Parallel Digital Video Processing System Architectures and Applications on ADSP-21160M SHARC DSP® DSP

D. Turgay Altilar, Silvano P. V. Barros, Yakup Paker
Queen Mary, University of London
Department of Computer Science
Mile End Road E1 4NS London-UK
{altilar,silvano, paker}@dcs.qmul.ac.uk

Abstract - This paper presents architectures for Parallel Digital Video Processing Systems and discusses different mappings suitable for DSP processors, in particular the ADSP-21160M SHARC DSP. Our objective is to define and implement low-cost, scalable and state-of-the-art systems for real-time digital video processing models. In this paper, systems hardware is considered extensively but software issues left untouched unless it is directly related to the hardware performance. The proposed ideas and developed system are the results of video stream processing projects conducted at Queen Mary, University of London, for almost a decade. Since the data volume involved in video processing is considerably large, data movement to and from intelligent frame buffers has to be optimised to attain good performance out of the processing system. The architectures are generic in nature. Although they have been developed for video processing, any algorithm dealing with huge amounts of periodic and continuous data can be prospective applications for the architectures.

Index Terms - Digital video processing, real-time processing, I/O intensive applications, frame buffer, dual port memory.

I. INTRODUCTION

Integrating the software and hardware technologies to construct and to handle video streams for the broadcast quality TV productions has been researched extensively for the last decade. With the use of high-speed communication technology coupled with powerful processors such as SHARC DSP ADSP-21160M, it is possible to process broadcast quality video sequences in real-time over a network. Digital video processing embodies a wide span of applications such as signal acquisition, production side signal processing, transmission, receiver side signal processing and signal presentation, most of which can be addressed by DSP based processing systems. We have focused on the production signal processing aspect, which is the most demanding in terms of the computing power requirements (Fig.1).
Since computing power to process video is considerable high, one of the ways to cope with the demands in real-time, we believe, is combining parallel processing with an intelligent frame buffer technology.
Computer graphics techniques such as rendering and radiosity as well as majority of the digital video processing applications such as mixing and chromakeying are compute intensive processes. These algorithms have been parallelised for various types of computer architectures [1]. Most of these algorithms suffer from the problems such as unbalanced load distribution and under-utilisation of processors and I/O channels [2].

Our architectural approach involves the use of computing clusters based on shared address space multi-processors. If greater computing power is required then a number of these clusters may be connected using a dedicated network to create a larger cluster, which satisfies the scalability criteria.

The programming approach is Single Program Multiple Data (SPMD) parallelism. The overall aim is to establish a real-time digital video processing architecture. Data distribution, load balancing, processor utilisation and fast data transfer are the key concerns to be addressed.

Digital video processing is known to be one of the most input/output intensive applications. A one-second duration PAL sequence of resolution 576x720 can be represented digitally by approximately 30 MB. Furthermore digital video processing does not only imply processing individual frames of a single stream, as in the case for some image deformation techniques or chromakeying, but it also involves the processing of more than one frame taken from different image streams simultaneously. For example, mixing two frames by using depth values requires four video streams to be processed simultaneously, increasing the processing requirements four fold to a total of 120 MB of data per second.

The other two important characteristics of video data are periodicity and continuity. A PAL video stream comprises 25 frames per second and a new frame becomes available for processing every 40 ms. (For NTSC the standard is 30 frames per second)

Queen Mary, University of London (QMUL) is internationally recognised as one of the pioneering centres for Parallel Computing and High Performance Computing. QMUL has been involved in video processing projects since 1992 funded by national and international sources such as EPSCRC, RACE, ESPRIT and ACTS [3]. MONALISA is a virtual studio application developed under the RACE, for which a demonstration system was developed using 4 Motorola 96002 based processing boards [4,5,6] each consisting of 2 DSP's. AMPA (Advanced Multimedia Parallel Accelerator) architecture has been designed under ACTS and ESPRIT as a scalable multimedia processing architecture, involving the use of multiple TMS320C80 DSP's [7].

