Korean Journal of Remote Sensing, Vol.18, No.3, 2002, pp.147~153

Software Buffering Technique For Real-time Recording of High Speed Satellite Data

Dong-Seok Shin*, Wook-Hyun Choi*, Moon-Gyu Kim, and Won-Kyu Park***

Image/System Division, SaTReC Initiative Co. Ltd.*

Remote Sensing Division, Satellite Technology Research Center, KAIST**

Abstract : The real-time reception and recording of down-link mission data from a satellite requires the highest reliability because the data lost in receiving process cannot be recovered. The data receiving and recording system has moved from a set of dedicated hardware and software components to commercial-off-the-shelf (COTS) components in order to reduce the system cost as well as to upgrade the system easily for handling other satellite data. The use of COTS hardware and middleware components prevents the system developer from correcting or modifying the internal operations of the COTS components, and hence, instant performance degradation of the COTS components which affects the reliable data acquisition must be covered by a software algorithm. This paper introduces the instant performance problem of a COTS data recording device which leads to the data loss in the real-time data reception and recording process. As a result, the requirement of the modification of the conventional data read/write technique is issued. In order to overcome the data loss problem due to the use of COTS components and the conventional software technique, a new algorithm called a software buffering technique is proposed. The experiments show that the application of the proposed technique results in reliable real-time reception and recording of high speed serial data.

Key Words : Software Buffering, Data Reception and Recording.

1. Introduction

As the spatial resolution of spaceborne Earth observation payloads becomes finer, the rate of the real-time payload data transmission has increased rapidly. The KOMPSAT-2 MSC data, for example, is designed to be transmitted to a ground station at the rate of 320Mbps (KARI, 2000). The transmission rates of the spaceborne payloads such as high-resolution optical cameras, synthetic aperture radars (SAR) and hyper-spectral sensors are expected to be a level of 500Mbps in the near future.

The real-time data receiving and recording system in a ground station must also be upgraded in order to cope with the increasing down-link data rate. In the very early stage of the real-time data recording system development, the received and demodulated serial digital data were recorded to a dedicated digital tape such as a high-density digital tape (HDDT) (EOC, 1994). The usage of HDDT and the corresponding tape

Received 27 April 2002; Accepted 2 June 2002.

drive, HDDR (High Density Digital Recorder), showed several drawbacks. Firstly, they were not compatible with computers, and therefore, additional interface hardware is required for the playback and processing of the recorded data. In addition, the HDDR equipment was very costly due to its dedicated usage for high-rate data recording only. From early 1990, the real-time satellite data recording system began to adopt a technique which ingested the received serial data to a commercial-off-the-shelf(COTS) computer and recorded the data to a computer compatible storage equipment, such as a hard disk drive or redundant array of independent disks (RAID) (SPOTIMAGE, 1997).

Since the payload data transmission system on-board the satellite is a integrated and dedicated hardware module, its data handling speed is generally higher than the corresponding data receiving and recording system in a ground station which uses COTS components. The COTS components include the operating system (OS) of the data receiving computer, an I/O host bus adapter such as a SCSI controller, device drivers as well as a RAID controller. It is very difficult to modify the internal functions and capabilities of the COTS components, and hence, any temporary malfunctioning or performance degradation of a COTS component cannot be prevented by a receiving and recording system developer. In practice, the data writing to a RAID system shows periodic and non-periodic speed degradations which result in the loss of receiving data (NLANR, 1998).

This paper describes a technique, called a software buffering technique, which enables reliable data recording in the case of instant speed degradation of COTS components. Section 2 describes the overall architecture and operation of the real-time satellite data receiving and recording system. The problem of the previous operation procedure is mentioned in Section 3. The software buffering technique is described in Section 4 in details. The experimental verifications of the software buffering technique are shown in Section 5.

2. Data Receiving and Recording Process

Fig. 1. shows a simplified view of the satellite data receiving and recording system (Kim *et al.*, 2001). The down-link data are demodulated and bit-synchronized in the demodulator. The resultant serial data and synchronized clock signal is ingested to the data receiving card (DRC). DRC converts the serial data to a parallel form, and stores them in a internal memory (FIFO) temporarily. When the DRC FIFO is filled with a certain amount of data, DRC generates an interrupt to the host computer. The software in the host computer

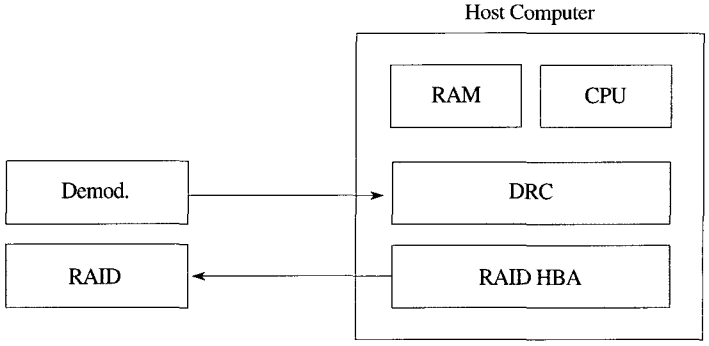

Fig. 1. System Hardware Components.

