A kind of millimeter wave signal processing system based on the multi-DSP architecture

Kai-Bo Cui, Xi Chen and Nai-Chang Yuan
Department of Electronic Science and Engineering National University of Defense, Technology, Changsha, Hunan, People's Republic of China
E-mail: 764608294@qq.com

With the rapid development of high-speed signal processing devices, the algorithm with large amounts of data can be implemented in real-time. This paper proposes a kind of signal processing system based on five pieces of multi-core DSP. Firstly, the interface circuit and program between FPGA and DSP, DSP and DSP, the system and computer are designed and tested. Then, this paper puts forward that the system can use Semaphore2 to schedule the task and use DDR3 as well as MSCM to share the data. Finally, the system is tested by using the imaging algorithm with a large amount of data. The time-consuming results of the system and imaging results demonstrate that this system is able to process the algorithm with a large amount of data in real-time.

Keywords: Multi-Core DSP; TMS320C6678; SRIO; HyperLink; Real-Time; Image System.

1. Introduction

In recent years, with the development of microelectronics technology, high-speed signal processing devices are being quickly updated: the speed of clock management module can be up to 600MHz in FPGA (Field Programmable Gata Array) which has more than 100 million logic elements (LE) and its speed of serial connection can be up to 12.5Gbps. DSP (Digital Signal Processor) has progressed to the multi-core stage. Its highest clock frequency of single-core can be up to 1.25GHz [3] and the clock frequency of multi-core can be up to 10GHz. The speed of the high-speed data memory access device also has reached 2ns or less. It is possible that the algorithm with large amounts of data can be implemented in real-time with the rapid development of these devices, such as high-resolution imaging algorithms, image processing algorithms and so on [10][11]. Especially for the millimeter wave radar, this king of radar has a large amount of data to be processed and the traditional signal processing system can't

meet the requirements. This paper presents a signal processing system based on the multi-chip multi-core DSP architecture [12]. This system is composed of five pieces of TMS320C6678 DSP, which can realize the work of communication, synchronization and real-time processing for the algorithm with large amounts of data.

2. The Processing System Based on Multi-chirp DSP

The entire signal processing system is shown as Fig. 1. The system is made of five pieces of DSP and a piece of FPGA. The FPGA is interconnected with DSP1, DSP2 and DSP5 through SRIO (Serial RapidIO). DSP1 and DSP3, DSP2 and DSP4 are interconnected through HyperLink respectively. DSP5 owns a network port, so the system can communicate with the computer via Ethernet.

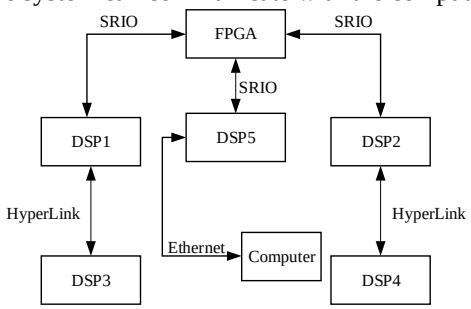


Fig. 1 The block diagram of the system

2.1. *Design and implementation of SRIO module*

SRIO [4] is a non-proprietary, high-bandwidth, system-level interconnect. In the signal processing system proposed by this paper, the DSP chip and FPGA chip are directly connected via SRIO which is configured as 4x mode. The DSP chip and FPGA chip are initialized firstly after the parameters have been set. Then the physical layer of them shake hands through exchanging the symbols. After the handshake, the output signal which is called “port initialized” of the FPGA chip’s IP core will be pulled up. In the same time, the port register bits which is called “Port OK” of the DSP chip will be pulled up too. Finally, the two devices can communicate normally. In order to harmonize the devices, the master device will send an interrupt to the other device after it sending a packet. The received device response to the interrupt firstly, then the received data will be processed in the interrupt function. Finally the interrupt will be cleared in order to response to the next interrupt. The flow chart of SRIO communication between the two devices is shown as Fig. 2.

