The Journal of Engineering

IET International Radar Conference (IRC 2018)

IET Journals
The Institution of Engineering and Technology

Design and implementation of MIMO radar real-time imaging system based on multicore DSP

eISSN 2051-3305
Received on 20th February 2019
Accepted on 2nd May 2019
E-First on 10th July 2019
doi: 10.1049/joe.2019.0332
www.ietdl.org

Hongyan Mei[1,2], Weiming Tian[1,2] ✉, Ruiguo Huo[1,2], Jingyang Wang[1,2]

[1]*Radar Research Laboratory, School of Information and Electronics, Beijing Institute of Technology, Beijing, People's Republic of China*
[2]*Key Laboratory of Electronic and Information Technology in Satellite Navigation (Beijing Institute of Technology),Ministry of Education, Beijing, 100081, People's Republic of China*
✉ *E-mail: tianwei6779@163.com*

Abstract: Multiple-input multiple-output (MIMO) imaging radar is a new kind of new imaging radar in recent years. It can achieve faster data acquisition speed than the traditional ground-based SAR due to the ability to get a large number of multi-channel data in a short period of time. However, traditional single-core digital signal processors (DSPs) have limited space and ability to deal with such large amount of data. To solve this problem, a real-time image processing software architecture for MIMO imaging radar algorithm is designed and carried out based on TMS320C6678 DSP of Texas Instruments (TI), which is the highest-performance fixed/floating-point DSP based on TI's KeyStone multicore architecture. The efficient imaging algorithms mapping from Matlab to DSP is realised, with efficient task allocation, resource reuse of echo data, and processing variable, and using the method of multicore parallel processing.

1 Introduction

The characteristic of MIMO imaging radar lies in that its multi-antenna array structures can be equivalent to a large synthetic aperture, which can improve the resolution of azimuth direction. This capability of acquiring information with multi-channel greatly improves the radar imaging performance. Using a single 'snapshot' data for imaging without time accumulation can avoid the problem of range migration in traditional synthetic aperture radar, which make it an emerging radar in recent years. Real-time processing of MIMO imaging radar echo data in DSPs has also become a new requirement. Besides, due to the large amount of echo data and the demand of faster processing speed, the implementation on multicore DSP need to be considered.

The traditional single-core DSP chips process data serially, and the total processing time is the accumulation of multiple sub-tasks. For multicore DSP chips, the algorithm tasks are implemented in parallel in cores, which can greatly reduce time. However, due to the existence of multiple cores interactive communication and synchronisation, the total processing time is not calculated as single-core processing time divided by the number of cores. The actual time is related with the task allocation, data exchange, memory management and so on. It is also necessary to evaluate the control mechanism and data communication requirements between the cores.

The purpose of task allocation is to minimise the overhead of communication and synchronisation for multicore interactions and maximise the real-time processing performance. In addition, the speed of data access decreases step by step as the number of cache levels increases. The closer to the CPU, the faster the memory operation. The closer to the CPU, the smaller the storage space is, due to the higher cost. As a result, the memory space is limited and the data storage location along with the data flow model is very important.

Due to the fixed program algorithm flow and the variety of the data amount, we use the data flow processing model [1] to achieve the task allocation. The entire algorithm process is designed according to the C6678 chip characteristics, resource of the memory, and the data flow direction and memory configuration. Each core processes a part of the data in each subtask; however, multicore synchronise should be considered between the allocated subtasks.

The remaining content of this paper is as follows. Section 2 introduces the characteristics of MIMO radar echo signal and the imaging algorithm of MIMO Imaging radar. Section 3 introduces the design and implementation steps of MIMO imaging radar real-time processing system, including the hardware platform, task division, and the key technologies. Section 4 proves the software architecture by an experiment. Section 5 draws the conclusion.