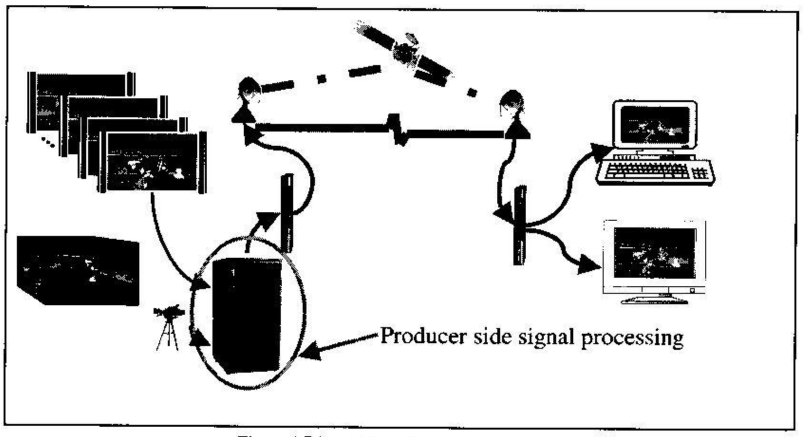

Figure 1 Digital Video Processing

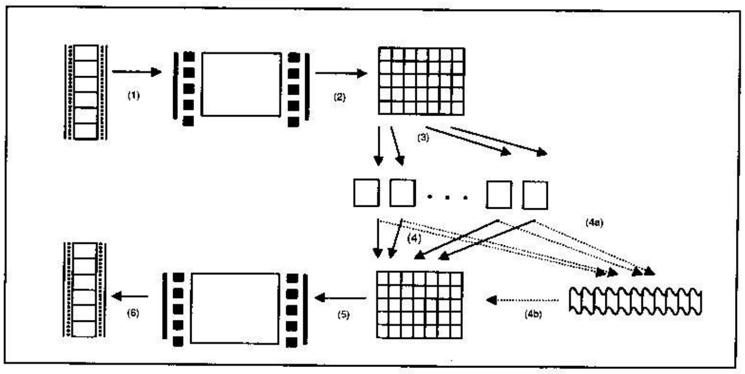

Figure 2 Data flow in video processing

The architectures and techniques presented in this paper have been developed and designed by taking into account the experiences gained from the aforementioned projects.

Although both hardware and software issues need to be addressed for the project, we present herein only the hardware system architectures and closely related scheduling and data distribution techniques. Scheduling and data partitioning approaches have been presented in previous publications [8,9] while software issues are the subject of a separate publication yet to be published.

The rest of the paper is structured as follows. Section II introduces a generic video processing algorithm classification, the use of the client-server programming model and the flow of video data in the implementation of SPMD class of programming. Section III explains the essential system architecture and proposes four mappings on ADSP-21160M utilising different features of the chip. The impact of the communication between the intelligent frame buffer and IOP during video processing is and the list of commands to be processed by the intelligent frame buffer is given discussed in Section IV. Section V discusses the use of I/O channels in processing and different frame buffer management techniques. Section VI proposes a three-buffer technique for the fastest throughput for two processors sharing a memory. The paper ends with a conclusions and further research section.

II. SOFTWARE ARCHITECTURE AND VIDEO DATA FLOW IN BRIEF

In video processing applications, processors fulfil two main roles, i.e. server and client, which are required to implement SPMD class programming model. A client is allowed to read/write data from/to the frame buffer under the authority of the server. In addition, the server is responsible for partitioning the input data sets and reconstructing the outputs after the processing has been completed by the clients. A typical parallel video processing data flow cycle is given for video-to-video processing in Fig.2. Data flow starts with the reading of a single frame by the server (step 1). The server partitions data by determining pointers or reconstructing the tiles (step 2). The server sends each client tiles (step 3) and receives output from clients after processing (step 4). The server finally reconstructs received data to produce a single output (step 5) and writes back to frame buffer (step 6).

The algorithm is data independent which has the benefit of simplifying the data distributing and load balancing task by the server during the tiling phase and the frame reconstruction afterwards. This is because the tile processing time is equal for every tile; hence, the order and the periodicity between input and output tiles are maintained in lock-step. Thus, processed tiles from clients are returned in the same order that input tiles were allocated to the clients. It is also assumed that initialisation and data transfer costs are equal in every communication session.

Considering the type of applications, the proposed architecture will serve for:

- stream based computations in which a stream (or a number of streams) is to be processed continuously
- frame based computations in which a stream (or a number of streams) is to be processed frame by frame
- algorithms that need feedback for forthcoming steps
- algorithms that never need feedback from the previous steps
- real-time or non-real-time applications

and considering the input and output data type

- video stream to data (stream)
- video stream to video stream
- data (stream) to video stream
- data (stream) to data (stream)

The number of input or output streams could be more than one as in mixing.