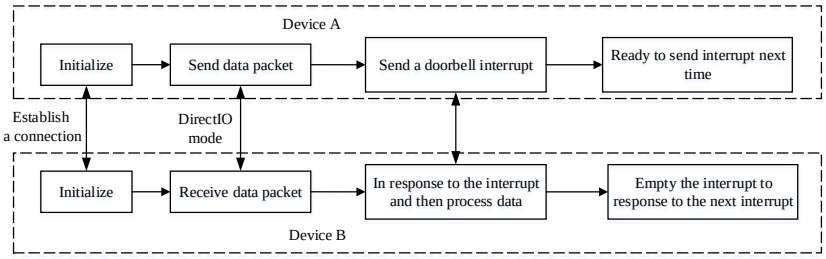


Fig. 2The flow chart of SRIO communication

2.2. *Design and implementation of HyperLink module*

HyperLink [6] provides a high-speed, low-latency, and low-pin-count communication interface that extends the internal CBA 3.x-based transactions between two C66x. In the system, the two DSP chips are connected used four-channel mode and the transmission rate of per channel is 3.125Gbps. Firstly, the two devices are initialized and mapped space address is established. In this way, the device can carry out normal communication. The two devices can send and receive data through coordinating the internal software interrupt in HyperLink. The local device sends a software interrupt firstly, and then it send a batch of data. After the interrupt is triggered, the remote device can receive data in the interrupt function. When it complete receiving the data, the interrupt will be cleared up to wait for the next interrupt. In order to ensure every batch of data is received accurately, we add a flag artificially when the data is transmitted. The communication process of the Hyperlink module between the two devices is shown in Fig. 3.

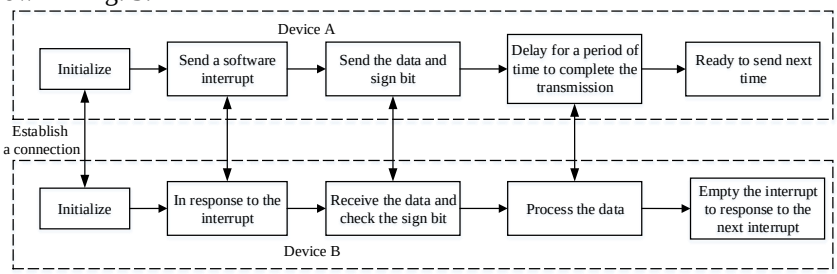


Fig. 3 The flow chart of the Hyperlink module communication

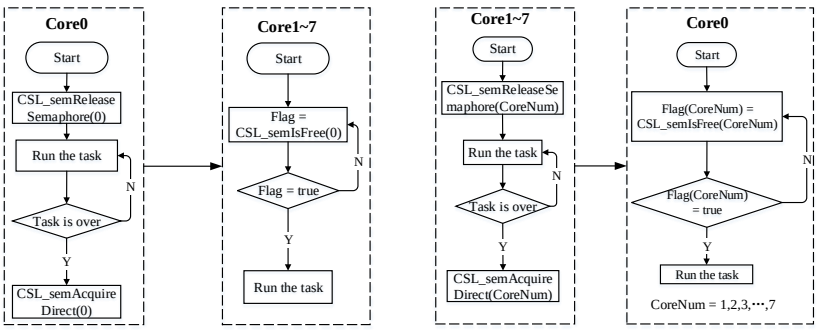
3. The Synchronization and Scheduling of Multi-core DSP

The signal processing system consists of five TMS320C6678 which is a high-performance 8-cores DSP launched by TI company with the newest KeyStone architecture [3]. The system uses Semaphore2 module to synchronize the multi-

core tasks. Core 0 is used as a control core and data core. Core 1 to core 7 are used as data cores.