2 Imaging algorithm of MIMO imaging radar

2.1 Characteristics of MIMO radar echo signal

The ground-based MIMO imaging radar geometry model is shown in Fig. 1. Assuming the MIMO array is collocated with M transmit elements and N receive elements and all T/R elements are on the same line. The multi-antenna array can be equivalent to a fixed large aperture, so the MIMO imaging radar has a superior azimuth resolution than real aperture radar. Each transmitting element transmits the frequency modulated continuous wave (FMCW), which is reflected back by many scattering points such as the Point P of the scene, and then is received by each receiving element. The collected echo data is mixed in the receiver to IF signal and then stored in the host computer.

The transmit signal model is:

$$S_{\mathrm{Tm}}(t) = p_m(t)\exp(j2\pi f_c t) \tag{1}$$

where $p_m(t)$ is the envelope of the transmit signal, f_c is the carrier frequency.

The distance between the m-th transmit element to the scatter point, and the scatter point to the n-th receive element is calculated as the follows:

$$\begin{aligned} R_{m,n} = &\sqrt{(r\sin\theta - t_r(m))^2 + (r\cos\theta)^2} \\ &+ \sqrt{(r\sin\theta - r_{\mathrm{e}}(n))^2 + (r\cos\theta)^2} \end{aligned} \tag{2}$$

Correspondingly, the delay between the m-th transmit element to the point target and then scattered to the n-th receiving element is as:

$$\tau_{m,n} = R_{m,n}/c \tag{3}$$

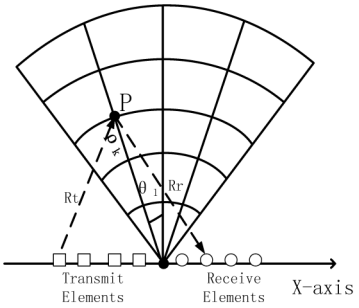

Fig. 1 *Geometry model of MIMO array*

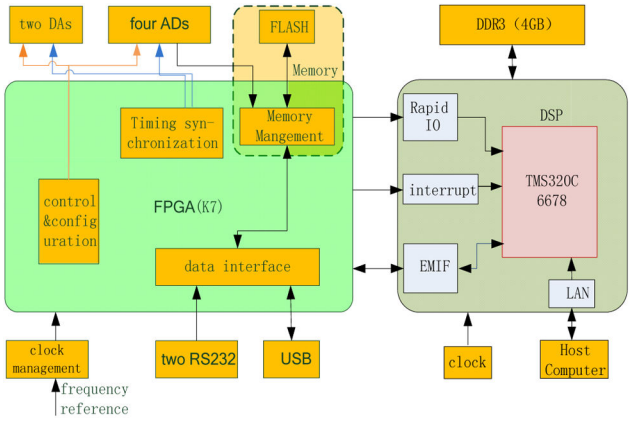

Fig. 2 *System hardware block diagram*

Without considering the influence of the antenna pattern and the receiver noise, the echo signal of the m-th transmit to n-th receive channel can be derived as:

$$S_{\mathrm{Rn}}(t)=\sum_{m=1}^{M}\sigma S_{\mathrm{Tm}}(t-\tau_{m,n})=\sum_{m=1}^{M}\sigma p_m(t-\tau_{m,n})\exp\left(j2\pi f_c(t-\tau_{m,n})\right) \tag{4}$$

After the received echo signal of each channel is mixed with the local oscillator signal, the echo signal expression of each separated channels of the MIMO radar is obtained:

$$S_{m,n}(t)=\sigma \mathrm{g}(t-\tau_{m,n})\exp(-j2\pi f_c\tau_{m,n}) \tag{5}$$

where σ is radar cross-section (RCS) of the target.

2.2 Back projection imaging algorithm

Since the back projection algorithm is not limited by the radar antenna array form, it becomes the preferred processing algorithm for MIMO radar imaging. Its basic idea is to mesh the imaging area, calculate the two-way delay between each point in the imaging area and transmit and receive elements of the antenna, and calculate the echo value corresponding to each grid point by interpolation. Then, traversing the imaging area of each pixel, you can achieve effective focus of the scattering points in the scene, and finally the scene imaging is achieved [2].