III. SYSTEMS ARCHITECTURE

The target parallel architecture is based on the client-server model having a point-to-point communication between the host and client processors for control purposes. The programming model is SPMD (Single Program Multiple Data) in which the same process (algorithm) runs on every client processor.

A typical hardware configuration for such a target comprises a server processor, a frame buffer and a number of client processors connected via a dedicated high speed I/O bus and a signal bus (Fig.3). Data transfers are made over the high speed I/O bus. The frame buffer is the medium into which a video stream is written by an input device such as video player or camera. It is also the final destination for the processed data written to by the clients. The written data will then be transmitted to either a display or a high-speed disk for future use.

Since the frame buffer can accommodate only one I/O transaction at a time, any access to the frame buffer should be under the control of the server for the given architecture.

The server has the added responsibility for controlling the overall application. It is the server's responsibility to initialise clients, to partition data, to send data addresses to clients to read and write, and to act as general arbiter of the high speed I/O Bus.

Given the internal memory blocks, multiple internal bus structure, integrated I/O subsystem and link-port architecture of the ADSP-21160M SHARC DSP processor [10], shared memory, distributed memory and hybrid memory architectures can be realised. As video processing algorithms vary, four generic cluster-based architectures were defined to fulfil different aspects of video processing algorithms as follows:-

- Standard Video Processing Cluster (VIP-C)
- Video Processing Super Cluster (VIP-SC)
- Video Processing Linked Cluster (VIP-LC)
- Video Processing Cross Cluster (VIP-XC)

A detailed description of each of these architecture classes follows (but commonality between the classes are omitted from B, C and D).

A. Standard Video Processing Cluster (VIP-C)

The VIP-C architecture comprises six ADSP-21160M SHARC DSP processors, a Dual-Port Shared Memory, an I/O processor and a host processor as shown in Fig.4. The intelligent frame buffer shown in Fig.4 is not a part of the cluster. The hardware structure of VIP-C is composed of three of different elements:

- The host processor running the user interface modules and the application programs;
- a processor pool of 6 ADSP-21160M's; and
- a frame buffer system to which at least one video input such as camera, video player and one video output such as video recorder, display are attached.

These elements are linked to each other over different mediums such as link port connections, external port bus, dedicated I/O and control connections. Video I/O is done via D1 codecs for digital serial video signals according to the CCIR 601/656 standards.

FIFO-buffered I/O processors (IOPs) are used to connect I/O devices and clusters to the frame buffer. All transfers to/from frame buffer is done under the supervision of the host processor. (Actual transfer is controlled by an address generator.).

The two main elements of VIP-C are the processor pool and the frame buffer. The hardware characteristics of these elements are given in this section. These specifications and features are common to the other three classes of architectures whose descriptions follow and will, therefore, not be repeated in later sections

