International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 8, Issue 9 (September 2013), PP. 42-51

Real Time Visualization of Full Resolution Data of Indian Remote Sensing Satellite

Ch Umakanth[1], Deepika Roy[2], M. Manju Sarma[3], B.Lakshmi[4]
National Remote Sensing Centre, Indian Space Research Organization, India.

ABSTRACT

As part of the Indian Space program, ISRO has been developing Indian Remote Sensing (IRS) satellites and deploying the satellite data processing software at various data reception stations. Starting from the first satellite IRS-1A, all IRS series of satellites, have been provided with a sub-sampled mode of image display at all IRS data reception stations. Initially the image extraction from the baseband data and the sub-sampled display was realized through a customized hardware processor. Later on the total real time quick look data processing and display of sub-sampled image data functionality has been shifted to software. With a need for variation in the scope for the application like only data acquisition, data acquisition & quick look display at full resolution /sub-sampled with quick geometric and radiometric correction, deployment based application re-configurability has become a requirement. This paper presents the design and implementation of a real-time application for IRS satellite data acquisition, archival, preprocessing and visualization of full resolution image data for Resourcesat-2 satellite by implementing task parallelism on multi-CPU multi core platform architecture which can be reconfigured for selected functions based on the need and platform being used.

Keywords:- Real time, Multi-spectral, Parallel process, Partitioning, Communication, Agglomeration, Mapping

I. INTRODUCTION

Traditionally the quick visualization of raw image data acquired by Indian Remote Sensing (IRS) satellites is done in a sub-sampled mode. All IRS series of satellites, starting from the first satellite IRS-1A (with a baseband downlink data rate of 10Mbits/sec), are provided with this feature. With the changing technology, the design & implementation schemes used for realizing the quick look visualization got enhanced. The image extraction from the baseband data and the sub-sampled display was realized through a customized hardware processor until early 1990. Since 1995, the real time quicklook processing and display of sub-sampled image data function is implemented in software. With Resourcesat-1 (2003) the data rates are increased to 105 x 2 Mbits/sec and the complexity of processing is increased as the onboard data is RS encoded and compressed, requiring additional processing to process the downlink data. With the increasing demand for real-time visualizations, the scope of real-time display function got redefined, requiring preprocessed full resolution display. The availability of multi-core and multi CPU platforms, the computing power and other resources have increased enhancing the capability to handle more processing functions in real-time. This enabled realization of pre-processed full resolution quick look displays even for satellites involving high input data rates and complex preprocessing.

This paper presents the design and implementation of a real-time application for satellite data acquisition, archival, preprocessing and visualization of image data at full resolution for all the three imaging sensors of Resourcesat-2 (RS-2) satellite by implementing task parallelism on multi-CPU multi core platform architecture [1].

Resourcesat-2 satellite is an Indian remote sensing satellite with three payloads, namely LISS-III, AWiFS and LISS-IV, onboard capable of providing multi-spectral imagery with different resolutions for various applications. RS-2 satellite data is down-linked in two streams of each 105 Mbits/sec. One stream contains two sensors data namely LISS-III and AWiFS and other stream contains LISS-IV data. Onboard the sensor's data is RS encoded to recover data from transmission errors, compressed using Differential Pulse Code Modulation (DPCM) and Multi Linear Gain (MLG) to reduce the downlink rate to 210 Mbits/sec. The application acquires the downlink data in real-time through the data acquisition card interfaced to the computer through PCI interface, archives the data onto high performance disk array after channel wise segregation. Each stream data is processed to extract and deliver the image to the display processes after appropriate segmentation as per the display resolution and number of display monitors for visualization. The processing involves de-randomization, channel separation, RS-decoding, decompression of data and conversion to 8 bit data before delivering to display monitors. The application is implemented on general purpose blade server which is a 6 core 4 CPU system with 64 GB RAM, using Linux operating system achieving the deterministic response by controlling the system environment by allowing only the required system services and using non-degradable priorities for process scheduling.[2]

II. DESCRIPTION OF THE REAL-TIME VISUALIZATION APPLICATION

The application carries out multiple tasks. They are acquisition of each stream data from hardware device, writing data to disk, stream processing based on the sensors and display. The application is scalable and can handle multiple streams and multiple sensors and multiple display processes based on the display resolution. This being a real time application, completion of the tasks within stipulated time is of more concern than minimizing/avoiding the processor idle time. Based on the requirements certain tasks are more critical than others.

III. PARALLEL PROCESS DESIGN

PCAM (Partitioning, Communication, Agglomeration, Mapping) design methodology is adopted for design [3][4] to realize the program. The scalability to execute on higher or lower configurations is based on plugging or unplugging the processes which in turn will increase or decrease the functional scope of the software.

A. Partitioning

In the Partitioning stage of design, various opportunities for parallel execution are identified. When the computation is decomposed into smaller tasks it is known as ‘Functional decomposition’ and if the data operated on by the computation is decomposed into smaller partitions it is known as ‘Data decomposition’.
In this application, Functional decomposition partitioning technique is used to divide the computation into independent tasks and then the data requirements of the identified tasks are examined. The identified tasks are the parallel tasks sharing the common data.
Fig. 1 below shows the partitioning of the application into different task/processes.

