Design and Implementation of Lossless Compression System for CCSDS Hyperspectral Images

Lili Zhang1, Tao Liu2, Kechun Zhang3
1Electronics and Information Engineering, Shenyang Aerospace University, Shenyang, Liaoning, 110135, China
2Electronics and Information Engineering, Shenyang Aerospace University, Shenyang, Liaoning, 110135, China
3Electronics and Information Engineering, Shenyang Aerospace University, Shenyang, Liaoning, 110135, China
Email: 20052727@sau.edu.cn

Abstract: Based on the xc7k325tffg900 FPGA chip of Xilinx Company and combining the actual hyperspectral image compression task requirements, the calculating speed and compression efficiency of the algorithm module are optimized. In the design of the prediction module in this article, the two-clock solution is used to solve the problem that the algorithm speed is restricted due to the feedback of the weight update. And the RTL-level design is implemented on the FPGA. CCSDS 123.0-B-1 hyperspectral image lossless compression algorithm is achieved. The system uses block compression to deal with the data source to avoid the entire system compression data errors caused by a certain pixel compression error. So the system's Error Resilience ability is improved. Finally the compressed code stream is decoded. The decompressed image is consistent with the raw one, which verifies the correctness of this design.

1. Introduction
The load data characteristics of high-speed multichannel require higher and higher spatial data communication rates. The ground receiving system needs to be able to receive and process Multi-payload satellite data in real time and accurately, which has become the key to the problem[1]. Therefore, it is necessary to compress image data. Due to the limited resources of space equipment, the compression algorithm is required to have the characteristics of high compression ratio, high stability and low complexity[2]. CCSDS (Spatial Data System Advisory Committee) released in May 2012 the first standard CCSDS123.0-B-1 specifically for lossless compression of satellite-borne multispectral and hyperspectral images[3]. This standard is a lossless compression algorithm based on 3-D space adaptive linear prediction[4]. The algorithm has low complexity and high compression rate, and has become an international standard for lossless compression of satellite-borne multispectral and hyperspectral images[5].

In this article, the algorithm based on the Xilinx 7 Series FPGA chip is implemented, and the calculation speed of the algorithm, the error resilience rate of the system, the compression ratio and the resource utilization rate are optimized.
2. Analysis of CCSDS hyperspectral image compression algorithm

The algorithm is composed of two parts: a predictor and an encoder. The CCSDS compression algorithm model is shown in Figure 1. The predictor adopts the method of 3-D space adaptive linear prediction to decorrelate the image data, and the adaptive Rice encoding module is responsible for compressing and encoding the decorrelated data.

![Figure 1. CCSDS compression algorithm model.](image)

2.1. Design of prediction algorithm

Hyperspectral image data is shown in Figure 2. The prediction process can be performed on any receiving channel of the image. Generally, using the values nearby current sampling point and the values of $P(0 \leq P \leq 15)$ pieces of adjacent spectral bands, the predicted value and the mapping value of the predicted value at the sampling point $S_{x,y,z}$ are calculated.

![Figure 2. Hyperspectral image data.](image)

In each spectral band, the predictor calculates the local sum of neighbor-oriented sample values. The local sum is used to calculate the local differences. Within each spectrum, the central local differences are equal to the difference between the local sum and 4 times the current sample values. The calculation of the local differences in the spectrum is shown in Figure 3. The local differences in the three directions are equal to the difference between the local sum and 4 times $N$, $W$, $NW$. The weight used in the calculation process will be adaptively updated according to the calculation result of each predicted sample value. Each predicted residual error, that is, the difference between the current sample value and the corresponding predicted value, will be mapped to an unsigned integer.

The weight coefficient is needed to calculate the predicted value. After the predicted value is obtained, it is needed to use the predicted value to update the weight coefficient to calculate the next predicted value. When this algorithm structure is mapped to hardware implementation, it will inevitably produce a feedback structure. The feedback structure will affect the speed of algorithm implementation on hardware, which is also a major factor restricting hardware performance. Since the update of the weight of each pixel value in the algorithm needs to use the predicted value of the current pixel, the parallel method cannot be used to improve the efficiency of the weight update in FPGA implementation. In this article, a two-clock solution is adopted to solve the feedback problem in the system. The clock is divided into fast clock and slow clock. The fast clock is used to update the weight coefficient. The slow clock is used on the encoding module.

