Table 3. ∇G of schemes in interferogram generation with different threads.

Scheme	ΔR_{LUT}	$\Delta R_{LUT}/R_{LUT}$ (%)	ΔT (ms)	T_i (ms)	$\Delta T/T_i$ (%)	∇G
Δcb	20,809	4.80	167.77	175.63	95.52	0.0503
Δca	31,246	7.21	503.32	175.63	286.57	0.0252
Δba	10,437	2.41	335.54	343.40	97.71	0.0247

When the thread was 4, 256-bit data width, two images were read from the two DDRs, respectively. The calculation results, 128-bit phase and 256-bit coherence coefficients were written into two DDR3 SDRAM chips. Since the data width of either DDR3 SDRAM channel was no more than 512 bits, there was no efficiency loss when data passed through the arbitrating structure. However, when the thread was 8, either channel reading or writing worked in 512 bits, which led to efficiency loss. Therefore, due to the hardware constraints, it was hard to analyze the performance of the 8-thread scheme.

By analyzing schemes with different threads, low ∇G (i.e., lower than 0.5) indicated that the greater processing capacity could be improved with fewer resources. As the thread of FPGA increased, the processing capacity increased accordingly.

4.3. Analysis of Threads in Coarse Registration

This part discusses the relationship between the thread and resources in coarse registration. Coarse registration requires two-dimensional FFT and IFFT operations. The FFT IP core that Xilinx provided consumed 48 DSPs. A total of 6 FFTs were needed to finish parallel processing. As the thread increased, the usage of FFT increased by double. Due to the limitation of the dual-channel DDR3 storage structure and the data width of 512 bits, the maximum thread of the coarse registration was 4. Similarly, time could be calculated according to the working clock. Resources and time are shown in Table 4.

Table 4. Theoretical resource usage and calculation time with different threads in coarse registration.

Scheme	Thread	R_i	R_{DSP}	R_i/R_{DSP} (%)	T_c (ms)	T_g (ms)	T_i (ms)
<i>d</i>	1	288	3600	8	2.62	167.77	175.63
<i>e</i>	2	576	3600	16	1.31	167.77	171.70
<i>f</i>	4	1152	3600	32	0.66	167.77	169.74

When the thread of interferogram generation was 4 (scheme *c*), the ideal time T_g was 167.77 ms. When the thread was 2 (scheme *b*), the T_g was 343.40 ms. As shown in Figure 6,

Δc_{xx} and Δb_{xx} in the figure represented the results when the threads of interferogram generation were 4 and 2, respectively. ∇G of Δc_{fe} was over 12, which meant that many resources were used, but the time improvement effect was very small. So, it could be seen that scheme *e* was more balanced than *f*. If scheme *e* was used as the initial state to analyze, Δc_{ed} was greater than 3, which showed that scheme *d* was more balanced than scheme *e*. In other words, when the thread of interferogram generation was 4, the scheme with 1 thread in coarse registration was the balanced design.

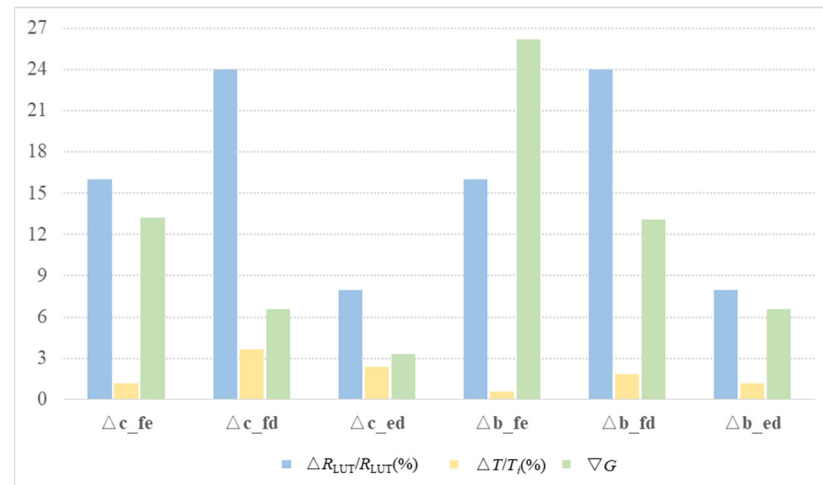


Figure 6. The theoretical analysis of coarse registration with different threads.

Similarly, under scheme *b*, ∇G was nearly doubled since the overall system processing time was almost doubled. It could be seen from Δb_{fd} that reducing the use of resources by 24% only deteriorated the system processing time by less than 2%. So, as the processing time of the system increased, the scheme with 1 thread in coarse registration (scheme *d*) was still the balanced design among other schemes.

4.4. Analysis of the Matrix Transposition Schemes

This part focuses on the matrix transposition structure when the thread of interferogram generation was 4 and the thread of coarse registration was 1. The transpose of the matrix with different burst lengths had different efficiencies of AXI η . The utilization of resources and efficiency of each scheme are shown in Table 5.

Table 5. Theoretical resource usage and calculation time of different matrix transposition schemes.

Scheme	Length	η	R_i	R_{BRAM_36K}	$R_i/R_{BRAM_36K}(\%)$	T_c (ms)	T_g (ms)	T_i (ms)
<i>g</i>	1024	90%	260	1470	17.69	8.7	167.77	176.47
<i>h</i>	256	75%	132	1470	8.98	10.5	167.77	178.27
<i>i</i>	64	50%	68	1470	4.63	15.6	167.77	183.37
<i>j</i>	16	20%	36	1470	2.45	39.3	167.77	207.07

The utilization of resources comes from the synthesis in Vivado, not HLS. Since matrix transposition structure could be designed separately, the resources R_i could be found after synthesis in Vivado. The structure efficiency could be tested on the FPGA board. Then, coarse registration processing time T_c could be calculated.