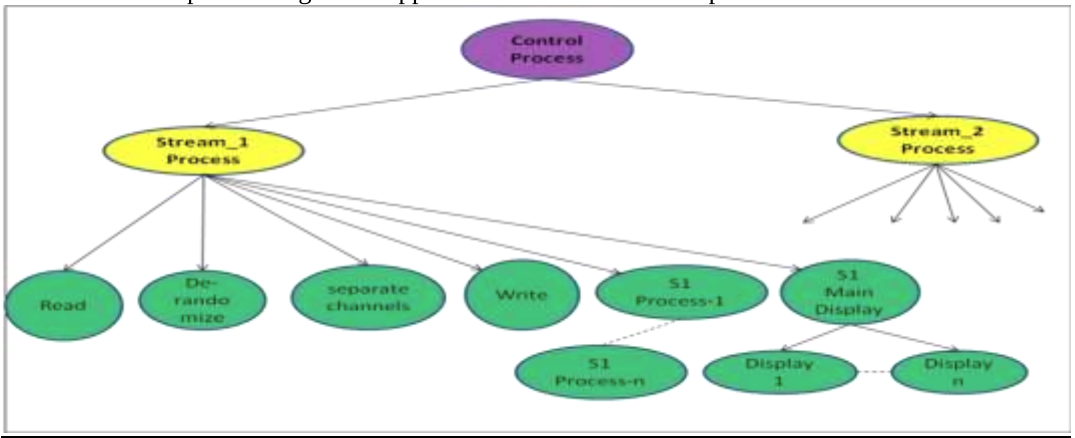

Fig. 1: Partitioning of the real time visualization application

A brief description of the tasks identified is as follows
Control process – This is the main process which creates and manages stream-wise processes involved in the application based on the number of streams of satellite downlink.
Stream-wise processes (Stream_1 process & Stream_2 process) – These are the processes created by the ‘control process’ for each of the streams namely LISS-IV stream and LISS-III stream. The Stream-wise processes create, control and manage the required processes for sensor specific processing.
Each stream-wise process needs to create the following processes.
Read process - This process acquires the data from data acquisition front end hardware (FEH), which is interfaced to the system through PCI, which in turn is interfaced with the data reception chain to receive the data.
De-randomization - The satellite data is randomized and de-randomization is the first processing to be done. It involves carrying out an exclusive OR operation on the entire input bit stream using a 127 bit pseudorandom sequence. The pseudorandom sequence is a 2048 byte sequence for LISS-III & AWiFS data and 1024 byte sequence for LISS-IV data.
Channel separation – The next process involved is Channel separation. Each stream data consists of two channels I & Q and it comes as interleaved I+Q data. Here in this process I+Q data is separated into I & Q channels.

Write process - After 'Channel separation' process, this process writes the de-randomized, channel separated data on to disk.

S1 Stream (LISS – III Stream) Processes - LISS-III Stream is S1 Stream and consists of LISS-III and AWiFS sensors data. The following are the different processes of LISS-III stream.

S1 Process-1 (LISS-III process) – RS Decoding

Process-1 of S1 stream is RS decoding of LISS-III data. In this process, RS encoded LISS-III data consisting of 86 RS blocks is decoded using RS(255, 247) code.

S1 Process-2 (LISS-III process) - Inverse DPCM

The compression scheme used for LISS-III sensor is Differential Pulse Code Modulation (DPCM). This process of S1 stream performs decompression of the RS decoded LISS-III data by applying Inverse DPCM. LISS-III payload gives data in 4 Bands (Band 2, Band 3, Band 4 and Band 5) which is received on the ground in an interleaved format. The first step of decompression is separating this band interleaved data into individual bands' data. Inverse DPCM is then applied on each individual band data. This involves taking a block of 4 pixels at a time consisting of 28 bit stream. The reference pixel is first reconstructed by repacking the most significant bit of each of the other 3 neighboring pixels. The other pixels are reconstructed by applying reverse look up table on their sign corrected value and then adding/subtracting this value from neighboring pixel values to get the final pixel value. After applying Inverse DPCM on a block of pixels, they are rearranged in proper order for display.

S1 Process-3 (LISS-III process) - Radiometric correction

Radiometric correction of LISS-III data is Process-3 of S1 Stream. After inverse DPCM, radiometric correction is applied to that LISS-III data by using look-up table.

S1 Process-4 (LISS-III process) - 10bit to 8 bit mapping

The radio-metrically corrected LISS-III data is a 10 bit data. In this process, the 10bit data is mapped to 8 bit for display purpose.

S1 Process-5 (AWiFS process) - RS decoding

Process-5 of S1 stream is RS decoding of Awifs data. In this process, RS encoded AWiFS data consisting of 256 RS blocks is decoded using RS(255, 247) code.

S1 Process-6 (AWiFS process) – Inverse Multi Linear Gain

The compression scheme used for AWiFS sensor is Multi Linear Gain (MLG). The S1 Process-6 performs the decompression of raw data by applying Inverse MLG. AWiFS sensor has two detectors: Detector A and Detector B, each consisting of 6000 pixels and 4 bands, Band 2, Band 3, Band 4 and Band 5. AWiFS payload gives 12 bit data per pixel which is compressed and transmitted as 10 bit data. On ground, band wise interleaved data in 10 bit format is received. In this process, first band wise data separation is done. Then each pixel value is decompressed from 10 bit to 12 bit by applying Inverse MLG. Inverse MLG is applied based on the window where the pixel value falls.

S1 Process-7 (AWiFS process) - Radiometric correction

Process-7 of S1 Stream is radiometric correction of the AWiFS data. After inverse MLG, radiometric correction is applied to the AWiFS data by using look-up table.