As shown in Fig. 1, There are $M\times N$ kinds of T/R antenna combination. Assuming the imaging region be divided into $K\times L$ pixel points, and the coordinate values of the pixel points in the distance and azimuth directions are represented by $\rho_k(k=1,2,\ldots K)$ and $\theta_l(l=1,2,\ldots L)$, the time delay from the point P to any combination of the transmitting and receiving antenna elements combination $(t_r(m), r_e(n))$ is calculated as:

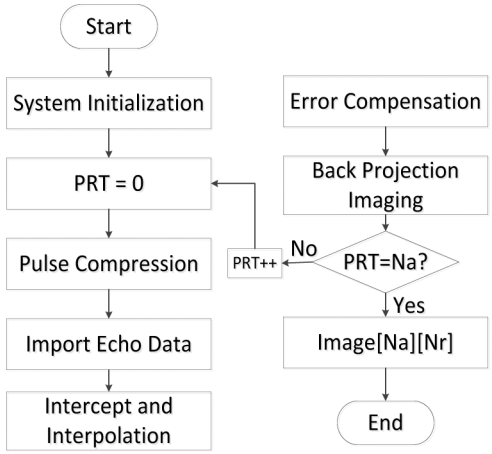

Fig. 3 *Imaging algorithm block diagram*

$$\tau_{(m,n)}^{(k,l)}=\frac{\sqrt{(\rho_k\sin\theta_l-t_r(m))^2+(\rho_k\cos\theta_l)^2}}{c}+\frac{\sqrt{(\rho_k\sin\theta_l-r_e(n))^2+(\rho_k\cos\theta_l)^2}}{c} \tag{6}$$

By focusing on each pixel in the scene area and substituting into (5), the complex scattering intensity of the point (ρ_k,θ_l) is:

$$I(\rho_k,\theta_l)=\sum_{m=1}^{M}\sum_{n=1}^{N}\sigma g\left(\tau_{(m,n)}^{(k,l)}-\tau_{(m,n)}^{(q,q)}\right)\times\exp\left(j2\pi f_c\left(\tau_{(m,n)}^{(k,l)}-\tau_{(m,n)}^{(q,q)}\right)\right) \tag{7}$$

where $\tau_{(m,n)}^{(q,q)}$ is the delay between the antennas and any target point.

3 Design of MIMO imaging radar real-time processing system based on C6678

3.1 Hardware platform introduction

The hardware structure of the signal processing board is shown in Fig. 2. It contains two DAs, four ADs, one Kintex-7 FPGA, and one TMS320C6678 chip. DSP is connected with 4 GB DDR3 and also connected with the host computer through the LAN port. The 16-bit EMIF interface is connected with FPGA, and the 4 × SRIO interface is connected directly with FPGA.

3.2 Task division and memory management

The overall imaging algorithm block diagram is shown in Fig. 3. According to the data stream input and output, the diagram can be divided into four subtasks, including range pulse compression, intercept and interpolation, error compensation, and back projection imaging. In addition, the subtasks are executed in sequence. The data flow output from the last subtask is used as the data flow input for the next subtask. According to the data amount, each subtask is divided into one-eighth to the eight cores to realise parallel processing. We need to focus on whether the eight cores are in the state of synchronisation before/after each subtask is processed, which ensures that the task flow and memory do not occur any unpredicted or unknown mistakes. In fact, the eight-core synchronisation debugging is the most time-consuming one of all steps in the entire software architecture design. Take range pulse compression for example, the eight-core task distribution and processing flow is as shown in Fig. 4. Before processing range pulse compression, the flags of each core need to be checked if they meet the conditions of synchronisation to carry out parallel processing. When the synchronisation is completed, carry on Fourier transform of range direction in eight cores. After that we need to execute eight-core synchronisation again to make the system in a steady state for the sake of system robustness.