Similarly, the ∇G calculation results are shown in Figure 7. ∇G of Δgh was 8.16, which meant that, compared with scheme *g*, the time of scheme *h* deteriorated by 1.0%, but the resources were saved by 8.7%. However, since the values of Δhi and Δhj were much less than 3, the scheme *h* was more balanced among the four above schemes. There would be efficiency loss during the storage process, so T_g was above 167.77 ms. Then $\Delta T/T_i$ would be reduced, and the value of ∇G would be increased, again confirming that scheme *h*

was the balanced design. To sum up, when the thread of interferogram generation was 4 and the thread of coarse registration was 1, the scheme with a burst length of 256 was the balanced design.

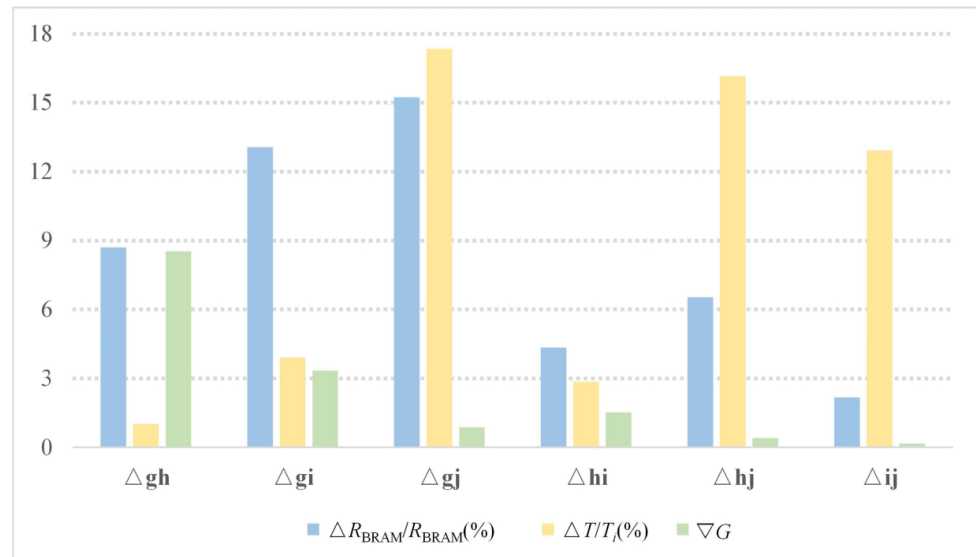


Figure 7. ∇G of matrix transposition schemes under the condition of c' .

Similarly, when the thread of interferogram generation was 2, the system processing time would be prolonged and scheme h, with 256 burst length, would still be the balanced design. The scheme g with 1024 burst length would exert less impact on the whole system while consuming more resources.

5. Experimental Results and Discussion

The structure and analysis designed in this paper were tested on the experimental platform, as shown in Figure 8. The FPGA chip from Xilinx xc7vx690tffg1761 worked at 100 MHz clock frequency, and the DDR3 from Micron (Boise, ID, USA) MT41K512M16HA-125 worked at 200 MHz. The development tools were Vivado2017.4 and HLS2017.4. Both SAR images used in this paper for testing had $16,384 \times 4096$ points with floating-point complex numbers.

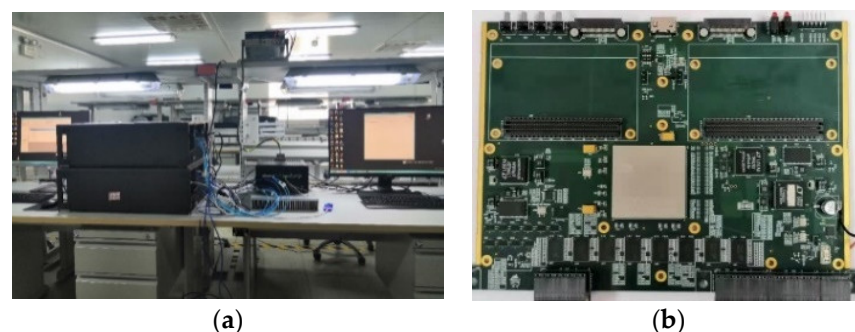


Figure 8. Experimental environment and platform. (a) Ground verification system; (b) Experiment platform.

The working course of our experimental platform was designed as follows. First, two SAR images were sent from the ground verification system through the TLK2711 interface and written in the two DDR3 SDRAM chips of FPGA, respectively. Then, algorithmic processing, i.e., coarse registration and interferogram generation, started until all calculation results were saved. Then, the calculation results were sent to the ground verification system and verified with the MATLAB calculation results. The IP core AXI-Timer provided by Xilinx recorded the time used during algorithmic processing. The verification in this section

was divided into two parts. Sections 5.1–5.3 verified the validity of the analysis in Section 4, while Section 5.4 verified the efficiency and accuracy of the structure designed in Section 3.

5.1. Test of Different Threads in Interferogram Generation Process

For the interferogram generation, the two images were read from the two DDR3 SDRAM chips, with the phase stored in DDRA and the coherence coefficient stored in DDRB. When the thread of coarse registration was 1, we set the threads of interferogram generation as 1, 2, 4, and 8, respectively, and the results of the tests are shown in Table 6. T_c was the measured time in the coarse registration process, and T_g was the estimated time in the interferogram generation process. T_i was the sum of T_g and T_c . In practice, the loss of efficiency in storage space could not be avoided, leading to the differences between Tables 2 and 6. It could also be seen from Table 6 that the processing time rose nearly linearly as the threads increased from 1 to 4. Moreover, the T_g of 8 threads was close to that of 4 threads, indicating that the dual-channel storage structure hampered the system's processing capacity.

Table 6. Test results of interferogram generation with different threads.