S1 Process-8 (AWiFS process) - 12bit to 8 bit mapping

The radio-metrically corrected AWiFS data is a 12 bit data. In this process, the 12 bit data is mapped to 8 bit for display purpose.

S2 Stream (LISS-IV Stream) Processes

The S2 Stream is LISS-IV Stream and it consists of only LISS-IV sensor data. The following are the different processes of LISS-IV stream.

S2 Process-1 (LISS-IV process) - RS decoding

Process-1 of S2 stream is RS decoding of LISS-IV data. In this process, RS encoded LISS-IV data consisting of 26 RS blocks is decoded using RS(255, 247) code.

S2 Process-2 (LISS-IV process) – Inverse DPCM

LISS-IV payload gives 10 bit data per pixel which is compressed onboard to 7 bits using Differential Pulse Code Modulation. The S2 Process-2 performs decompression of the raw data to 10 bit data by applying Inverse DPCM. Initially for Short MX mode, band data separation is also done. Inverse DPCM involves taking a block of 4 pixels at a time consisting of 28 bit stream. The reference pixel is first reconstructed by most significant bit repacking from the other 3 pixels in the block. The other pixels are reconstructed by applying reverse look up table on their sign corrected value and then adding/subtracting this value from other neighboring pixel values to get the final pixel value. After applying Inverse DPCM on a block of pixels, they are rearranged in proper order for display.

S2 Process-3 (LISS-IV process) – Stagger & Radiometric correction

Stagger & Radiometric correction of the LISS-IV data is Process-3 of S2 Stream. In LISS-IV sensor there is a stagger between the odd and even numbered pixels. Thus after inverse DPCM stagger correction is applied to get properly aligned decompressed image. Then radiometric correction is applied to the LISS-IV data by using look-up table.

S2 Process-4 (LISS-IV process) – 10 bit to 8 bit conversion

The radio-metrically corrected LISS-IV data is a 10 bit data. In this process, the 10bit data is mapped to 8 bit for display purpose.

Display Initiator process (S1 Main Display & S2 Main Display)

The display processes are created and controlled by this process. Each of the display processes displays a segment of the total pixels. The number of display processes to be created is sensor specific. The necessary information such as the segment of the processed data to be displayed, the display terminal on which the segment is to be displayed is provided by this process to the display processes.

Display Processes

The display processes create the necessary display environment on the display monitors, like creating the display window on the monitor and setting the display depth. Each display process is responsible for handling the display on one monitor. These processes wait for the availability of processed payload data in the shared memory for display. Once the processed data is available, the image is exported and scrolled on the remote monitors. The number of display processes created, N, depends on the number of pixels in one line of image of the sensor and display monitor resolution. The display processes take the corresponding processed data sets from the shared memory and send them to respective display monitor systems over the network. The image is then displayed on the display monitors. Synchronization of the scrolling of images on all the monitors simultaneously is achieved with the aid of semaphores.

The number of pixels in one line of image for LISS-IV sensor is 12000, for LISS-III 6000 and for AWiFS 12000. In the current configuration, where the display monitors have a resolution of 1920*1200 pixels, the number of display processes to be created and thus the number of display monitors configured for display are 4 for LISS-III, 7 for LISS-IV and 7 for AWiFS.

B. Communication

The tasks resulting from partitioning are intended for parallel execution with dependencies. The computation to be done by one task requires data associated with another task. Data must be shared between tasks so as to allow computation to proceed. This information flow is specified in the communication phase of design. First a channel structure that links tasks that require data with tasks that possess those data is defined. Second, the messages that are to be sent and received on these channels are specified. Communication requirements in parallel algorithms obtained by functional decomposition correspond to the data flow between tasks.

Shared memory programming model is used to share a common data, which they read and write asynchronously. Various mechanisms such as locks and semaphores may be used to control access to the shared memory. An advantage of this model from the programmer's point of view is that the notion of data “ownership" is lacking, and hence there is no need to specify explicitly the communication of data from producers to consumers. This model can simplify program development.

In this application, for S1 stream, the ‘de-randomize’ process requires data from ‘read’ process. The output data of ‘de-randomize’ process is required by ‘channel separate’ process. The ‘write’ process requires data from ‘channel separate’ process. S1_Process-1 requires data from ‘channel separate’ process and so on. Similarly, individual display processes S1_Display-1 to S1_Display-n require data from S1 Main Display process. Similar data requirements exist for S2 stream processes.

Using the Shared memory programming model, a shared memory data structure is to be defined for communication of the data between the different processes.

Specifically the communication structures that are to be defined between the processes of LISS-III Stream for LISS-III sensor processing & display are as follows.

(1) ‘read’ and ‘de-randomize’ processes,
(2) ‘de-randomize’ and ‘channel separate’ processes,
(3) ‘channel separate’ and ‘write’ processes,
(4) ‘channel separate’ and ‘LISS-III RS Decoding’ processes,
(5) ‘LISS-III RS Decoding’ and ‘LISS-III Inverse DPCM’ processes,
(6) ‘LISS-III Inverse DPCM’ and ‘LISS-III Radiometric correction’ processes,
(7) ‘LISS-III Radiometric correction’ and ‘LISS-III 10 to 8 bit mapping’ processes,
(8) ‘LISS-III 10 to 8 bit mapping’ and ‘LISS-III Main Display processes’ processes,
(9) ‘LISS-III Main Display process’ and ‘LISS-III Display-1 to LISS-III Display-4’ processes.