Fig. 4 *Process steps of range pulse compression*

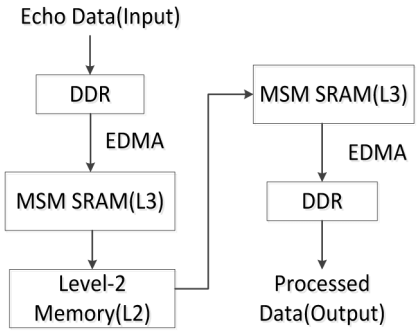

Fig. 5 *Overall data flow direction on cores*

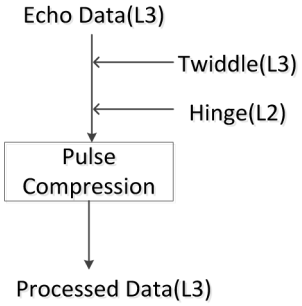

Fig. 6 *Data flow of pulse compression*

In general, the principle of resource allocation is: according to the entire algorithm of the data volume and data storage space, calculate the required size of each memory segment, as the basic foundation of configuring CMD file. 4096KB MSM SRAM Memory is shared by eight DSP C66x cores [3], so the data which is needed to be used by eight-core should be located at the MSM SRAM. I configure the MSM SRAM as Level-3(L3) Memory Mode. For the data which is purely used by its one alone core should be located at Level-2 (L2) memory so that the other cores do not have access to the data. Specific memory mapping method can be found from Multicore DSP Data Manual of TI. According to the data flow between the subtasks, the overall data flow diagram is as shown in Fig. 5.

Take pulse compression for example again, enhanced direct memory access (EDMA) helps move the echo data from DDR memory to L3 memory, which is ready for pulse compression [4]. As shown in Fig. 6, the twiddle factors are same for eight cores, located at L3 memory. The hinge factors are different for eight cores, located at L2 memory. Each core processes one-eighth of the data to proceed FFT, and put the processed data on L3. After that, make eight-core synchronisation preparing for the next subtask processing. As for the twiddle and hinge factor in Fig. 6, it will be introduced in the next chapter.

3.3 Key technologies of DSP imaging process

3.3.1 Big points FFT technology: FFT is the basic operation of signal processing. The commonly used signal processing chip will optimise the FFT operation and give the method or library function to realise FFT on the chip. For C6678 chip, TI official also provides real-time FFT operation of the library functions. Unfortunately, although C6678 has strong power of multicore processing, TI's official FFT method only supports single-core implementation. When the amount of data increases to some extent, it will get inefficient, time-consuming, which constraints real-time performance. In this context, the laboratory developed a correspondingly more efficient library function for FFT processing of very large points. The main idea is dividing the N points FFT into $N1 \times N2 (N1 \times N2 = N)$ two-dimensional matrix to achieve FFT,

Table 1 Ranks distribution table of big point FFT

Points	Rows($N1$)	Columns($N2$)
16 K	128	128
32 K	256	128
64 K	256	256
128 K	512	256
256 K	512	512

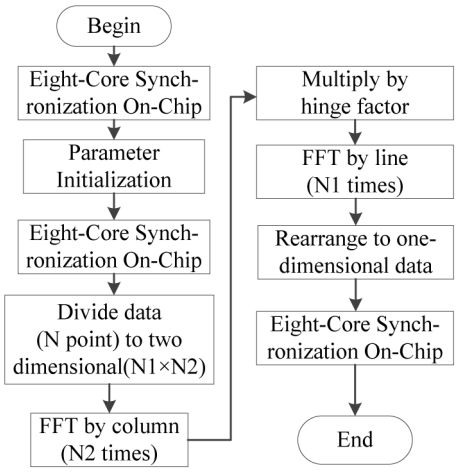

Fig. 7 *Flow chart of FFT process*

based on the definition of DFT. The data to be processed will change from one-dimension to two-dimension and finally go back to one-dimension in this process.