2.2. Design of encoding algorithm

The low-bit data of the image data after prediction is similar to noise, and encoding these data will cause encoding expansion. The solution is to divide the encoder into two parts: $K$ value selection and encoding. $K$ means that no encoding is performed from the lowest bit to $K$ bit, and Rice encoding is performed on data higher than $K$ bit. The formulas of counter and accumulator used to calculate the $K$ value are as formula (1) and formula (2). $t$ represents the pixel position currently calculated, and...
\( \Gamma_z(t) \) represents the value of the corresponding counter. The counting range of the counter is \([0, 2^{\gamma} - 1]\). \( \sum_z(t) \) represents the value of the accumulator and \( \delta_z(t) \) represents the mapping value of the predicted value. If the counter and the accumulator are allowed to increase indefinitely, with the continuous input of image data, the weight will be too high when calculating the average value, so the effect of calculating the average value is reduced by continuously dividing by 2. When formula (3) is established, \( k_z(t) = 0 \), otherwise, it needs to calculate \( k_z(t) \) according to formula (4). The selection of 49/128 is conformed by CCSDS through experimental comparison and testing. The calculation method of \( k_z(t) \) is, if \( 2^{k_z(t)} \) is between 16 and 32, then the value of \( k_z(t) \) is 4.

\[
\Gamma_z(t) \begin{cases} 
\Gamma_z(t) = \Gamma_z(t-1) + 1, & \Gamma(t-1) < 2^{\gamma} - 1 \\
\frac{\Gamma_z(t-1) + 1}{2}, & \Gamma(t-1) = 2^{\gamma} - 1
\end{cases}
\]

\[
\sum_z(t) = \begin{cases} 
\sum_z(t-1) + \delta_z(t-1), & \Gamma(t-1) < 2^{\gamma} - 1 \\
\frac{\sum_z(t-1) + \delta_z(t-1) + 1}{2}, & \Gamma(t-1) = 2^{\gamma} - 1
\end{cases}
\]

\[
\Gamma(t) > \sum_z(t) + \left[ \frac{49\Gamma(t)}{2^{\gamma}} \right]
\]

\[
\Gamma(t) \cdot 2^{k_z(t)} \leq \sum_z(t) + \left[ \frac{49\Gamma(t)}{2^{\gamma}} \right]
\]

Golomb-rice encoding is performed after \( k_z(t) \) is determined. When \( u_z(t) < u_{max} \), the calculation method of \( u_z(t) \) is formula (5), in which, \( \delta_z(t) \) represents the mapping value of the predicted value.

\[
u_z(t) = \left\lfloor \frac{\delta_z(t)}{2^{k_z(t)}} \right\rfloor
\]

3. Design and implementation of FPGA

The data needs to be cached through external memory DDR3, because the storage format of the Hyperspectral image used in this article is BIL (band interleaved by line), and the amount of data is large. The system mainly includes DDR3 control module, control module, prediction module and encoding module.

The main function of the DDR3 control module is to control the reading and writing of external memory. The image data sent in from the outside is written into the memory in time, and the data is read in segments and sent to the subsequent modules for compression.

The control module mainly controls the entire system, mainly including the start and end of compression and configuration parameters. At the same time, another important function of the control module is to compress the image data in blocks within the compressor.

The main work of the prediction module is to calculate the local sum and differences and the predicted value. The predicted value is mapped and the weight value is constantly updated.

The main function of the encoding module is to calculate the K value and adaptively encodes the predicted value.

3.1. Implementation of the prediction module

Considering the speed of FPGA implementation and the occupation of logic resources, the data of the current spectral band and the first four spectral bands are selected for compression. The main function of this module is to buffer the input image data to obtain synchronously output data of 5 consecutive spectrum segments, and each spectrum segment includes the data of current line and the previous line. The external memory provides image data with a width of 32 bits. The high 16 bits are the image data of the previous line, and the low 16 bits are the data of current line. The cache module uses 5 FIFO to
buffer data to realize the synchronization of the image data output of 5 spectrum bands. After buffered, the output data are processed with a delay of two clock cycles using the register. The data delayed by one clock cycle in the current line is the data central that needs to be predicted. The data delayed by two clock cycles is W. The data delayed by one clock cycle in the previous line is N. The data delayed by two clock cycles in the previous line is NW. The data that has not yet entered the register is NE.

3.2. Implementation of the coding module
The encoding module is divided into two sub-modules: K value calculation module and code stream splice module. The K value calculation module determines the K value. The splice module generates the code, and splices the variable-length code to output a fixed 16-bit width code stream.