The above communication requirements for LISS-III sensor data processing & display, through shared memory, are as depicted in Fig. 2.

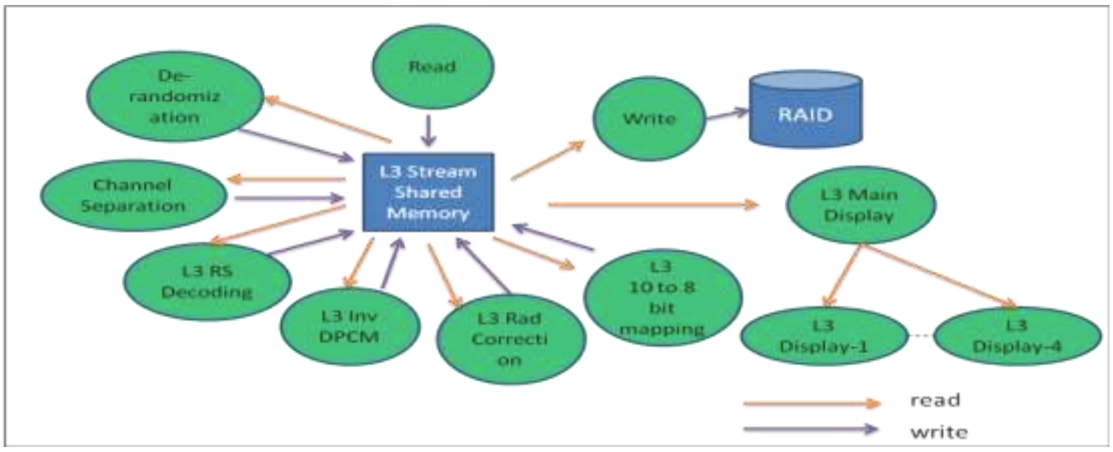

Fig. 2: Communication Requirements of LISS-III Stream (LISS-III Sensor)

The communication structures that are to be defined between the AWiFS processes of LISS-III Stream are as follows
(1) ‘channel separate’ and ‘AWiFS RS Decoding’ processes,
(2) ‘AWiFS RS Decoding’ and ‘AWiFS Inverse MLG’ processes,
(3) ‘AWiFS Inverse MLG’ and ‘AWiFS Radiometric correction’ processes,
(4) ‘AWiFS Radiometric correction’ and ‘AWiFS 12 to 8 bit mapping’ processes,
(5) ‘AWiFS 12 to 8 bit mapping’ and ‘AWiFS Main Display’ processes,
(6) ‘AWiFS Main Display’ and ‘AWiFS Display-1 to Display-7’ processes.

The above communication requirements for AWiFS sensor data processing & display, through shared memory, are as depicted in Fig. 3

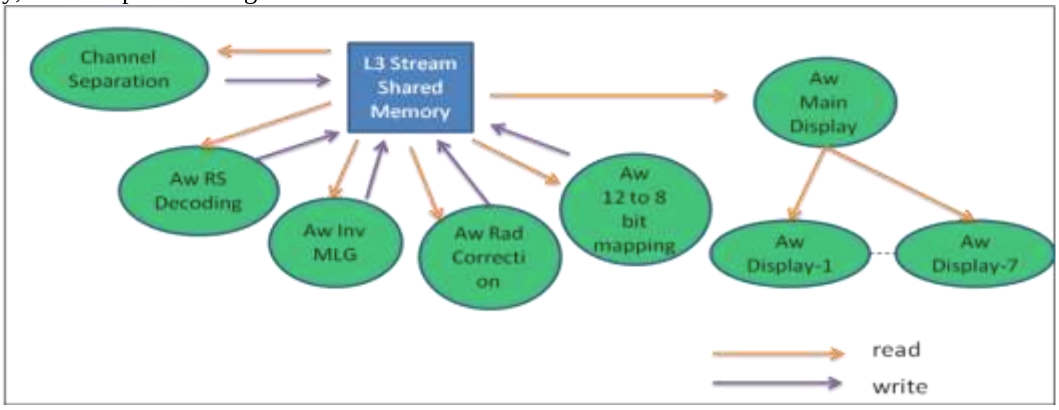

Fig. 3: Communication Requirements of LISS-III Stream (AWiFS Sensor)

Similarly, the communication structures that are to be defined between the processes of LISS-IV Stream for LISS-IV sensor data processing & display are as follows.
(1) ‘read’ and ‘de-randomize’ processes,
(2) ‘de-randomize’ and ‘channel separate’ processes,
(3) ‘channel separate’ and ‘write’ processes,
(4) ‘channel separate’ and ‘LISS-IV RS Decoding’ processes,
(5) ‘LISS-IV RS Decoding’ and ‘LISS-IV Inverse DPCM’ processes,
(6) ‘LISS-IV Inverse DPCM’ and ‘LISS-IV Stagger & Radiometric correction’ processes,
(7) ‘LISS-IV Stagger & Radiometric correction’ and ‘LISS-IV 10 to 8 bit mapping’ processes,
(8) ‘LISS-IV 10 to 8 bit mapping’ and ‘LISS-IV Main Display’ processes,
(9) ‘LISS-IV Main Display’ and ‘LISS-IV Display-1 to LISS-IV Display-7’ processes.