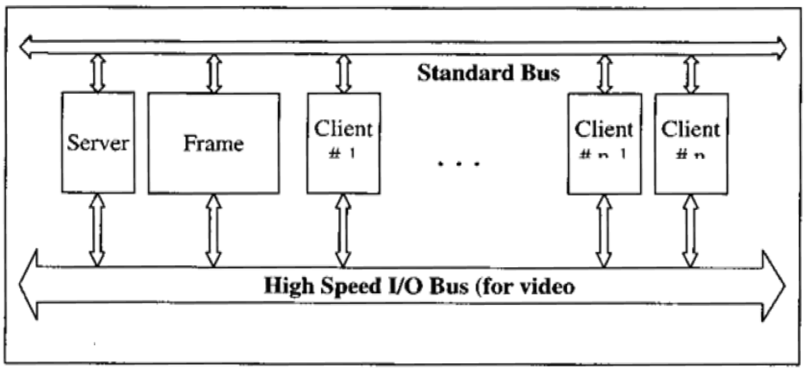

Figure 3 A parallel video processing system

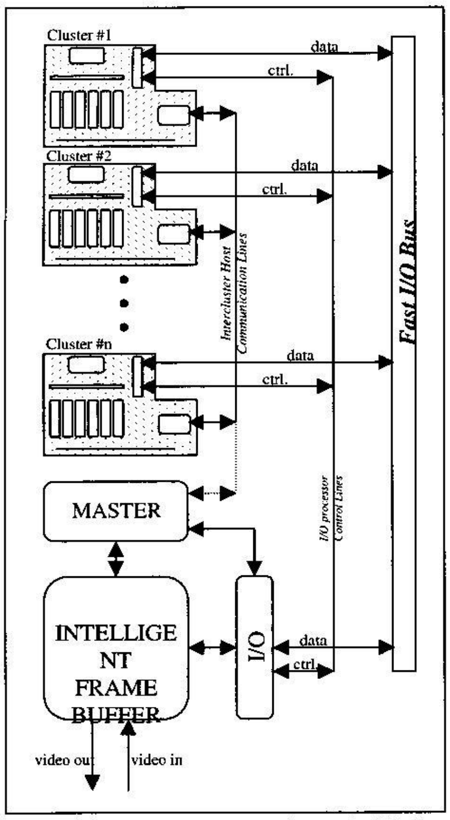

Figure 5 A Super Video Processing Cluster (SupVIP-C)

The Processor Pool
The processor pool is based on ADSP--21160M SHARC DSP processors operating at a speed of 80 MHz (480 MFLOPS Peak, 320 MFLOPS sustained). Each ADSP-21160M has a local dual-ported SRAM of 4 Mbit. Embedded I/O Processor provides 6 Link ports for inter-DSP communication, 2 serial port for serial communication and DMA controller. External port is capable of addressing 16 GB (2 GB per processor). Six ADSP-21160M processors connect to a common bus to access an external dual-port memory. The external memory is the video input source as well as video output destination for processor pool. Control of pool processors and data transfer between pool processors and the host is done via link port connections. Serial lines are left untouched since the control will be done via link-port connections.

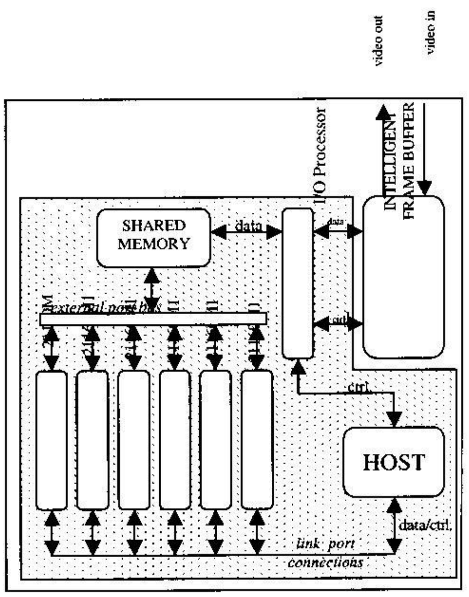

Figure 4 A Video Processing Cluster (VIP-C)

The Frame Buffer
The frame buffer system is used for video input/output. Its multitasking I/O capabilities facilitate handling of several independent video streams in real time up to a total rate of 500 MB/s.

The memory bank has a capacity of 128 MB, which can be expanded up to 4 GB dynamic RAM. One of the ADSP-21160M can also take over the tasks done by both I/O processor and the host. This slight variation in the configuration is possible since ADSP-21160 has an embedded I/O unit. However in this case the controlling DSP communicates with a computer on which a user interface would run. If the user interface is not complicated even serial ports can be used to drive a terminal.

B. Video Processing Super Cluster (VIP-SC)

This class of architecture is essentially based on the use of a number of VIP-C units. In order to create more powerful clusters a modular and scalable architecture (VIP-SC) is proposed in Fig 5. A number of VIP-C clusters are used and the control of the intelligent frame buffer is assigned to the master. The host processors of each VIP-C maintains local control but communicate with an overall master which maintains coordination

among the different VIP-C clusters. This hierarchic order of command (from master through local hosts to the cluster processors) regulates super cluster wide processes as a single SPMD process. The data transfer starts between the intelligent frame buffer and external memory of a cluster through IOP, Fast I/O Bus and cluster IOP. The number of VIP-C clusters depends on the speed of the Fast I/O Bus. It is usual that such constraints stem from the data transfer side of the process since the bottleneck of parallel computing is data transmission rather than computation for most of the video processing algorithms. The Fast I/O Bus should be fast enough to feed clusters with video data so that processors within each cluster should never become idle. The ratio of I/O channel rates and the complexity of the computation would determine the best number of clusters used in a particular implementation. Further details describing such an analysis can be seen in a recent paper [8].

