Engineering Technology Trends Vol 2 Issue 4 2024

Original Research Article

Open Access

Omniscient Pte.Ltd.

Imaging Design of SAR Measured Data Based on SSMEC DSP

Wen-Qing Wang*, Jiang He, Peng Wang, Tao Zhang

Shenzhen State Microelectronics Co., Ltd, Shenzhen, Guangdong, 518000, China

***Correspondence to:** Wen-Qing Wang, Shenzhen State Microelectronics Co., Ltd, Shenzhen, Guangdong, 518000, China, E-mail: wqwang@ssmec.com

Abstract: Aiming at the problem of large computational load and long processing time of Synthetic Aperture Radar (SAR), this paper adopts SSMEC DSP to complete the low-squint SAR imaging processing, including range compression, azimuthal FFT, Range Cell Migration Correction (RCMC) and azimuthal compression. Eventually, the effectiveness and high efficiency of SSMEC DSP realizing RDA is verified by the real data imaging, which can process 4096*4096 points of data within 831.2ms to meet the real-time imaging requirements.

Keywords: Synthetic aperture radar; Range Doppler algorithm; Multi-core DSP

1. Introduction

Synthetic Aperture Radar (SAR) is characterized by all-weather, all-day, and high-resolution capabilities, making it widely applicable in many fields[1]. The imaging algorithm is at the core of SAR, and among them, the Range Doppler Algorithm (RDA) is both efficient and highly accurate[2].

SAR imaging involves large amounts of data and complex signal processing, making the realization of high-performance real-time SAR one of the primary challenges[3]. SSMEC DSP chip integrates CPU and DSP processor, with the highest performance of 512 GFLOPS.This paper uses the classic RDA and, based on the characteristics of the SSMEC DSP, designs accelerated operators and parallel processing algorithms to achieve real-time SAR imaging.

2. Range Doppler Algorithm Principle

The classic RDA reduces imaging to two decoupled one-dimensional compression processes: through range compression, achieving high resolution in the range direction; by interpolation in the range-Doppler domain, achieving decoupling of the range and azimuth directions, and completing Range Cell Migration Correction (RCMC); and through azimuth compression, achieving high resolution in the azimuth direction, thus completing target imaging[4][5].

2.1 Range Compression

The baseband signal after demodulation is obtained by performing FFT on both the signal and the reference function, followed by an IFFT transformation in the range direction to complete range compression. The baseband signal after demodulation is

$$s_o(\tau,\eta)=A_o\omega_r\left[\tau-\frac{2R(\eta)}{c}\right]\omega_a[\eta-\eta_c]exp\left\{-\frac{j4\pi f_0R(\eta)}{c}\right\}exp\left\{j\pi K_r\left[\tau-\frac{2R(\eta)}{c}\right]^2\right\}$$

The range-direction matched filter is chosen as $H(f_\tau)$

$$H(f_\tau)=rect\left(\frac{f_\tau}{|K|T}\right)exp\left\{+j\pi\frac{f_\tau^2}{K}\right\}$$

Then the range compression output is

$$s_{rc}(\tau,\eta)=IFFT\{S_o(f_\tau,\eta)H(f_\tau)\}=A_op_r\left[\tau-\frac{2R(\eta)}{c}\right]\omega_a[\eta-\eta_c]exp\left\{-\frac{j4\pi f_0\mathrm{R}(\eta)}{c}\right\}$$

2.2 Azimuth Fourier Transform and Range Walk Correction

In the case of a low squint angle, the range equation is approximated as a parabola, and the range-compressed signal is

$$S_{rc}(\tau,\eta)\approx A_op_r\left[\tau-\frac{2R(\eta)}{c}\right]w_a[\eta-\eta_c]exp\left\{-j\frac{4\pi f_0R_0}{c}\right\}exp\left\{-j\pi\frac{2V_r^2}{\lambda R_0}\eta^2\right\}$$

The signal after azimuth FFT is

$$S_1(\tau,f_\eta)=FFT_\eta\{S_{rc}(\tau,\eta)\}=A_op_r\left[\tau-\frac{2R_{rd}(f_\eta)}{c}\right]W_a[f_\eta-f_{\eta_c}]exp\left\{-j\frac{4\pi f_0R_0}{c}\right\}exp\left\{j\pi\frac{f_\eta^2}{K_a}\right\}$$