The above communication requirements for LISS-IV sensor data processing & display are as depicted in Fig. 4

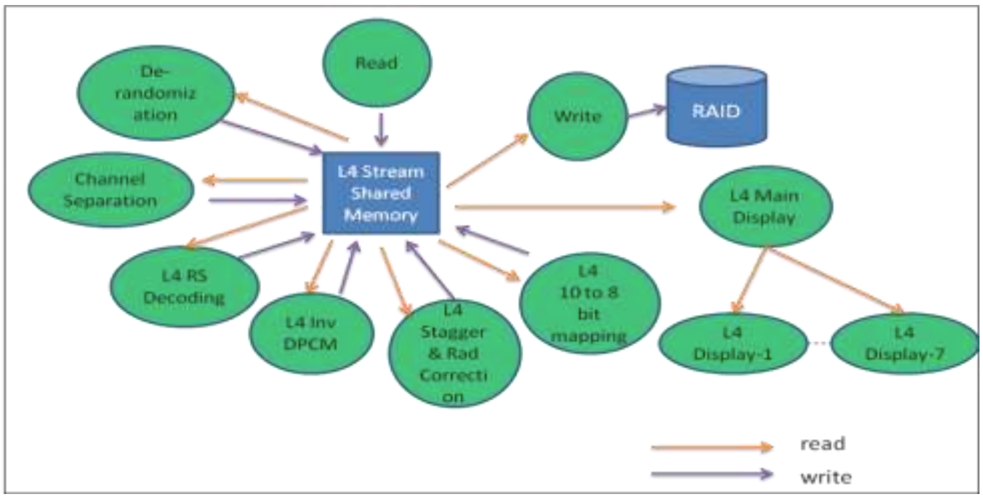

Fig. 4: Communication Requirements of LISS-IV Stream (LISS-IV Sensor)

C. Agglomeration

In the Agglomeration stage of the design, the task and communication structures defined are evaluated with respect to performance requirements and overheads. Tasks are combined into larger tasks to improve performance. Decisions made in the partitioning and communication phases are revisited with a view to obtaining an algorithm that will execute efficiently on the target computer system. The number of tasks yielded by the agglomeration phase is checked to be either less than or equal to the number of processors.

1) Agglomeration of the S1 (LISS-III) stream processes: The agglomeration of the S1 (LISS-III) stream processes is done as follows.

(1) The 'de-randomize', 'channel separate' and 'write' processes are combined to form a single process and named as 'write' process.

(2) 'LISS-III RS Decoding', 'LISS-III Inverse DPCM', 'LISS-III Radiometric correction' and 'LISS-III 10 to 8 bit mapping' processes are combined to form a single process named 'LISS-III process'.

(3) 'AWiFS RS Decoding', 'AWiFS Inverse MLG', 'AWiFS Radiometric correction' and 'AWiFS 12 to 8 bit mapping' processes are combined to form a single process named 'AWiFS process'.

The partitioning diagram for LISS-III stream, after agglomeration phase, showing the hierarchy of processes is as shown in Fig. 5.

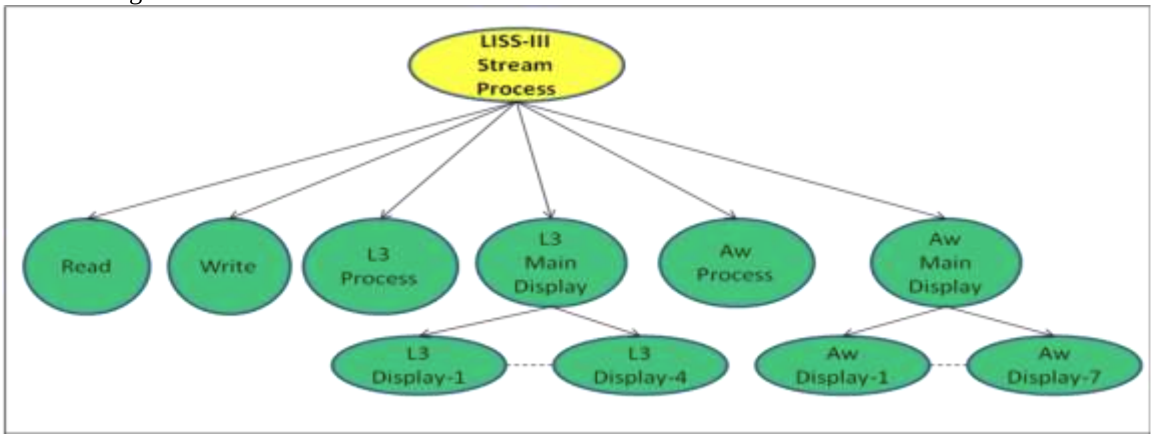

Fig. 5: LISS-III Stream processes after Agglomeration

Therefore now after agglomeration, the communication structure is defined between the processes of LISS-III Stream for LISS-III sensor data processing & display are as follows.

(1) 'read' & 'write' processes,

(2) 'write' and 'LISS-III process',

(3) 'LISS-III process' and 'LISS-III Main Display' processes,

(4) 'LISS-III Main Display' and 'LISS-III Display-1 to LISS-III Display-4' processes.

The communication structure that is defined between the processes of AWiFS sensor data processing & display is as follows.

(1) 'write' and 'AWiFS process',
(2) 'AWiFS process' and 'AWiFS Main Display' processes,
(3) 'AWiFS Main Display' and 'AWiFS Display-1 to Display-7' processes.

Fig. 6 shows the LISS-III stream communication requirements after agglomeration.