During the actual implementation, due to the eight-core participation, the data to be processed must be shared on L3 memory. For one single C6678 chip, it has 4 M of MSMC SRAM, which can process data of 512 K points. For engineer applications, due to some space occupied by interface buffers, code sections and so on, the chip can process maximum 256 K-point FFT. The ranks divide method as shown in Table 1.

After deciding the ranks number, divide the task. It will lead that the core operations are not synchronised if the division to eight-core is unreasonable. Part of the core is in idle state while the other part is busy with processing data, which did not take full advantage of C6678 multicore parallel processing. Therefore, divide the task in average to eight-core, and then the workload of each core is $N1/8$ and $N2/8$. After the eight-core synchronisation, complete the FFT calculation. FFT calculation process flow is as shown in Fig. 7. First, generate twiddle factor of $N1$ points and hinge factor of $N2$ points. Second, divide one-dimensional data to two-dimensional and proceed FFT by column and by line separately.

3.3.2 Multicore synchronisation and communication: TMS320C6678 has eight cores, and one of its significant advantages is the powerful parallel capability. Therefore, the key is multicore synchronisation and communication. For multicore parallel processing, the task progress is not easy to control. In general, the algorithm has a clear sequence of processing. Before the next step begins, the last step must be finished; otherwise, it will result in some unknown error. So, we need eight-core synchronisation to make the cores in the expected state to do things immediately. The most used three ways of synchronisation include semaphore synchronisation, shared memory synchronisation, and IPC interrupt synchronisation. In this paper, shared memory synchronisation is used.

The shared memory synchronisation set eight flags assigned to each core, and every core set/clear their own flag to detect the state of other cores' flag to achieve eight-core synchronisation. As the flag of shared memory area interacts frequently, we need to map the area to the non-Cache area, to avoid frequent maintenance operation of Cache, which affects real-time performance. The synchronisation process is as shown in Fig. 8.

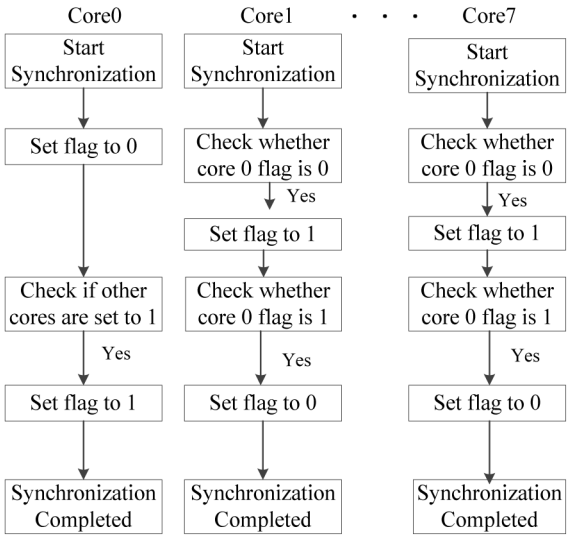

Fig. 8 *Flow chart of eight-core synchronisation*

Fig. 9 *MIMO Radar System for the experiment*

For the use of the synchronisation function, it is necessary to ensure that all eight cores are effectively synchronised over and then call this function for the second time after the synchronisation function is called for the first time. Otherwise, the same flag is used for the second time, which may be affected by the first time.

4 Experimental verification

4.1 Imaging process result

To verify the proposed analysis, an experiment is performed in this section. The experiment verification uses the echo data that is collected by the MIMO imaging radar system of the laboratory. The MIMO radar system contains 16 transmit elements and 16 receive elements for the experiment, as Fig. 9 shows. The scene optical image is shown in Fig. 10*a*. The imaging scene is the gymnasium of Beijing Institute of Technology. The radar was arranged at the top of the central building, overlooking the scene at a certain angle. The distance from the gymnasium to the radar is about 120∼250 metres, and the beam angle is about−45°∼45°.

Load the echo data into the C6678 memory to perform the imaging algorithm. Get the imaging data in DSP, then save the imaging data into a file of .dat format. Matlab is used to process the data imaging directly. The imaging result can be obtained as shown in Fig. 10*b*. We can find that the imaging result corresponds to the characteristics of the gymnasium.