Scheme	Thread	R_i	R_{LUT}	R_i/R_{LUT} (%)	T_c (ms)	T_g (ms)	T_i (ms)
a'	1	10,477	433,200	2.42	8.91	710	718.91
b'	2	20,914	433,200	4.83	8.91	357	365.91
c'	4	41,723	433,200	9.63	8.91	186	194.91
d'	8	83,048	433,200	19.17	8.91	165	173.91

The calculation results of ∇G are shown in Figure 9. In the figure, $\nabla G < 0.8 < 3$ meant that there was no balanced scheme. Moreover, it could be seen from the gap between $\Delta R_{LUT}/R_{LUT}$ and $\Delta T/T_i$ that resource changes would sharply deteriorate the processing time.

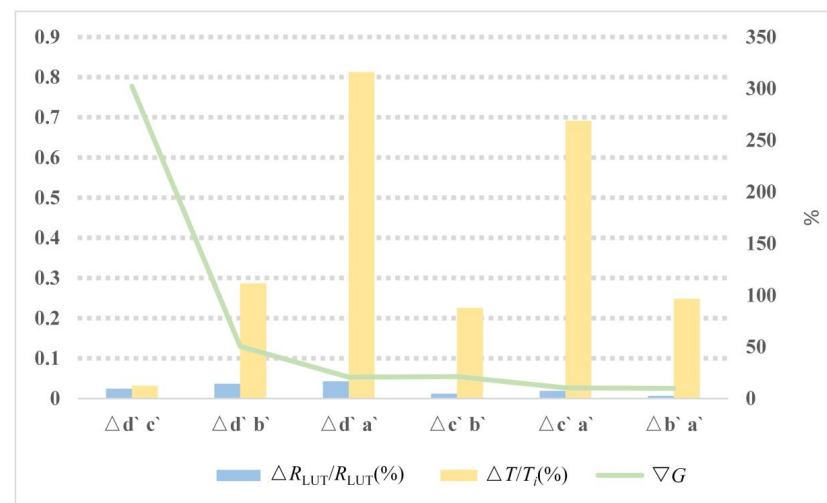


Figure 9. ∇G of schemes in interferogram generation with different threads.

In addition, there would be a slight difference between measured ∇G_T and theoretical ∇G_B due to the efficiency loss in interface or logic, as shown in Table 7. From this table, the relative error ε did not exceed 10% when the thread was 1, 2 and 4, which verified our analysis.

Table 7. Comparison between ∇G in theory and practice.

Scheme	∇G_T (∇G_B)	ε (%)
$\Delta c'b'(\Delta cb)$	0.0545 (0.0503)	8.29
$\Delta c'a'(\Delta ca)$	0.0267 (0.0252)	6.02
$\Delta b'a'(\Delta ba)$	0.0250 (0.0247)	1.28

5.2. Test of Different Threads in Coarse Registration

It has been pointed out in Section 4.3 of Section 4 that when the thread of interferogram generation was 4, the scheme of the thread of coarse registration as 1 was the balanced design. In practice, for interferogram generation, the actual time of the scheme with 8 threads was 164.7 ms, which was longer than the theoretical value of 4 threads (i.e., 160 ms); when the thread of interferogram generation was 8, the thread 1 in coarse registration was still the balanced design of the system in theory. So, as shown in Table 8, we only needed to compare the scheme with 1 thread and the one with 2 threads in the coarse registration to find the balanced design.

Table 8. Test results of coarse registration with different threads.

Scheme	Thread	R_i	R_{DSP}	R_i/R_{DSP} (%)	T_c (ms)	T_g (ms)	T_i (ms)
e'	1	288	3600	8	9.24	164.7	173.94
f'	2	576	3600	16	4.72	164.7	169.42

As shown in Table 9, ∇G of scheme f' and e' was around 3, indicating that the scheme with the thread of coarse registration as 1 was the balanced design. The experiment result confirmed our previous analysis. From Table 4, ∇G in theory between thread 1 and thread 2, that is the scheme d and e , could be calculated according to the Formula (7). ∇G_B was about 3.50 while ∇G_T was around 3.00. Therefore, the relative error ε of ∇G was at 14.21%.

Table 9. ∇G of schemes with different threads in coarse registration.

Scheme	ΔR_{DSP}	$\Delta R_{DSP}/R_{DSP}$ (%)	ΔT (ms)	T_i (ms)	$\Delta T/T_i$ (%)	∇G	ε (%)
$\Delta f'e'$	288	8	4.52	169.42	2.67	3.00	14.21

The value ε was relatively large for the following two reasons. First, the theoretical analysis did not take into account the efficiency loss during the storage process with the continuous address. In addition, after the 512×512 points in the center were found from the $16,384 \times 4096$ image, the addresses would be changed every 512 points taken, which also added to the efficiency loss.

5.3. Verification of Matrix Transposition Schemes

The matrix transposition schemes' theoretical resource requirement and efficiency were pointed out in Section 4.4 of Section 4. Now that the test results of interferogram generation and coarse registration were ready, this section focused on verifying matrix transposition schemes on FPGA. To facilitate the comparison with the theoretical analysis, the thread of interferogram generation and coarse registration was set as 4 (scheme c') and 1 (scheme e'), respectively. Different matrix transposition schemes had varying efficiency, influencing coarse registration processing time T_c , as shown in Table 10.

Table 10. Test results of matrix transposition schemes under the condition of c' and e' .

Scheme	R_i	R_{BRAM_36K}	R_i/R_{BRAM_36K} (%)	T_c (ms)	T_g (ms)	T_i (ms)
g'	260	1470	17.69	9.24	186	195.24
h'	132	1470	8.98	11.16	186	197.16
i'	68	1470	4.63	16.28	186	202.28
j'	36	1470	2.45	39.75	186	225.75

The comparison between the measured value ∇G_T (calculated from Table 10. Test results of matrix transposition schemes under the condition of c' and e') and the theoretical value ∇G_B (calculated from Table 6) is displayed in Figure 10. It can be seen that ∇G_T of $\Delta g'h'$ was above 8 while ∇G_T of $\Delta h'i'$ was below 2, which confirmed the theoretical analysis that scheme h was more balanced than schemes g, i, j . Due to the loss of efficiency,

the measured time T_g and T_i was longer than the theoretical value. As a result, the actual value (∇G_T) was above the theoretical value (∇G_B), which, in turn, caused an increase in the relative error. Yet, the relative error was still lower than 10%.