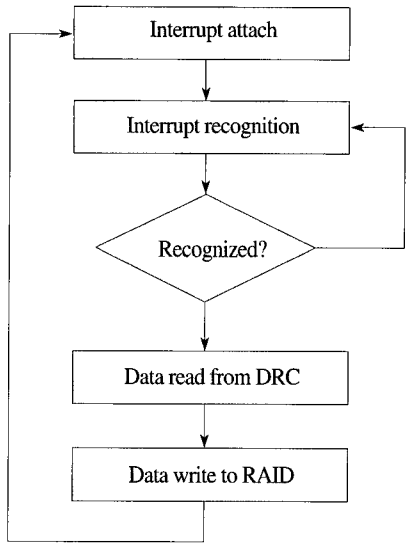

Fig. 2. Sequential data read and write flow.

recognizes the interrupt from DRC, transfers data from DRC FIFO to the main memory, and finally sends the data to RAID through the RAID host bus adapter (HBA).

In summary, the operation process of the data receiving and recording procedure consists of three sequential steps.

- interrupt recognition
- data read from DRC FIFO
- data write to RAID

Fig. 2 shows the operation flow of the sequential data read and write algorithm which is iterated until the end of data acquisition.

3. Problem Description

The problem of the sequential data read and write process described in the previous section arises from the two factors.

- limited size of DRC FIFO
- instant speed degradation of RAID

The commercially available and commonly used FIFO chips are provided with the size of up to 512KB currently. The physical size limitation of the host adaptor card, DRC, also limits the maximum size of the internal FIFO. In practice, four FIFO chips which are connected in a cascaded manner are integrated in DRC

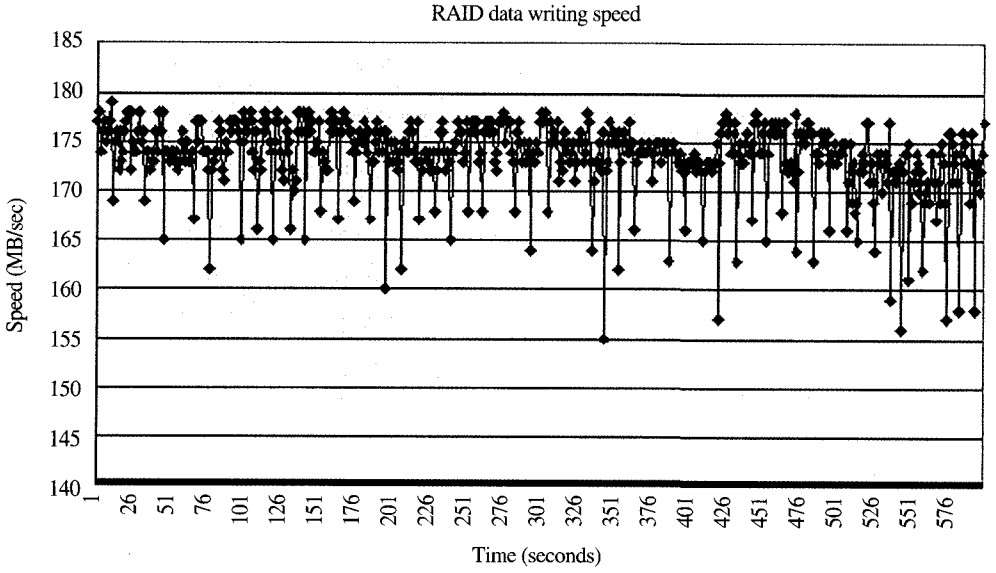

Fig. 3. RAID data writing speed example.

currently providing 2MB in total. Since the input data is ingested continuously into the DRC FIFO even during the data read process from the host software, it is required to preserve some data space in FIFO during the data read and write process of the host software. In this sense, the host interrupt is generated when the half of the total FIFO space is filled up.

Due to the limited size of DRC FIFO the data is lost if the two successive data read steps do not occur frequently enough. In the case of 300Mbps input data rate and a total of 2MB DRC FIFO size, for example, the time for filling a half of FIFO up takes 26.7 milliseconds. In other words, if the time between the two successive data read steps is larger than 27 milliseconds, the FIFO is filled up and the incoming data is lost.