Fig. 4-(a) is the schematic diagram of core 0 synchronizing with the other cores. Core 1 ~ 7 first wait for the flag whose initial value is "false". The program waits until the flag is set to "true", and then the program begins normal operation. The flag value is generated by the state of core 0's Seamphore2. Core 0 first uses "CSL_semReleaseSemaphore (0)" to ensure the semaphore 0 has been released. Then core 0 will acquire the semaphore for direct access after the completion of a specific task by using "CSL_semReleaseSemaphore (0)". Core 1 ~ 7 always check if the specified semaphore is acquired or not through "CSL_semIsFree (0)". If the semaphore has been obtained, it returns true, otherwise it returns false. So after the completion of a specific task, core 1 ~ 7 will be to the next move.

Similarly, core 1 ~ 7 synchronization with core 0 also performed as described above and the synchronization diagram is as Figure 4-(b). "CoreNum" is the number of the core.



(a) Core 0 synchronize with the other cores (b) Core 1~7 synchronize with core 0

Fig. 4 Multicore synchronization schematic

There are 8G Bytes DDR3 [5] that can be visited by each core in each DSP. Although its access rate is only 1/3 of the core frequency, but through EDMA3 communications, data can be moved to MSCM or RAM firstly and then be processed. Therefore, large-capacity DDR3 is suitable for the algorithm with large amounts of data. The multi-core of TMS320C6678 share a 4M Bits RAM, which can be configured to be L2RAM or L3RAM. This means that the data moved by EDMA3 from DDR3 can be put in MSCM and each core can directly process it. MSCM can also store some universal parameters and data in arithmetic processing. The data sharing in DDR3 and MSCM is shown as Fig. 5.

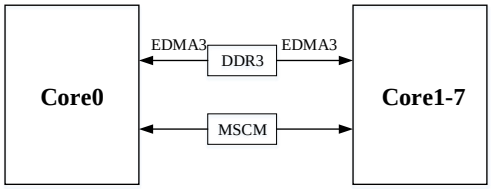


Fig. 5 The data sharing in DDR3 and MSCM

In conclusion, we use Semaphore to schedule the task and use DDR3 and MSCM to share the data.

4. Algorithm Test

For the signal processing system, we use the imaging algorithm with a large amount of data to test [8][9][10]. This paper ignores the imaging theory and tests the system directly.

The data flow is shown as Figure 6. FPGA received the data which is processed through AD and then transferred it to DSP through SRIO with Ping-Pong manner. The odd block data was transferred to DSP1 and the even block data was transferred to DSP2. DSP1 received the first odd block data and then transferred it to DSP3. DSP3 sent it back to DSP1 after processing and then DSP1 sent the data to the FPGA. DSP2 received the first even block and then transferred it to DSP4. DSP4 sent it back to DSP2 after processing and then DSP2 sent the data to the FPGA. DSP1 received the second odd block of data and then sent it back to the FPGA after processing. DSP2 received the second even block of data and then sent it back to the FPGA after processing. FPGA sent all results to DSP5 through SRIO and then DSP5 sent the data to a computer.