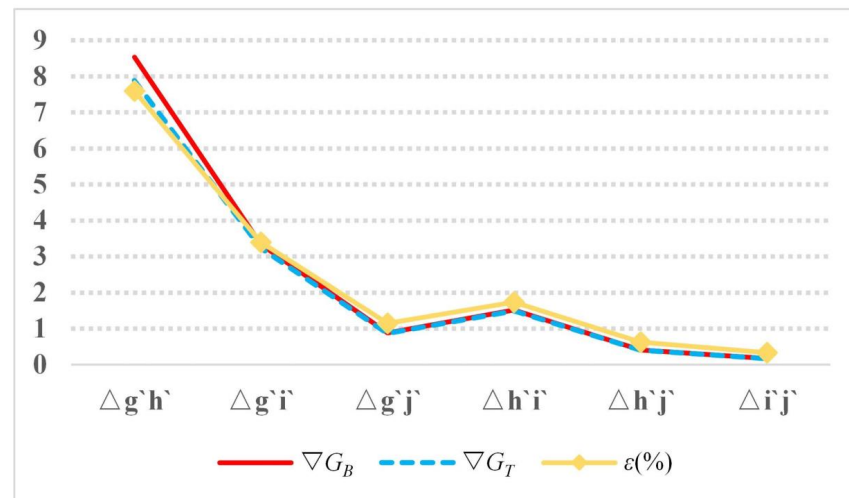


Figure 10. ∇G of matrix transposition schemes under the condition of d' .

It can be proved that under the same resource condition, if the relative error between the measured value (T_{c_test}, T_{g_test}) and the theoretical value is within 20%, that is, $0.8 T_c \leq T_{c_test} \leq 1.2 T_c$ and $0.8 T_g \leq T_{g_test} \leq 1.2 T_g$, then the relative error of ∇G will also be within 20%. From the perspective of engineering, when the ∇G_{gate} is set (e.g., 3) and the relative error range is set (e.g., 20%), the two-dimensional constraint problem of resources and processing capacity could be solved using the theoretical analysis of FPGA proposed in this paper, thus shortening the system optimization time.

We analyzed the influence on the system when the thread of the interferogram generation was 4. When the thread of the interferogram generation changed from 4 to 1, the results are shown in Figure 11. ∇G_T of either $\Delta a'_g h'$ or $\Delta a'_h i'$ was above 3, which meant schemes h' and i' were more balanced than the others. We can see that when the requirement of system processing capacity changed, the balanced scheme would change accordingly. Such a changing relationship could be dynamically reflected by the index ∇G proposed by this paper.

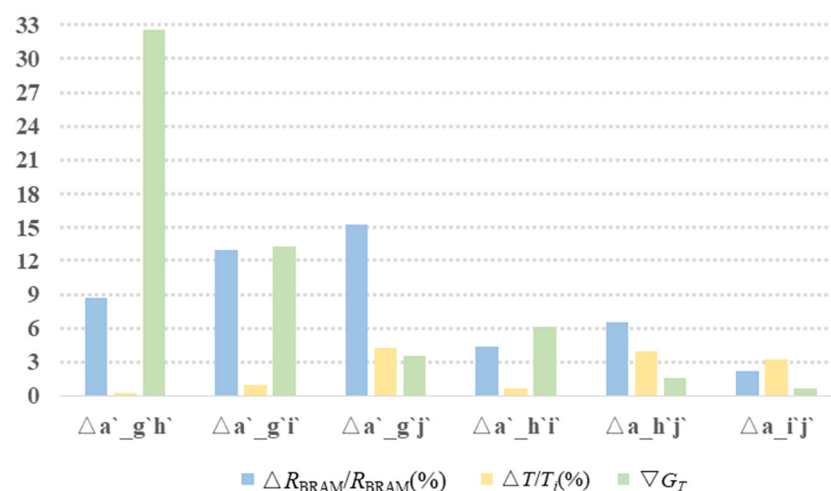


Figure 11. ∇G of matrix transposition schemes under the condition of a' .

5.4. Verification and Comparison of System Performances

This section verifies the system performance from the perspective of accuracy and efficiency.

By comparing the measured results with the theoretical value, the accuracy verification calculates the absolute and relative errors, as shown in Table 11. The absolute error of phase and coherence coefficient was less than 2×10^{-7} , and the relative error was less than 5×10^{-3} , which showed that the FPGA balanced design proposed in this paper had realized the expected algorithm with high precision.

Table 11. Comparison of the accuracy and efficiency of FPGA, DSP, and GPU.

Accuracy										
Phase Coherence coefficient					Relative Error			Absolute Error		
					2.7×10^{-3}			2×10^{-7}		
					4.4×10^{-3}			1×10^{-7}		
Efficiency										
Platform		Ours			[24]		[42]			
		FPGA Virtex-7 100 MHz			DSP TMS320C6678 multicore 1 GHz		GPU NVIDIA GTX Titan Black 980 MHz, 2880 cores			
Interferogram generation	Image (points)	$16,384 \times 4096$			2048×2048		$1494 \times 21,478$	$1505 \times 25,392$	$1510 \times 24,531$	
	Time (ms)	164.7	710		866.52		83.41	98.92	97.31	
	Throughput ($\times 10^6$ points/s)	407.46	94.52		4.84		384.70	386.32	380.66	
Coarse registration	Image (points)	512×512			512×512	1024×1024	2048×2048	n/a	n/a	n/a
	Time (ms)	9.24 ^a	11.16 ^b	39.75 ^c	7.182	17.79	57.52	n/a	n/a	n/a
	Throughput ($\times 10^6$ points/s)	28.37	23.49	6.59	36.50	58.94	72.92	n/a	n/a	n/a

^a, this result is obtained under the condition of g' and e' . ^b, this result is obtained under the condition of h' and e' .
^c, this result is obtained under the condition of j' and e' .