4.2 Performance analysis

Performance analysis is evaluated mainly from two aspects: resolution and processing time. The actual resolution was calculated by selecting the corner reflector in the experimental scenes and compared with the theoretical value. The resolution and the peak side lobe ratio (PSLR) of the corner reflector are listed in Table 2, acceptable within the error range.

For the evaluation of the real-time performance, compare the DSP processing time with Matlab processing time, as shown in Table 3. We can find out that the processing time in DSP is much shorter than Matlab, which verifies that the real-time imaging processing software architecture is right and effective.

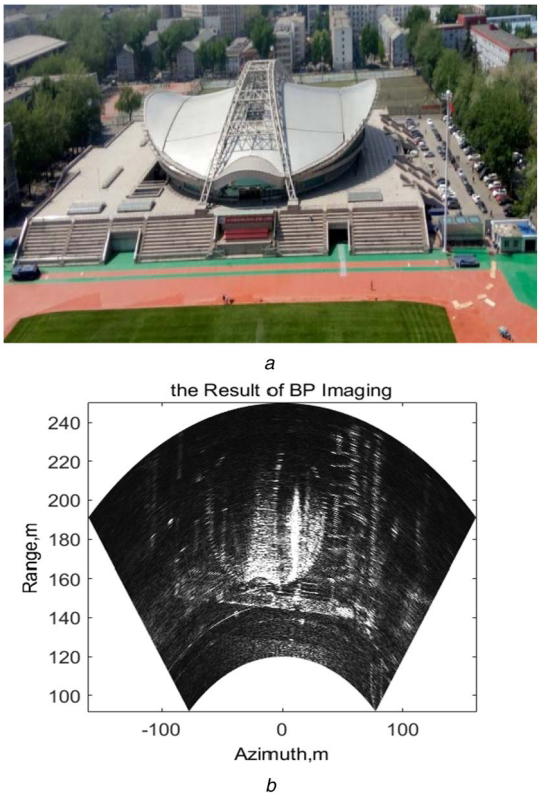

Fig. 10 *Comparison of experiment scene and imagining*
(a) Imaging scene of the experiment ***(b)*** Image process result of DSP

Table 2 Performance evaluation of the image

Parameters	Theoretical values	Practical values
range resolution	0.163 m	0.208 m
range PSLR	−13.26 dB	−9.38 dB
azimuth resolution	7.80 mrad	8.09 mrad
azimuth PSLR	−13.26 dB	−10.25 dB

Table 3 Real-time processing performance

Platform	The scene imaging range	Time cost of 256 PRT
DSP	120∼250 m	35.15 seconds
Matlab	120∼250 m	171.03 seconds

5 Conclusion

In this paper, the real-time image processing software architecture for MIMO imaging radar based on TMS320C6678 is designed and carried out. First, the back projection imaging algorithm of MIMO imaging radar is derived. Second, it is revealed that how to realise the algorithm on C6678 chip. The principle of task division and memory management is introduced and two key technologies of the process are proposed. Finally, the proposed architecture is verified by an experiment. This real-time imaging system can greatly improve the imaging efficiency and provide hardware assurance for rapid imaging of MIMO radar.

6 Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61601031, 61427802, 61625103, 31727901), Chang Jiang Scholars Program (Grant No. T2012122) and 111 Project of China (Grant No. B14010).

7 References

[1] Texas Instruments: '*Multicore programming guide*' (2012), USA
[2] Wang, H.: '*Imaging algorithm research of MIMO radar[D]*' (National University of Defense Technology, Changsha, 2010)

[3] Texas Instruments, Inc: '*Multicore fixed and floating-point digital signal processor*' Data Manual (2014), USA

[4] Dang, D.: '*The real-time imaging processing design of missile-borne SAR based on TI DSP C6678[D]*' (Xidian University, Xi'an, 2014)