The RCM that needs to be corrected is

$$\Delta R(f_\eta)=\frac{\lambda^2R_0f_\eta^2}{8V_r^2}$$

In this paper, sinc interpolation is used to implement RCMC. If the correction is accurate, the interpolated signal is

$$S_2(\tau,f_\eta)=A_op_r\left[\tau-\frac{2R_0}{c}\right]W_a[f_\eta-f_{\eta_c}]exp\left\{-j\frac{4\pi f_0R_0}{c}\right\}exp\left\{j\pi\frac{f_\eta^2}{K_a}\right\}$$

At this point, the range envelope p_r is independent of the azimuth frequency, indicating that the RCM has been accurately corrected.

2.3 Azimuth Compression

The azimuth matched filter can be chosen as

$$H_{az}(f_\eta)=exp\left\{-j\pi\frac{f_\eta^2}{K_a}\right\}$$

To achieve azimuth compression, multiply $S_2(\tau,f_\eta)$ by $H_{az}(f_\eta)$ and then apply IFFT to complete the compression

$$s_{ac}(\tau,\eta)=IFFT_\eta\{S_3(\tau,f_\eta)\}=A_op_r\left[\tau-\frac{2R_0}{c}\right]p_a(\eta)exp\left\{-j\frac{4\pi f_0R_0}{c}\right\}exp\{j2\pi f_{\eta_c}\eta\}$$

3. RDA Design Based on SSMEC DSP

3.1 Introduction to SSMEC DSP

The SSMEC DSP is a high-performance heterogeneous intelligent computing SoC that integrates a dual-core CPU main processor and a dual-core DSP coprocessor (as shown in **Figure 1**). It offers a peak performance of up to 512 GFLOPS, features 20MB of large on-chip memory, and provides a variety of I/O interfaces. It is suitable for applications in digital signal processing, image processing, communication control, data security and more.

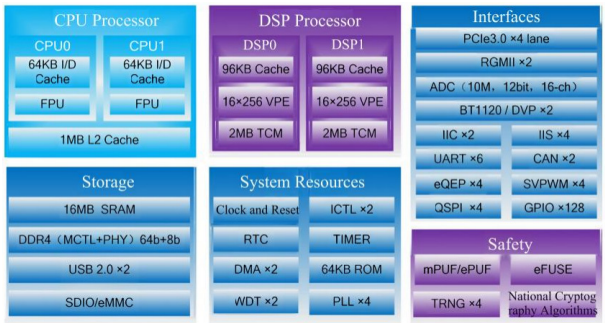

Figure 1. SSMEC DSP Architecture

3.2 RDA Implementation Process

The RDA (Range-Doppler Algorithm) includes

frequency-domain range reference signal generation, range compression, azimuth FFT, RCMC, and azimuth compression[6]. The frequency-domain range reference signal is needed multiple times, so precomputing it can save processing time. Range compression involves the FFT of the echo signal, multiplication by the reference signal, and IFFT. To ensure data addresses are continuous, a matrix transpose is performed before the azimuth FFT, making full use of the DMA's ability to move data continuously. The RCMC is handled by a combination of CPU and DSP operators. Azimuth compression includes reference signal multiplication and IFFT.

3.3 Operator Design

As shown in **Figure 2**, the FFT, IFFT, vector multiplication, matrix transpose, and RCMC are called frequently, and their processing efficiency determines the overall efficiency of the RDA. This paper focuses on the design of the more complex operators.

3.3.1 FFT/IFFT Operator

The DSP performs the FFT calculations, while the CPU handles flow control, DMA data movement, and coordination with the DSP calculations. To balance speed and versatility, a radix-2 method is used for the FFT, requiring a total of log2(N) iterations. DSP acceleration is based on vector instructions, with input data arranged to maximize computational efficiency. The input complex signals are alternately stored as real and imaginary parts, taking full advantage of the in-place storage benefits of butterfly calculations. The IFFT can reuse the same calculation architecture as the FFT.