As for efficiency verification, we listed the different processing capacities of different threads to find the range of system processing capacity. Then, the processing capacity was expressed as the throughput, which would be compared with the existing research. The Formula of throughput is $B = D/T$. D is the number of data points, and T is the entire calculation time. The throughput index B represents how fast data points are processed, so it could be a yardstick to measure the processing capacity.

In terms of efficiency verification, we compared the performance of FPGA with central processors like DSP and GPU in coarse registration and interferogram generation. The experimental results are shown in Table 11. Based on the dual-channel storage structure, when the thread of the interferogram generation was 8 and the FPGA working clock was 100 MHz, the throughput of FPGA could reach 407.46×10^6 points/s, which exceeded 386.32×10^6 points/s of the NVIDIA GTX Titan Black GPU. When the thread of interferogram generation was 1, the throughput could reach 94.52×10^6 points/s, which was still far beyond the processing capacity of DSP. Notably, fetching data discontinuously in the calculation process would hamper the data transfer rate of DSP/GPU, which would, in turn, affect the throughput. However, the proposed specialized structure could reduce data transfer loss of FPGA. Thus, the proposed FPGA structure had advantages in the computationally intensive process of InSAR, compared with GPU and DSP.

We have pointed out that when the thread of the generated interferogram was 8, the thread of coarse registration as 1 was the balanced design. At this time, our FPGA took at least 9.24 ms to finish the coarse registration as the scheme g' . The slowest scheme took 39.75 ms as j' . The scheme h' took 11.16 ms as the optimum trade-off, which only increased the entire time by 1.1% but reduced the FPGA BRAM usage by 8.7%. The throughput is shown in Table 11. DSP processed an image with 512×512 points in 7.182 ms, demonstrating a slight advantage over FPGA. To process a 2048×2048 image,

the throughput of DSP (i.e., 72.92×10^6 points/s) was over twice as large as that of FPGA (i.e., 28.37×10^6 points/s), which showed that DSP had better processing capacity in the coarse registration. However, the time of coarse registration only took up less than 10% of the entire time, which meant that coarse registration had little influence on the overall processing capacity of the system. Therefore, it was feasible to realize coarse registration and interferogram generation based on FPGA to process remote sensing images efficiently. The processing capacity could be further improved when the working clock frequency of FPGA rose.

The experimental results indicated that the proposed method could be applied to InSAR on-board processing in the field of ocean observation. With a broader area, the amount of global ocean observation data was much larger than that of land observation. That said, the traditional processing methods on the ground were unsuitable because of the limited satellite-ground communication network speed. By reducing the volume downlink data, on-board processing could fulfill the requirements in ocean observation. Thus, the proposed method could contribute to ocean observation research by improving the performance of spaceborne remote sensing processing.

To our best knowledge, little research has applied on-board processing technology to other fields, such as landslide detection, planetary mission, and volcano monitoring, etc. These fields prioritize obtaining high-precision tiny deformations to extract landslide, planetary, and volcano information. In other words, these fields put more emphasis on high resolution instead of real-time performance at the current stage. However, with the development of on-board processing and multi-satellite collaboration technology [57,58], more complex algorithms of InSAR could be realized. In the future, it is possible that on-board processing technology could also be applied in other InSAR scenarios, which requires more research to balance the needs for accuracy and timeliness.

Focusing on the key issue of on-board processing, that is, the trade-off between resources and processing capacity, this method could be applied in on-board processing technology. Besides, the method can also be applied to other fields, such as edge computing [59,60] and UAV platforms [61] etc.

6. Conclusions

In this paper, research on hardware structure and design method was carried out, based on the processes of InSAR processing, namely coarse registration and interferogram generation. First, the proposed efficient system can ensure fast data processing from the perspective of data communication, processing flow, and storage. Second, given the dual constraints between resources and processing capacity, this paper proposes an FPGA design method based on the gradient descent theory. Mapping the two constraints to two-dimensional coordinates, this method helps identify the optimum trade-off through gradient values, providing guidance for engineering practice in general and, thus, shortening the system optimization process. In the empirical part, the experiment results verify the system's accuracy, the structure's efficiency, and the design method's effectiveness. The results show that the proposed method can be applied to the field of ocean observation. Finally, this paper discusses the generalizability of the proposed method. This paper argues that the proposed method is most suitable for ocean observation research and can be potentially applied to other fields as on-board processing technology develops and the algorithm's complexity increases.

In the near future, we will verify the proposed FPGA design method in other algorithms and study the generalizability of the design method under multiple constraints.

Author Contributions: Conceptualization, J.L. and Y.X.; methodology, J.L. and M.X.; validation, M.X., J.L. and Y.X.; formal analysis, J.L.; investigation, M.X.; resources, H.C. and Y.X.; writing—original draft preparation, J.L.; writing—review and editing, M.X. and H.C.; supervision, H.C.; project administration, Y.X.; funding acquisition, H.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Science Fund for Distinguished Young Scholars under Grant 2018-JCJQ-ZQ-046 and partly by the Civil Aviation Program under Grant B0201.