3.2.1 K value calculation module. The block diagram of the K value calculation module is shown in Figure 4. Adaptive coding selects the K value through an accumulator and a counter, and updates it automatically during the coding process.

3.2.2. Coding splice module. The block diagram of the coding splice module is shown in Figure 5. If it is directly coded after the K value is obtained, there may be a large number of consecutive ‘0’ code streams, which affects the compression efficiency of the algorithm. So a comparator is added in the design. By comparing the size between the highest bit to the K bit of the mapping value and the threshold value (the threshold value is selected as 64 in this article), only the mapping values lied on the highest bit to the K bit which are less than the threshold value are encoded. When they are greater than the threshold value, they are not encoded, and the raw data are directly written into the encoding.

3.3. Design of control module
The finite state machine is in the idle state. When full ("full is high" means the data buffer area of the compressor is full) is equal to 0, the finite state machine jumps to the wr_data state and requests data from the outside until the compressor cannot receive new data. The compressor designed in this article will be initialized once after compressing the 32 lines of data of all the spectrum bands, which can effectively avoid incorrect transmission. When clear is equal to 1, it means that the compressor is initialized and the finite state machine jumps to the idle state again.

4. Experimental test
The design implementation structure diagram is shown in Figure 6. The entire compressor is mainly composed of five modules: compress_top, vec_top, predict, k_calculate and encoding. The control module is inside the compression top module. The cache module and the local sum and difference computation module are inside the vec_top module.
As shown in Table 1. The resources of LUT, BUFG and FF occupy less than 10%, and the resources of LUTRAM, BRAM and DSP occupy less than 5%. In summary, the FPGA chip selected in this article can not only implement the CCSDS algorithm, but also occupies very little resources. In actual projects, it is possible to use the multicore parallel solution to obtain higher data throughput.

| Resource | Utilization/Available | Utilization/|
|----------|------------------------|-------------|
| LUT      | 14724/203800           | 7.22        |
| FF       | 34473/407600           | 8.46        |
| LUTRAM   | 66/64000               | 0.10        |
| BRAM     | 12.50/445              | 2.81        |
| DSP      | 14/840                 | 1.67        |

5. Conclusion

Based on the CCSDS 123.0-B-1 issued by the Consultative Committee for Spatial Data Systems (CCSDS), the CCSDS Hyperspectral image compression system is designed and implemented in this article. The prediction module, encoding module and control module are designed and implemented on FPGA board. When designing, the actual application scenarios and algorithm principles are considered. Reasonable improvements and optimizations are made to the traditional design. The speed and error resilience of the entire system are improved. Through a large amount of data simulation test and hardware verification and the comparison of image test data and the raw data, the reliability and correctness of the system can be proved. In terms of performance, it can meet most of the mission requirements of spaceborne hyperspectral image compression.

References

[1] Liu, X.X, Dong, Z.X, Zhu, Y. (2017) FPGA design of real-time satellite data processing based on CCSDS standard [J]. Microelectronics and computers. 34(08): 52-56+61.

[2] Sun, J.W, Zhang, Z.W, Zheng, T, Xue, C.B, Chen, C. (2019) Design of CCSDS space-borne data lossless compression system based on FPGA [J]. Chinese journal of space science. 39(05): 694-700.
[3] CCSDS. CCSDS123. 0-B-1. (2012)Lossless Multispectral & Hyperspectral Image Compression[S]. Washington DC: CCSDS Secretariat.

[4] ARANKI N, BAKHSHI A, KEYMEULEN D, et al. (2009)Fast and adaptive lossless on-board hyperspectral data compression system for space applications[c]. In : IEEE Aerospace conference, New York. 1-8.

[5] Yin, B, Pei, J.J, Meng, G.Y, Kuang, J.J, Wang, H.F, Wu, M.Y. (2016)FPGA Implementation of CCSDS Multi-spectral image lossless compression algorithm [J]. Journal of wuhan university (science edition). 62(06): 531-536.

[6] Gong, Z.M, Yang, J.X, Li, S.W. (2015)FPGA implementation of parallel frame synchronization system based on CCSDS standard [J]. Microelectronics and computers. 32(07): 82-85.

[7] Zhao, W.J, Wang, Z. (2008)Telemetry synchronization and channel coding system and FPGA implementation in accordance with CCSDS standard [J]. Electronic Design Engineering. 26(01): 89-92+97.