3.3.2 Matrix Transpose Operator

Each VPE (Vector Processing Engine) is divided into two groups with a total of four parts. In each group, one part stores the source data, and the other stores the transposed result. The two groups process in a ping-pong fashion. The source data is loaded into the DSP by rows, and within the DSP, matrix transposition is accomplished through a combination of vector permutation and rearrangement instructions. The transposed result is stored by columns, and a VDMA controller is used to move the data as needed. Data is divided into complex submatrices, sliced by columns, and then distributed among the 16 VPEs for transposition. Other submatrices are handled by the CPU, which splits tasks to complete the matrix transposition.

3.3.3 RCMC Operator

The DSP's internal vector registers have a maximum equivalent width of 512 bytes, allowing up to 128 floating-point data sequences to be loaded at once. Therefore, the original matrix needs to be column-split into 32 submatrices of size 4096×128. During initialization, the CPU generates a 256-length sinc sequence, with each value repeated 128 times and loaded into the DSP via DMA. In subsequent calculations, vector instructions are used to parallelize the lookup of 128 sequences, obtaining calculation coefficients and column offsets.

During each calculation, the DMA loads 4096×128 numbers into the DSP. Each column of data undergoes offset processing, with specific offsets pre-calculated. The vector load instructions can extract 128 columns of data into vector registers in parallel, and vector store instructions sequentially write back the data. These two instructions complete the offset for 1×128 data points, and repeating this process 4096 times completes the offset for all data.

For the sliding correlation calculation, vector load instructions sequentially extract 8 adjacent rows of data from DMEM and perform multiply-accumulate operations. After traversing all the rows, the sliding correlation for the 4096×128 matrix is completed. Once all calculations are finished, the DMA moves the data to external memory, and the next 4096×128 submatrix is loaded for computation, repeating until all data is processed.

3.4 Multi-core Parallel Processing Design

To fully utilize the multi-core DSP resources and optimize performance, the following processing flow is implemented (as shown in **Figure 2**):

3.4.1 The CPU starts DSP0 and uses DMA to move the first batch of data into storage area 0.

3.4.2 After the transfer is complete, DSP0 starts processing the data with multiplication, while DSP1's DMA transfers the second batch of data into storage.

3.4.3 DSP1 completes the data transfer and performs multiplication.

3.4.4 After DSP0 finishes the calculation, the CPU starts DSP0's DMA to transfer the results to DDR.

3.4.5 After DSP1 completes its calculations, the CPU

starts DSP1's DMA to transfer its results to DDR.

3.4.6 After DSP1 finishes the data transfer, steps 1-5 are repeated.

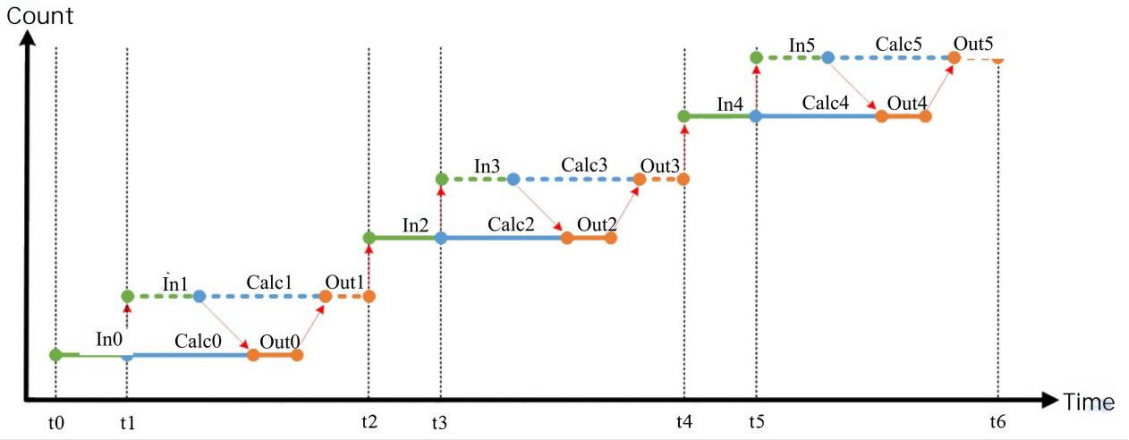

Figure 2. Multi-core Parallel Processing

From the above figure, the multi-core processing time is

$$T_d = 0.5N(T_i + T_a)$$

Where T_d represents the multi-core processing time, T_i is the time for DMA data transfer, and T_a is the total time for operator processing.