Acknowledgments: The authors would like to thank the reviewers for their kind comments and thoughtful suggestions which greatly improve this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Massonnet, D.; Rossi, M.; Carmona, C.; Adragna, F.; Peltzer, G.; Feigl, K.; Rabaute, T. The displacement field of the Landers earthquake mapped by radar interferometry. *Nature* **1993**, *364*, 138–142. [CrossRef]
2. Fielding, E.J.; Talebian, M.; Rosen, P.A.; Nazari, H.; Jackson, J.A.; Ghorashi, M.; Walker, R. Surface ruptures and building damage of the 2003 Bam, Iran, earthquake mapped by satellite synthetic aperture radar interferometric correlation. *J. Geophys. Res. Earth Surf.* **2005**, *110*, B03302. [CrossRef]
3. Shen, Z.-K.; Sun, J.; Zhang, P.; Wan, Y.; Wang, M.; Burgmann, R.; Zeng, Y.; Gan, W.; Liao, H.; Wang, Q. Slip maxima at fault junctions and rupturing of barriers during the 2008 Wenchuan earthquake. *Nat. Geosci.* **2009**, *2*, 718–724. [CrossRef]
4. Massonnet, D.; Briole, P.; Arnaud, A. Deflation of Mount Etna monitored by spaceborne radar interferometry. *Nature* **1995**, *375*, 567–570. [CrossRef]
5. Amelung, F.; Jónsson, S.; Zebker, H.; Segall, P. Widespread Uplift and ‘Trapdoor’ Faulting on Galapagos Volcanoes Observed with Radar Interferometry. *Nature* **2000**, *407*, 993–996. [CrossRef]
6. Pritchard, M.; Simons, M. An InSAR-based survey of volcanic deformation in the central Andes. *Geochem. Geophys. Geosyst.* **2004**, *5*. [CrossRef]
7. Dai, K.; Xu, Q.; Li, Z.; Tomás, R.; Fan, X.; Dong, X.; Li, W.; Zhou, Z.; Gou, J.; Ran, P. Post-disaster assessment of 2017 catastrophic Xinmo landslide (China) by spaceborne SAR interferometry. *Landslides* **2019**, *16*, 1189–1199. [CrossRef]
8. Li, L.; Yao, X.; Yao, J.; Zhou, Z.; Feng, X.; Liu, X. Analysis of Deformation Characteristics for a Reservoir Landslide before and after Impoundment by Multiple D-InSAR Observations at Jinshajiang River, China. *Nat. Hazards* **2019**, *98*, 719–733. [CrossRef]
9. Xue, F.; Lv, X. Applying time series interferometric synthetic aperture radar and the unscented Kalman filter to predict deformations in Maoxian landslide. *J. Appl. Remote Sens.* **2019**, *13*, 014509. [CrossRef]
10. Massonnet, D.; Holzer, T.; Vadon, H. Land subsidence caused by the East Mesa Geothermal Field, California, observed using SAR interferometry. *Geophys. Res. Lett.* **1997**, *24*, 901–904. [CrossRef]
11. Amelung, F.; Galloway, D.; Bell, J.W.; Zebker, H.A.; Lacznia, R.J. Sensing the ups and downs of Las Vegas: InSAR reveals structural control of land subsidence and aquifer-system deformation. *Geology* **1999**, *27*, 483–486. [CrossRef]
12. Hoffmann, J.; Zebker, H.A.; Galloway, D.; Amelung, F. Seasonal subsidence and rebound in Las Vegas Valley, Nevada, observed by Synthetic Aperture Radar Interferometry. *Water Resour. Res.* **2001**, *37*, 1551–1566. [CrossRef]
13. Balzter, H.; Rowland, C.; Saich, P. Forest canopy height and carbon estimation at Monks Wood National Nature Reserve, UK, using dual-wavelength SAR interferometry. *Remote Sens. Environ.* **2007**, *108*, 224–239. [CrossRef]
14. Brown, C.; Sarabandi, K. Model-based estimation of forest canopy parameters using polarimetric and interferometric SAR. In Proceedings of the IGARSS 2001. Scanning the Present and Resolving the Future. Proceedings. IEEE 2001 International Geo-science and Remote Sensing Symposium (Cat. No. 01CH37217), Sydney, NSW, Australia, 9–13 July 2001; IEEE: New York, NY, USA, 2001; Volume 1, pp. 357–359. [CrossRef]
15. Ghail, R.C.; Hall, D.; Mason, P.J.; Herrick, R.R.; Carter, L.M.; Williams, E. VenSAR on EnVision: Taking Earth Observation Radar to Venus. *Int. J. Appl. Earth Obs. Geoinf.* **2018**, *64*, 365–376. [CrossRef]
16. Seu, R.; Smrekar, S.; Hensley, S.; Pierfrancesco, L. A SAR Interferometer Experiment to Explore the Surface of Venus. In Proceedings of the EUSAR 2016: 11th European Conference on Synthetic Aperture Radar, Hamburg, Germany, 6–9 June 2016; VDE VERLAG GMBH: Offenbach, Germany, 2016; pp. 1242–1244.
17. Hensley, S.; Smrekar, S.; Shaffer, S.; Paller, M.; Figueroa, H.; Freeman, A.; Walkemeyer, P. VISAR: A Next Generation Interferometric Radar for Venus Exploration. In Proceedings of the 2015 IEEE 5th Asia-Pacific Conference on Synthetic Aperture Radar (APSAR), Singapore, 1–4 September 2015; IEEE: New York, NY, USA, 2015; pp. 362–366.
18. Dellepiane, S.; De Laurentiis, R.; Giordano, F. Coastline extraction from SAR images and a method for the evaluation of the coastline precision. *Pattern Recognit. Lett.* **2004**, *25*, 1461–1470. [CrossRef]
19. Schulz-Stellenfleth, J.; Lehner, S. Ocean wave imaging using an airborne single pass across-track interferometric SAR. *IEEE Trans. Geosci. Remote Sens.* **2001**, *39*, 38–45. [CrossRef]
20. Schulz-Stellenfleth, J.; Horstmann, J.; Lehner, S.; Rosenthal, W. Sea surface imaging with an across-track interferometric synthetic aperture radar: The SINEWAVE experiment. *IEEE Trans. Geosci. Remote Sens.* **2001**, *39*, 2017–2028. [CrossRef]
21. Sugimoto, Y.; Ozawa, S.; Inaba, N. Spaceborne Synthetic Aperture Radar Signal Processing Using FPGAs. In *High-Performance Computing in Geoscience and Remote Sensing VII*; Huang, B., López, S., Wu, Z., Eds.; SPIE: Warsaw, Poland, 2017; p. 12.
22. JAXA | Japan Aerospace Exploration Agency (JAXA) and Alouette Technology Develop Onboard Image Processor for Synthetic Aperture Radar (SAR) Data. Available online: https://global.jaxa.jp/press/2020/02/202002261_e.html (accessed on 10 September 2022).