Although the average speed for the interrupt recognition and the data read/write procedures is high enough to meet the performance requirement of the real-time data recording process, the data writing process shows instant and frequent speed degradation as shown in Fig. 3. This instant speed degradation may be caused by internal operations of the operating system, the RAID host bus adaptor or the RAID controller. Whatever the reason of the instant speed degradation of data writing process is, the problem cannot be solved easily because all problematic candidates are COTS components.

As described above, the instant speed degradation of the COTS components in data writing process causes data loss, and consequently unreliable data receiving and recording capability.

Several solutions on this problem can be addressed. For example, a number of FIFO chips are integrated into DRC so that the incoming data are stored continuously in DRC FIFO even when the instant speed degradation occurs. The RAID with much higher performance can be adopted so that even the instantly degraded data writing speed satisfies the performance requirement of the system. In this paper, the simplest, the least costly and the most feasible solution, named as software buffering technique, is proposed.

4. Software Buffering Technique

The concept of the software buffering technique is simple. The software uses a large amount of host main memory, say ~50MB, as a buffer. The memory buffer eventually plays the same role as the DRC FIFO does.

This software buffering technique can be achieved by using two different threads which run concurrently (the read thread and the write thread).

The conceptual diagram of the software buffering is shown in Fig. 4. As soon as the read thread recognizes the first interrupt from DRC, it reads the data from DRC FIFO and stores them to the memory buffer at the count 0. The write thread starts to write the data from the buffer count 0 while the read thread returns directly to the next interrupt recognition. When the read thread detects the next interrupt from DRC, it checks whether the current write thread has completed its writing operation. If completed, the read thread puts the next data from DRC FIFO to the buffer count 0. Otherwise, it puts the data to the buffer at the next count, i.e. count 1.

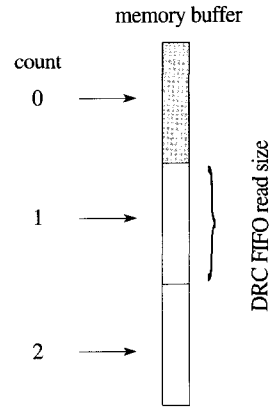

Fig. 4. Memory buffering sequence.

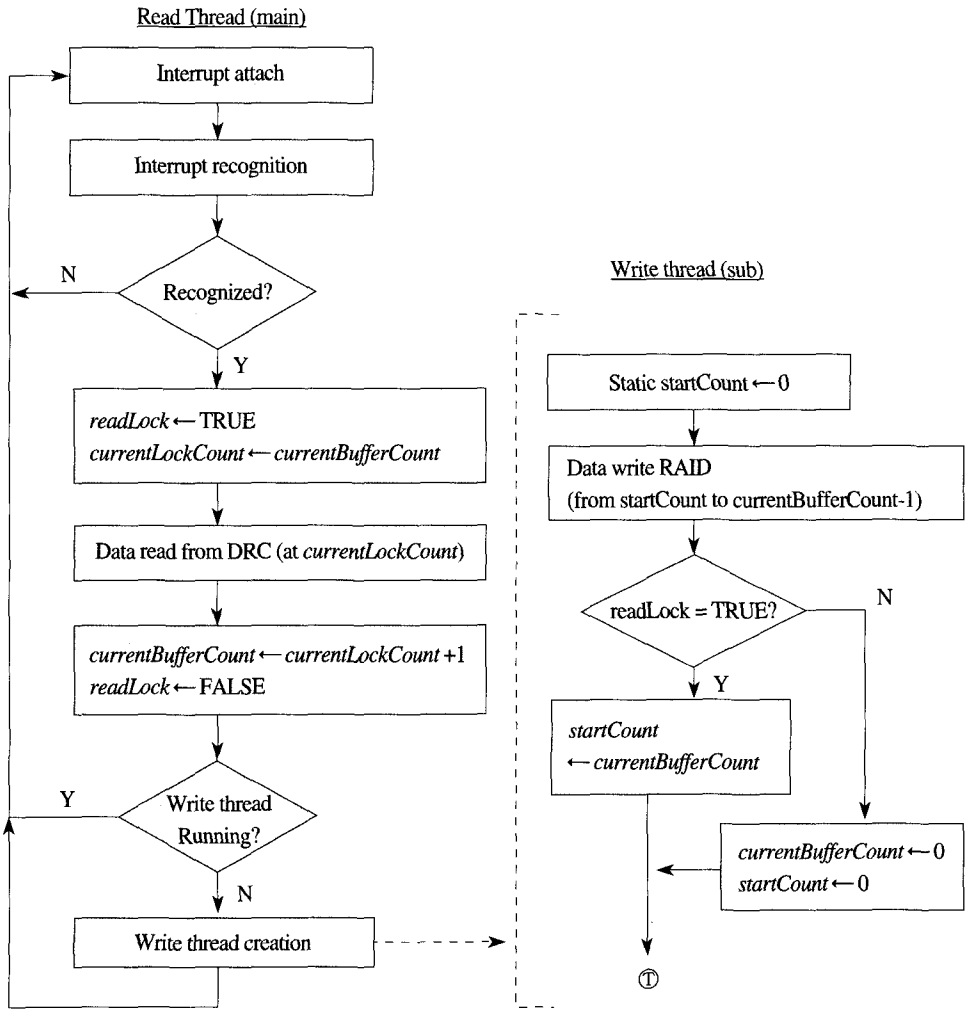

Fig. 5. Software buffering algorithm.