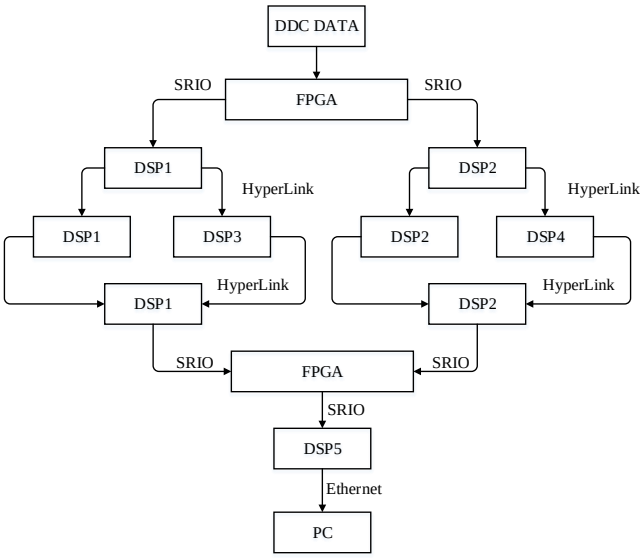


Fig. 6 The data flow of the algorithm

Firstly the imaging parameters and window function are stored in MSCM. Core0 is responsible for receiving the DDC data from the FPGA through SRIO and stores it in DDR3 and then segments it in the azimuth dimension. When the received data reaches a certain number, Core0 will release a Semaphore signal. Core1 ~ 7 once detect it, they begin to move the data from DDR3 for the Doppler centroid estimation, range compression and range migration correction. The Doppler centroid will be stored in MSCM. Upon completion of the above task, core1 ~ 7 will release Semaphore signals separately. When core0 has detected all the signals, it starts to compensate the Doppler center for every block of data and then releases the Semaphore signal. When core1 ~ 7 detect all the signals, they will begin to estimate Doppler frequency of every block of data by the distance dimension and the value of frequency will be stored in MSCM. Then core1 ~ 7 release Semaphore signals separately. When detecting all the signals, core0 will get the value of Doppler frequency from MSCM and obtain the motion error parameters which will be stored in the MSCM and then release Semaphore signal. When core1 ~ 7 detect all the signals, they begin to compensate the motion error and release the Semaphore signals separately. When detecting all the signals, core0 will integrate the data by the azimuth dimension and release the Semaphore signal. Once core1 ~ 7 detect all the signals, they will perform the matched filtering in the azimuth dimension and more visual processing and then obtain the maximum values of each block of

data which will be stored in the MSCM. Finally they will release the Semaphore signals. When detecting all the signals, core0 will obtain the maximum values of each block of data from MSCM to calculate the maximum value of the whole image and then store it in the MSCM. Finally, core0 will release the Semaphore signal. Once core1 ~ 7 detect the signal, they will quantize the results data and then release the Semaphore signal. When detecting all the signals, core0 will send the results to the FPGA. The flow of the entire algorithm running on the system is shown as Fig. 7.

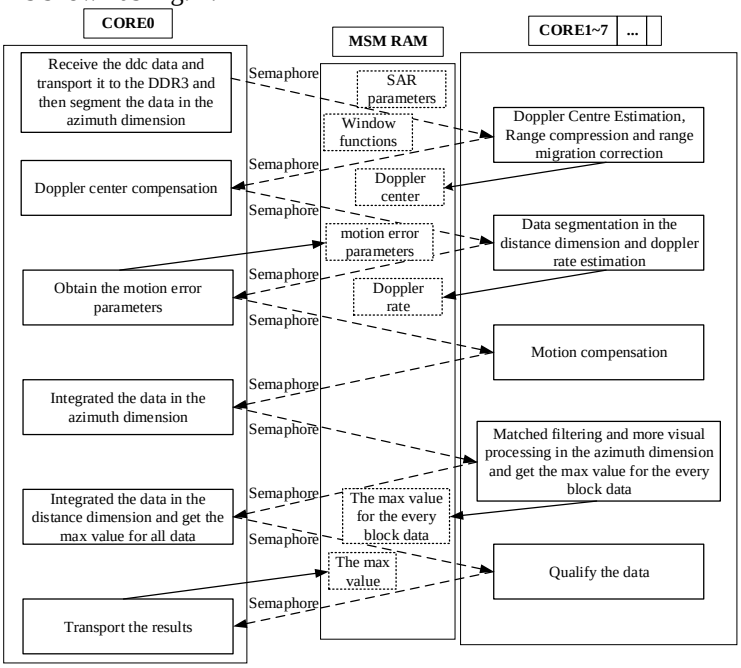


Fig. 7 The flow of the algorithm running on the system

According to the above, we test the time-consuming of the system and the test results are shown in table 1. Among them, the form of the number of data is “the point of the distance dimension” * “the point of the azimuth dimension”. The main frequency of DSP is 1GHz, that is: its period is 1ns. The number of more visual processing is 16. For the data whose size is 4096 * 4096, the running time of the entire imaging algorithm is 3784ms. This system uses 4 pieces of DSP to process the data by Ping-Pong, which provides enough resources for real-time processing. As can be seen from the results, as long as the PRF the system is 3.784K or less, it can meet the requirements of real-time processing.