23. Peral, E.; Esteban-Fernandez, D. Swot Mission Performance and Error Budget. In Proceedings of the IGARSS 2018—2018 IEEE International Geoscience and Remote Sensing Symposium, Valencia, Spain, 22–27 July 2018; IEEE: Valencia, Spain, 2018; pp. 8625–8628.
24. Flight Systems | Mission. Available online: <https://swot.jpl.nasa.gov/mission/flight-systems> (accessed on 10 September 2022).
25. Zhu, X.X.; Bamler, R.; Lachaise, M.; Adam, F.; Shi, Y.; Eineder, M. Improving TanDEM-X DEMs by Non-Local InSAR Filtering. In Proceedings of the EUSAR 2014: 10th European Conference on Synthetic Aperture Radar, Berlin, Germany, 3–5 June 2014; pp. 1–4.
26. Baier, G.; Zhu, X.X.; Lachaise, M.; Breit, H.; Bamler, R. Nonlocal InSAR Filtering for DEM Generation and Addressing the Staircasing Effect. In Proceedings of the EUSAR 2016: 11th European Conference on Synthetic Aperture Radar, Hamburg, Germany, 6–9 June 2016; pp. 1–4.
27. Costantini, M. A novel phase unwrapping method based on network programming. *IEEE Trans. Geosci. Remote Sens.* **1998**, *36*, 813–821. [[CrossRef](#)]
28. Huang, Q.; Zhou, H.; Dong, S.; Xu, S. Parallel Branch-Cut Algorithm Based on Simulated Annealing for Large-Scale Phase Unwrapping. *IEEE Trans. Geosci. Remote Sens.* **2015**, *53*, 3833–3846. [[CrossRef](#)]
29. Reza, T.; Zimmer, A.; Blasco, J.M.D.; Ghuman, P.; Aasawat, T.K.; Ripeanu, M. Accelerating Persistent Scatterer Pixel Selection for InSAR Processing. *IEEE Trans. Parallel Distrib. Syst.* **2017**, *29*, 16–30. [[CrossRef](#)]
30. Song, B.; Pan, M.; Hu, X. Fast Implementation Method of Interferometric Synthetic Aperture Radar Raw-Signal Simulation. *J. Comput. Theor. Nanosci.* **2016**, *13*, 6155–6160. [[CrossRef](#)]
31. Guerriero, A.; Anelli, V.W.; Pagliara, A.; Nutricato, R.; Nitti, D.O. Efficient Implementation of InSAR Time-Consuming Algorithm Kernels on GPU Environment. In Proceedings of the 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Milan, Italy, 26–31 July 2015; IEEE: New York, NY, USA, 2015; pp. 4264–4267.
32. Heping, Z.; Sen, Z.; Zhen, T.; Jinsong, T. A Fast Quality-guided Phase Unwrapping Algorithm in Heterogeneous Environment. *Geomat. Inf. Sci. Wuhan Univ.* **2015**, *40*, 756–760. [[CrossRef](#)]
33. Gao, S.; Zeng, Q.; Jiao, J.; Liang, C.; Tong, Q. Parallel Processing of InSAR Interferogram Filtering with CUDA Programming. *Sci. Surv. Mapp.* **2015**, *1*, 54–68.
34. Chao, W.; Zhang, Z.; Tang, Y.; Zhang, H.; Chen, F. A Fast InSAR Processing System Based on Digital Signal Processor. In Proceedings of the 2016 4th International Workshop on Earth Observation and Remote Sensing Applications (EORSA), Guangzhou, China, 4–6 July 2016; pp. 83–86.
35. Huey, K. Xilinx Virtex-5QV Update and Space Roadmap. *Indico* **2016**, *17*. Available online: https://indico.esa.int/event/130/contributions/717/attachments/767/943/Xilinx_V5QV_Space_Roadmap_2016.03.17_SEFUW.pdf (accessed on 1 September 2022).
36. Qinwen, W.U. High Efficiency Matrix Transposition Method Based on FPGA and DDR. *Mod. Radar* **2017**, *39*, 34–40.
37. Akin, B.; Milder, P.A.; Franchetti, F.; Hoe, J.C. Memory Bandwidth Efficient Two-Dimensional Fast Fourier Transform Algorithm and Implementation for Large Problem Sizes. In Proceedings of the 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines, Toronto, ON, Canada, 29 April–1 May 2012; IEEE: New York, NY, USA, 2012; pp. 188–191. [[CrossRef](#)]
38. Huanghui, S.; Zhensong, W.; Weimin, Z. An Efficient Memory Access Strategy for Transposition and Block Operation in Image Processing. *J. Comput. Res. Dev.* **2013**, *50*, 188.
39. Yang, C.; Li, B.; Chen, L.; Wei, C.; Xie, Y.; Chen, H.; Yu, W. A Spaceborne Synthetic Aperture Radar Partial Fixed-Point Imaging System Using a Field-Programmable Gate Array—Application-Specific Integrated Circuit Hybrid Heterogeneous Parallel Acceleration Technique. *Sensors* **2017**, *17*, 1493. [[CrossRef](#)]
40. Li, Q.; Yan, B.-Z.; Mao, H.-K.; Xue, X.-F.; Han, Q.; Guo, H. High-Speed and Adaptive FPGA-Based Privacy Amplification in Quantum Key Distribution. *IEEE Access* **2019**, *7*, 21482–21490. [[CrossRef](#)]
41. Li, B.; Shi, H.; Chen, L.; Yu, W.; Yang, C.; Xie, Y.; Bian, M.; Zhang, Q.; Pang, L. Real-Time Spaceborne Synthetic Aperture Radar Float-Point Imaging System Using Optimized Mapping Methodology and a Multi-Node Parallel Accelerating Technique. *Sensors* **2018**, *18*, 725. [[CrossRef](#)]
42. Sun, T.; Xie, Y.; Li, B. Efficiency balanced matrix transpose method for sliding spotlight SAR imaging processing. *J. Eng.* **2019**, *2019*, 7775–7778. [[CrossRef](#)]
43. Wang, S.; Zhang, S.; Huang, X.; An, J.; Chang, L. A Highly Efficient Heterogeneous Processor for SAR Imaging. *Sensors* **2019**, *19*, 3409. [[CrossRef](#)]
44. Wang, G.; Chen, H.; Xie, Y. An Efficient Dual-Channel Data Storage and Access Method for Spaceborne Synthetic Aperture Radar Real-Time Processing. *Electronics* **2021**, *10*, 662. [[CrossRef](#)]
45. Fu, Y.; Wu, E.; Santhaseelan, V.; Denolf, K.; Khan, K.; Kathail, V. Embedded Vision with INT8 Optimization on Xilinx Devices. *WP490* **2017**, *19*, 15.
46. Yu, W.; Xie, Y.; Lu, D.; Li, B.; Chen, H.; Chen, L. Algorithm Implementation of On-Board SAR Imaging on FPGA+DSP Platform. In Proceedings of the 2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP), Chongqing, China, 11–13 December 2019; IEEE: Chongqing, China, 2019; pp. 1–5. [[CrossRef](#)]
47. Ye, W.; Li, J.; Li, L. Design and Development of a Real-Time Multi-DSPs and FPGA-Based DPOS for InSAR Applications. *IEEE Sens. J.* **2018**, *18*, 3419–3425. [[CrossRef](#)]