The write thread writes the whole data in the buffer to RAID. This procedure prevents the incoming data from being lost due to the instant writing speed degradation because the incoming data is temporarily stored at the large host memory buffer while the writing operation is being carried out. As long as the average data read/write speed is higher than the incoming data rate, this software buffering technique provides reliable real-time data acquisition process.

The detailed flow of the proposed software buffering algorithm is shown in Fig. 5. The key issue of the algorithm is the use of semaphore which locks and unlocks the current buffer count value, so that the two concurrently running threads can be synchronized perfectly.

5. Experiments

The experiments were carried out for showing the

data acquisition reliability of the proposed software buffering algorithm as well as the conventional sequential data read/write algorithm. The real-time data transmission was simulated by using a data generator, Tektronix DG2040, which can generates the ECL level serial data up to 1Gbps. The transmitted data was ingested to DRC with 2MB internal FIFO, and the software with a configurable buffering technique was used for reading and writing the data from DRC to RAID. A PentiumIII 1GHz Windows 2000 was used as a host computer. A Netcom LX5000 RAID with five data striping disks and U160 SCSI connection was used as a data recording device.

A data pattern was programmed to the data generator in order to detect the amount of data loss during the data receiving and recording process. The pattern consists of consecutive packets and each packet consists of a packet synchronization block, an incremental packet counter block and a dummy data block, which results in 256 bytes in total.

The purpose of the experiments is not to quantify the data loss due to system noise (e.g. bit error rate) but the data loss due to the instant overflow of the system. Therefore, the amount of the lost data can be calculated by replaying the recorded data and counting the missing packet counter.

Table 1 shows the results of the experiments. Up to the input data rate of 260Mbps, both techniques recorded the data without any lost packets. This means that the level of instant performance degradation is small enough for using DRC FIFO buffer. Both techniques resulted in packet loss when the data rate is higher than 360Mbps. In other words, the average (not instant) speed of the read/write procedure cannot catch up with the input data rate in spite of using the software buffering technique.

Table 1. Number of lost packets.

data rate (Mbps)	size (GB)	# of lost packets	
		sequential	buffering
260	3.8	0	0
280	4.1	14	0
300	4.4	91	0
320	4.7	33	0
340	5.0	12	0
360	5.2	3027	1285

However, the conventional technique showed packet loss errors on the data rate range of 280Mbps to 340Mbps while the proposed software buffering technique recorded the input data without any data loss. The packet loss of the conventional technique is due to the instant performance degradation of COTS components. The results show that the periodicity and the level of the instant degradation is not regular. The proposed technique provided the reliable data acquisition process by buffering the input data in a large amount of host memory.

6. Conclusions and Discussions

In this paper, the software buffering technique was proposed in order for achieving reliable operation of real-time high speed satellite data. The problems of using COTS components and the sequential read/write technique were described. It was shown that the proposed technique solved the data loss problem due to the instant speed degradation of the COTS components.

In this paper, only data read and write operations were described. When the real-time processing is required during the data reception and recording operation, a modified algorithm which is more complicated than the algorithm shown in Fig. 5 must be proposed. The real-time processing is highly required when the system receives the image data from a satellite because the system provides, in general, a real-time moving window display (MWD) function. In this sense, a new software thread called a processing thread must be defined in addition to the read and the write threads.

The relationship of these three threads must be defined clearly in order to achieve reliable data archiving as well as graphical display of the image being received.

References

EOC, 1994. JERS-1 Data Users Handboook, Earth Observation Center, National Space Development Agency of Japan.

KARI, 2000. KOMPSAT-2 System Requirements Review, Korea Aerospace Research Institute.

Kim, M.G., T. Kim, S.O. Park, D. Shin, M.N. Hong, S. Kwak, W. Choi, Nov 2001. Automated Image Reception, Processing and Distribution System For High Resolution Remote Sensing Satellites, Proceedings on Asian Conference on Remote Sensing, Singapore.

NLANR, 1998, Hard Disk Continuous-Write Measurements, http://moat.nlanr.net/Dskwtst/

SPOTIMAGE, Feb 1997. SPOT Receiving Stations, SOSS XI Meeting, Maspalomas, Spain.