Tab. 1 The time-consuming of the algorithm

Algorithm step	Data points	Cycle	Time(ms)
Doppler center estimation	4096*512	6110074	6.2
Range compression	4096*512	138342046	138
Range migration correction	4096*512	62825070	62
Doppler rate estimation	512*4096	141484866	141
Obtain the motion error parameters	512*4096	544226	0.5
Matched filtering	512*4096	2064973614	2064
More visual processing	512*4096	149043509	149
Get the max value	512*4096	138776637	139
Qualify the data	512*4096	1056379	1.1
The whole algorithm	4096*4096	3673710876	3784

The image result is shown as Fig. 8. As can be seen from the figure, this signal processing system can perform real-time imaging algorithm with large amounts of data.

Fig. 8 The image result of the algorithm

5. Conclusion

It is possible that the algorithm with large amounts of data can be implemented in real-time with the rapid development of high-speed signal processing devices. This paper proposes a signal processing system which uses a piece of FPGA and five pieces of DSP. DSP uses TI's latest multi-core product: TMS320C6678. In this paper, we designed the interface circuit and program between FPGA and DSP, DSP and DSP, the system and computer and tested all the links. This paper also researched how to synchronize and schedule the multi-core tasks. The system can use Semaphore to schedule the task and use DDR3 and MSCM to share the data. Finally, the system was tested using the imaging algorithm with a large amount of data. We designed the algorithm flow and task allocation and then verified the design with the measured data. The time-consuming results of

the system and imaging results demonstrated that this system was able to process the algorithm with a large amount of data in real-time. The results also explain that this system has a more broad application prospects in many fields.

References

1. TEXAS INSTRUMENTS. KeyStone Architecture Network Coprocessor (NETCP) User Guide, http://www.ti.com.
2. I. G. Cumming and H. W. Frank. Digital Processing of Synthetic Aperture Radar Data: Algorithms and Implementation. Bei Jing: Publishing House of Electronics Industry, 2007.
3. TEXAS INSTRUMENTS. TMS320C6678 (Multicore Fixed and Floating-Point Digital Signal Processor) Data Manual, http://www.ti.com.
4. TEXAS INSTRUMENTS. Keystone Architecture Serial RapidIO (SRIO) User Guide, http://www.ti.com.
5. TEXAS INSTRUMENTS. Keystone Architecture Enhanced Direct Memory Access (EDMA3) Controller User Guide, http://www.ti.com.
6. TEXAS INSTRUMENTS. Keystone Architecture HyperLink User Guide, http://www.ti.com.
7. TEXAS INSTRUMENTS. TI Network Developer's Kit (NDK) v2.21 User's Guide, http://www.ti.com.
8. J. Y. Wei, Y. T. Ye, Y. F. Wu and J. J. Deng. Multi-DSP Real Time Image Processing System Based on Rapid IO Protocol, 2008 International Conference.
9. S. Y. Peng. Research on Key Technologies of Missile-borne Synthetic Aperture Radar Imaging, Ph.D School of National University of Defense Technology, 2011.
10. J. H. Duan, Y. L. Deng and G. Kun. Development of Image Processing System Based on DSP and FPGA. The Eigth International Coference on Electronic Measurement and Instruments, 2007.
11. Z. H. Zhan, W. Hao, Y. Tian, D. W. Yao and X. H. Wang. A Design of Versatile Image Processing Platform Based on the Dual Multi-core DSP and FPGA. The Fifth International Symposium on Computational Intelligence and Design, 2012.
12. J. Battle, J. Marti, and P. Ridao. A New FPGA/DSP-Based Parallel Architecture for Real-time Processing. Real-Time Imaging, 2002.