The single-core processing time is

$$T_s = NT_a$$

The speed-up ratio of the multi-core processing is

$$r = \frac{T_s}{T_d} = \frac{NT_a}{0.5N(T_i + T_a)} = \frac{T_a}{0.5(T_i + T_a)} = \frac{2}{1 + T_i / T_a}$$

When the T_i is smaller, the speed-up ratio is larger.

4. Imaging Verification and Result Analysis

This paper selects radar echo data collected by the RADARSAT-1 satellite over the Vancouver area of Canada in June 2002. The raw data is first processed with parameter setting, data extraction, and format conversion. Then, the extracted data and parameters are input into the SSMEC DSP for processing.

4.1 Imaging with Measured Data

The imaging area covered in this paper includes Gambier Island, Bowen Island, and the surrounding waters north of Vancouver, Canada. After processing with the SSMEC DSP, the output is further adjusted for contrast, ultimately presenting the grayscale image shown on the left side of **Figure 3**. The image clearly reveals the land-sea boundary, islands, and mountains. The right side of **Figure 3** shows the corresponding area as captured by a satellite map. A comparison reveals a good match between the positions and outlines of the islands, coastlines, and mountains, demonstrating the effectiveness of the implementation proposed in this paper.

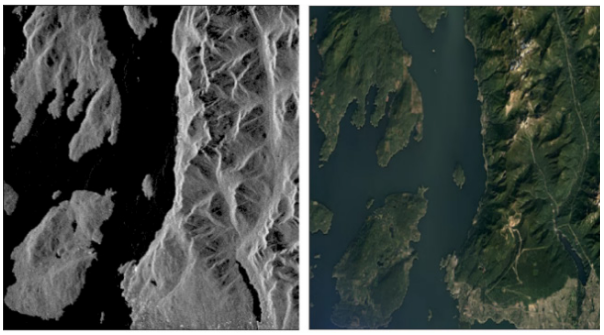

Figure 3. Imaging Result from Measured Data (Left) and Optical Image of the Radar Imaging Area (Right)

4.2 Processing Time Analysis

Table 1. Processing Time Comparison

Module	SSMEC DSP Time (ms)	TMS320C6678 Time (ms)
Range Compression	348.16	620.80
Matrix Transpose	42.88	215.04
Azimuth FFT	133.76	170.48
RCMC	122.08	709.12
Azimuth Compression	141.44	182.80
Matrix Transpose	42.88	215.04
Total	**831.20**	**2113.28**

Table 1 compares the time required by the SSMEC DSP and the TMS320C6678 to process the same image, including the time spent on each module. It can be seen that the SSMEC DSP requires less time for range compression, azimuth FFT, RCMC, and azimuth compression, indicating the high efficiency of the implementation described in this paper.

Conclusion

This paper verifies the effectiveness and efficiency of implementing RDA using the SSMEC DSP through imaging with measured data. The implementation includes range compression, azimuth FFT, RCMC, and azimuth compression. Compared to the TMS320C6678, the SSMEC DSP significantly reduces processing time, providing greater support for real-time processing in low squint SAR imaging and promoting the application of SAR technology.

Reference

[1] Deng Yunkai, Yu Weidong, Zhang Heng, et al. Development trend of future satellite-based SAR technology[J]. Radar Journal, 2020, 9(01): 1-33.

[2] Liu Mengjie, Dai Dahai, Wu Hao, et al. Research on airborne SAR live data imaging based on RD algorithm[J]. Radio Engineering, 2019, 49(09): 807-811.

[3] Chen Y. Design of FPGA+DSP based ballistic SAR signal processing system[J]. Electronic Technology Application, 2019, 45(09): 101-105. DOI:10.16157/j.issn.0258-7998.190441.

[4] You Chang,Wan Liping,Lu Di.SAR processing-principle of range Doppler algorithm[J]. Mapping and Spatial Geographic Information, 2013, 36(11): 144-147.

[5] Jia Haowen,Yang Pengju. A study on the imaging of satellite-borne SAR live data based on range Doppler algorithm[J]. Journal of Yan'an University (Natural Science Edition), 2024, 43(01): 96-101. DOI:10.13876/J.cnki.ydnse.230019.

[6] Hao Dongqing,Liu Yan.Algorithm design and implementation of SAR imaging on DSP platform[J]. Computer Simulation, 2016, 33(11): 5-8.