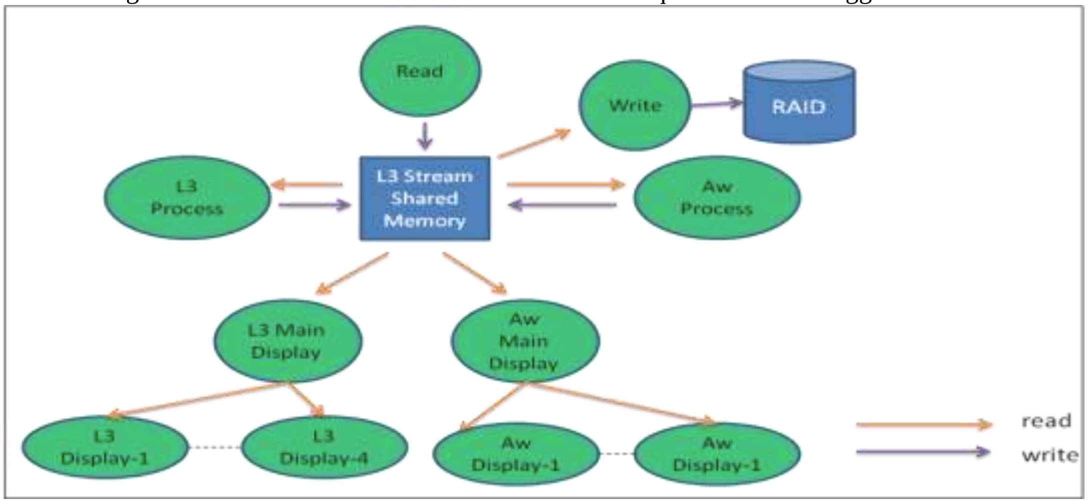

Fig. 6: Communication Requirements of LISS-III Stream (after Agglomeration)

2) *Agglomeration of the S2 (LISS-IV) stream processes:* The agglomeration of the S2 (LISS-IV) stream processes is done as follows.
(1) The 'de-randomize', 'channel separate' and 'write' processes are combined to form a single process and named as 'write' process.
(2) 'LISS-IV RS Decoding', 'LISS-IV Inverse DPCM', 'LISS-IV Stagger & Radiometric correction' and 'LISS-IV 10 to 8 bit mapping' processes are combined to form a single process named 'LISS-IV process'.
The partitioning diagram for LISS-IV stream, after agglomeration phase, showing the hierarchy of processes is as shown in Fig. 7.

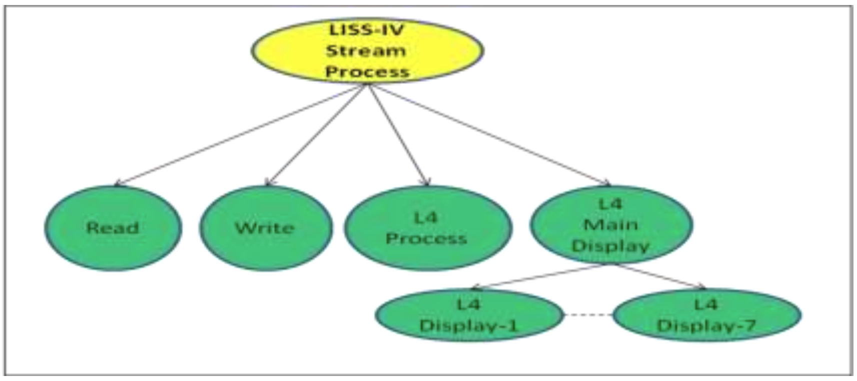

Fig. 7: LISS-IV Stream processes after Agglomeration

After agglomeration, the communication structure defined between the processes of LISS-IV Stream for LISS-IV sensor data processing & display are as follows.
(1) 'read' & 'write' processes,
(2) 'write' and 'LISS-IV process',
(3) 'LISS-IV process' and 'LISS-IV Main Display' processes,
(4) 'LISS-IV Main Display' and 'LISS-IV Display-1 to LISS-IV Display-7' processes.

The Fig. 8 shows the LISS-IV stream communication requirements after agglomeration.

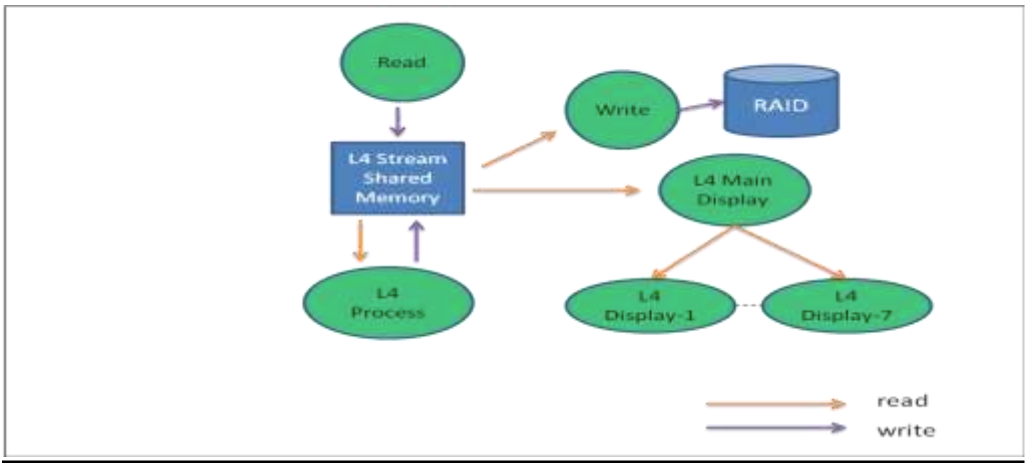

Fig. 8: Communication Requirements of LISS-IV Stream (after Agglomeration)

D. Mapping

The number of tasks is reduced to exactly one per processor.