48. Ding, Z.; Xiao, F.; Xie, Y.; Yu, W.; Yang, Z.; Chen, L.; Long, T. A Modified Fixed-Point Chirp Scaling Algorithm Based on Updating Phase Factors Regionally for Spaceborne SAR Real-Time Imaging. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 7436–7451. [\[CrossRef\]](#)
49. Qiu, J.; Wang, J.; Yao, S.; Guo, K.; Li, B.; Zhou, E.; Yu, J.; Tang, T.; Xu, N.; Song, S.; et al. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. In Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 21–23 February 2016; ACM: Monterey, CA, USA, 2016; pp. 26–35.
50. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv* **2016**, arXiv:1510.00149.
51. Massonnet, D.; Feigl, K.L. Radar interferometry and its application to changes in the Earth's surface. *Rev. Geophys.* **1998**, *36*, 441–500. [\[CrossRef\]](#)
52. Yu, Y.; Balz, T.; Luo, H.; Liao, M.; Zhang, L. GPU accelerated interferometric SAR processing for Sentinel-1 TOPS data. *Comput. Geosci.* **2019**, *129*, 12–25. [\[CrossRef\]](#)
53. Li, Y.; Jiang, W.; Zhang, J. A time series processing chain for geological disasters based on a GPU-assisted sentinel-1 InSAR processor. *Nat. Hazards* **2021**, *111*, 803–815. [\[CrossRef\]](#)
54. Papalambros, P.Y.; Wilde, D.J. *Principles of Optimal Design: Modeling and Computation*; Cambridge University Press: Cambridge, UK, 2000; ISBN 978-0-521-62727-6.
55. AMD Xilinx. Available online: https://docs.xilinx.com/v/u/en-US/ug761_axi_reference_guide (accessed on 1 September 2022).
56. AMD Xilinx. Available online: https://www.xilinx.com/content/dam/xilinx/support/documentation/ip_documentation/axi_ref_guide/latest/ug1037-vivado-axi-reference-guide.pdf (accessed on 1 September 2022).
57. Liu, Y.; Dai, Y.; Liu, G.; Yang, J.; Tian, L.; Li, H. Distributed Space Remote Sensing and Multi-Satellite Cooperative on-Board Processing. In Proceedings of the 2020 International Conference on Sensing, Measurement & Data Analytics in the era of Artificial Intelligence (ICSMD), Xi'an, China, 15–17 October 2020; IEEE: New York, NY, USA, 2020; pp. 551–556. [\[CrossRef\]](#)
58. Xue, Y.; Li, Y.; Guang, J.; Zhang, X.; Guo, J. Small satellite remote sensing and applications—History, current and future. *Int. J. Remote Sens.* **2008**, *29*, 4339–4372. [\[CrossRef\]](#)
59. Biookaghazadeh, S.; Zhao, M.; Ren, F. Are {FPGAs} Suitable for Edge Computing? In Proceedings of the USENIX Workshop on Hot Topics in Edge Computing (Hot Edge 18), Boston, MA, USA, 11–13 July 2018.
60. Rodríguez, A.; Navarro, A.; Asenjo, R.; Corbera, F.; Gran, R.; Suárez, D.; Nunez-Yanez, J. Exploring heterogeneous scheduling for edge computing with CPU and FPGA MPSoCs. *J. Syst. Arch.* **2019**, *98*, 27–40. [\[CrossRef\]](#)
61. Lou, Y.; Clark, D.; Marks, P.; Muellerschoen, R.J.; Wang, C.C. Onboard Radar Processor Development for Rapid Response to Natural Hazards. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2016**, *9*, 2770–2776. [\[CrossRef\]](#)