IV. IMPLEMENTATION

There are two modes of multi tasking possible on multi-core, multiprocessor systems. Thread based multitasking and process based multitasking. Thread based multi tasking has advantages in terms of sharing the same address space resulting in less resource utilization and optimizing the CPU idle time. In process based approach the processes are insulated from each other by operating system which prevents the error in one thread causing collapsing of other process. Even though there are different processes, the read and write process have higher priority and are more critical than processes and the error in other processes should not lead to stopping of these processes.

The software is developed using C language. It uses Xlib/Motif for displaying of satellite data. Motif is a high level Graphical User Interface toolkit [5]. Xlib provides a low level interface to X window system and it has less overhead when compared to other display tool kits. Since display is the visual interface, the processing for display needs to be done in an optimized way. Therefore Xlib/Motif was used for displaying of the satellite data.

The display monitors belong to separate workstations connected to the main processing node through high speed 1Gbps network link over a Gigabit Ethernet, 24 port switch and each has a resolution of 1920 x 1200. To implement shared memory and synchronization between processes System-V IPC mechanisms are used. There are other IPC mechanisms such as POSIX semaphores which provide no mechanism to wake a waiting process when a different process dies while holding a semaphore lock. There is also no POSIX way of listing the semaphores in the OS to identify and clean them up. The POSIX section on SysV IPC does specify the ipcs and ipcrm tools to list and manipulate global SysV IPC resources. No such tools or even mechanisms are specified for POSIX IPC. Also POSIX IPC is less widely implemented.

The functional scope of the application is reconfigurable at compile time based on the number of processors available on the target configuration. This is achieved using common interface definitions for each of the processes which in turn are implemented as functions and function pointers are used to either load a functional module with no code level changes. As the application is POSIX compliant it is portable.

V. DEPLOYMENT

The total number of processes for this application is 31 which include control process (1), read processes (2), write processes (2), stream wise processes (2), sensor wise processes (3), main display processes (3), display processes (18). Among these, control and stream wise processes are in a suspended state after the child processes are triggered and once the display processes are created, the main display processes are also in a suspended state till the display processes return. Hence the total active processes are 25.The minimum number of cores/logical processors required for these processes is 25.

This version of application is deployed on an Intel Xeon, 4 CPU, 6 Core server with 64GB memory and Red Hat Enterprise Linux version 5.6.

The display monitors belong to separate workstations connected to the main processing node through high speed 1Gbps network link over a Gigabit Ethernet, 24 port switch. Fig. 9 shows the configuration of the system connectivity. The range of the pixel numbers displayed on each monitor is also shown. For example, on monitor number 1, pixels number 1 to 1920 are displayed, on monitor number 2 pixels number from 1921 to

3840 are displayed and so on. The amount of data processed and displayed for LISS-IV sensor is 14.3Mbytes/sec, LISS-III sensor is 6.25 Mbytes/sec and for AWiFS sensor is 6.25 Mbytes/sec.

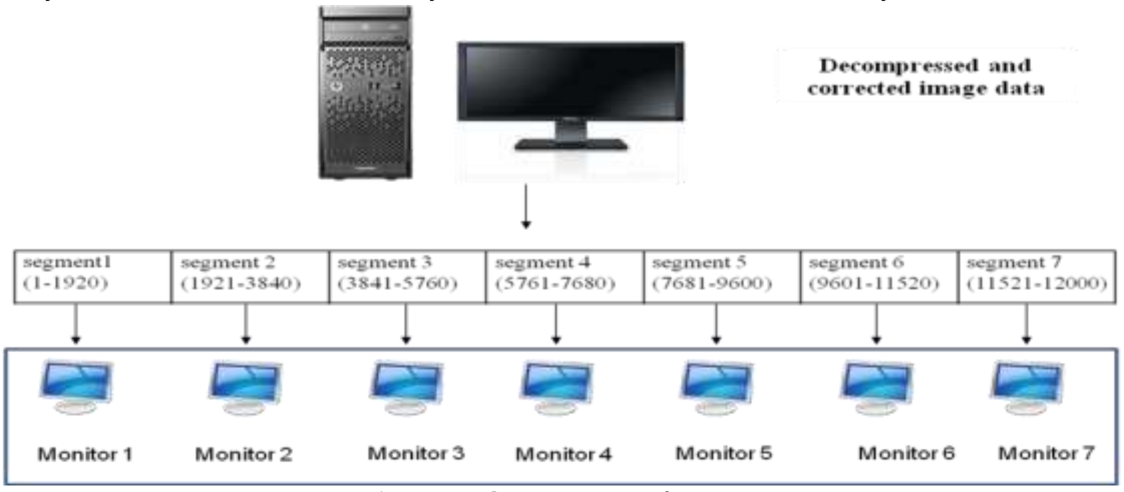

Fig. 9: Display systems configuration

A. Re configurability

The application is basically developed for full resolution display of the 3 sensors with total number of parallel processes equal to 25 requiring a system with a minimum configuration of 25 cores/logical processors. Reduced scope versions are deployed at different location. The first one is configured for sub-sampled mode of display of all the 3 sensors wherein the processing is done only for a sub-sampled number of the pixels. Here the total number of active processes is 7. The second version is configured for single sensor visualization. The third version is only with acquisition function alone and the total number of active processes is 4.

VI. PERFORMANCE EVALUATION

A. Timing Analysis

The time available for each of the processes is analyzed based on the data periodicity. The down linked data from the data acquisition is handled as a block rather than each frame. One block of LISS-IV from FEH which is of 463 KB is sent to system at 220 Mbytes /sec in a time of 4 ms. So time interval of 31.6 (36 lines x 0.8782 msec LIT) ms – 4 ms = 27.6 ms is available for other processes. Hence for each process a time of 27.6 ms is available to complete its function.

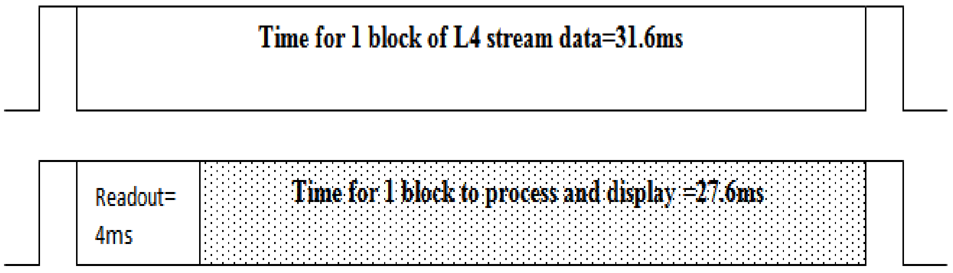
Time for 1 block of L4 stream data=31.6ms
Readout= 4ms
Time for 1 block to process and display =27.6ms

Fig. 10: L4 data processing timing

One block of LISS-III+AWiFS from FEH which is of 291.75 KB is sent to the system at 220 Mbytes /sec in a time of 2.6 ms. So time interval of 19.8 - 2.6ms = 17.2ms is available for the two sensor wise processes that is LISS-III and AWiFS.

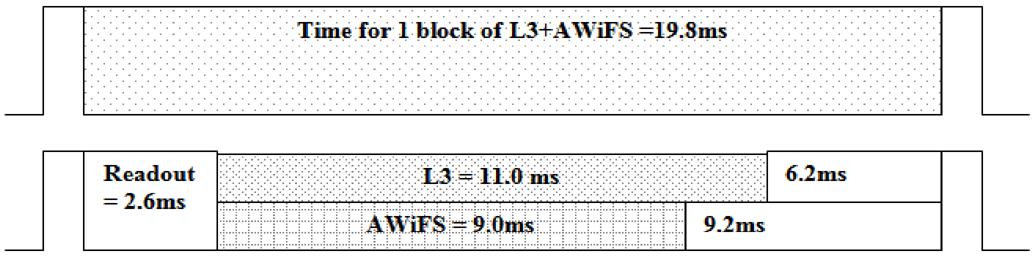

Fig. 11: LISS-III+AWiFS data processing timing

B. Data Volume and Number of Operations

Amount of data to be written to disk
LISS-IV stream data = 463 KB in 28.6 ms = 16.1 MB/s
LISS-III stream data = 256 KB in 17.2 ms = 15 MB/s

1) No of operations performed per one block: One block of LISS-IV stream consists of 36 lines with 13176 bytes record length and it has to be separated into I & Q channels performing de-randomization in real time. This requires 6406439 operations. The preprocessing of LISS-IV data for display consisting of Inverse DPCM, radiometric and stagger correction and image display requires 19142568 operations to be performed. So total number of operations performed per block of LISS-IV data are 6406439 + 19142568 = 25549007.

One block of LISS-III stream data consists of interleaved LISS-III and AWiFS data, of which 6 lines are of LISS-III sensor each of 21784 bytes record length and 2 lines are of AWiFS sensor each of 65352 bytes record length. This LISS-III stream data has to be separated in I & Q channels which correspond to LISS-III sensor data and AWiFS sensor data performing de-randomization in real time. This requires 3921201 operations to be performed. The preprocessing of LISS-III and AWiFS data involves band data separation, Inverse DPCM/Inverse MLG and radiometric correction. This preprocessing along with image display of LISS-III and AWiFS sensors require 6045778 operations. So total operations performed per block of LISS-III and AWiFS data are 9966979.

VII. CONCLUSION

The application design is modular and reconfigurable based on the number of cores available on the system. Due to this re-configurability testing with newer satellites involving additional processes is easier at the same time the lower configurations will still support the acquisition with reduced scope performing the essential processes for acquisition.

Acknowledgements

The authors extend their sincere gratitude to Deputy Director, SDAPSA and Director, NRSC for their constant encouragement and support.

REFERENCES

[1]. A.C. Sodan, Jacob Machina, Arash Deshmeh, Kevin Macnaughton, Bryan Esbaugh "Parallelism via Multithreaded and Multicore CPUs"
[2]. The Linux Programmer's Guide by Sven Goldt, Sven van der Meer, Scott Burkett, Matt Welsh
[3]. Designing and Building Parallel Programs, by Ian Foster http://www.mcs.anl.gov/~itf/dbpp/text/book.html
[4]. Texas Instruments Multi core programming guide
[5]. Ibrahim F. Haddad, X/Motif Programming, http://www.linuxjournal.com/article/3666