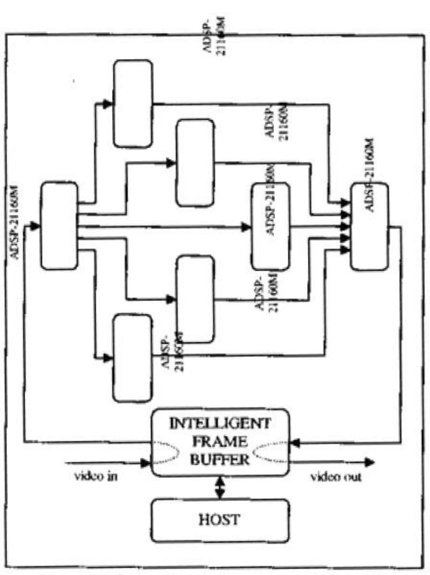

Figure 6 A Video Processing Cluster (VIP-LC-R)

C. Video Processing Linked Cluster (VIP-LC)

The VIP-LC architecture is based on a reduced number of processors (Fig.6) with respect to the constraints of the six link-port DSP design. There are two essential differences between VIP-C and VIP-LC. The first one is the use of ADSP-21160M's as I/O processors since an embedded I/O processing unit exists in each ADSP-21160M. In this case, ADSP-21160M should communicate with the Intelligent Frame Buffer directly. Section IV describes the system calls to be supported by ADSP-21160M for use with this architecture. In this configuration the input and output processors can additionally perform pre- and post-processing activities on the frame data if required. VIP-LC involves the use of link-ports rather than an external memory. The internal memory which is 4M bits and the link port speed which is 80MB/s would accommodate most video processing functions within the 40ms time constraint. Although not shown in Fig.6 a very small size shared memory is used for cluster-wide communication purposes.

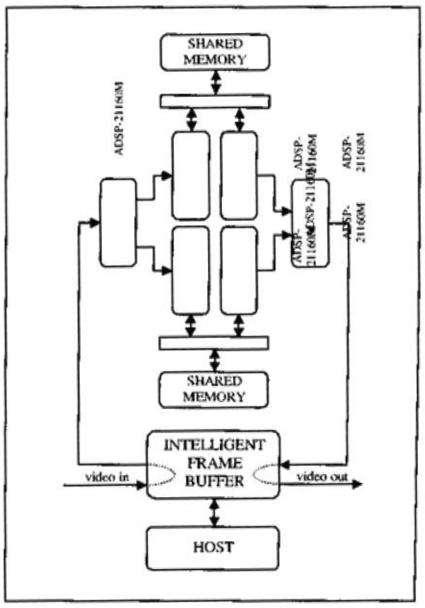

Figure 7 A Video Processing Cluster (VIP-XC-R)

D. Video Processing Cross Cluster (VIP-XC)

In VIP-XC, two ADSP-21160M's act as I/O processors (one input and one output) and also to pre- and post-prosess frame data when required (Fig.7). The remaining four ADSP-21160M's, access two separate shared memory modules simultaneously but in pairs. Access to a shared memory does not yield performance gain unless processing and I/O durations can be overlapped. An enhancement is, therefore, proposed in Section VI for this architecture to improve the performance. Although not shown in Fig.7 a very small size shared memory is used for cluster-wide communication.

among the different VIP-C clusters. This hierarchic order of command (from master through local hosts to the cluster processors) regulates super cluster wide processes as a single SPMD process. The data transfer starts between the intelligent frame buffer and external memory of a cluster through IOP, Fast I/O Bus and cluster IOP. The number of VIP-C clusters depends on the speed of the Fast I/O Bus. It is usual that such constraints stem from the data transfer side of the process since the bottleneck of parallel computing is data transmission rather than computation for most of the video processing algorithms. The Fast I/O Bus should be fast enough to feed clusters with video data so that processors within each cluster should never become idle. The ratio of I/O channel rates and the complexity of the computation would determine the best number of clusters used in a particular implementation. Further details describing such an analysis can be seen in a recent paper [8].

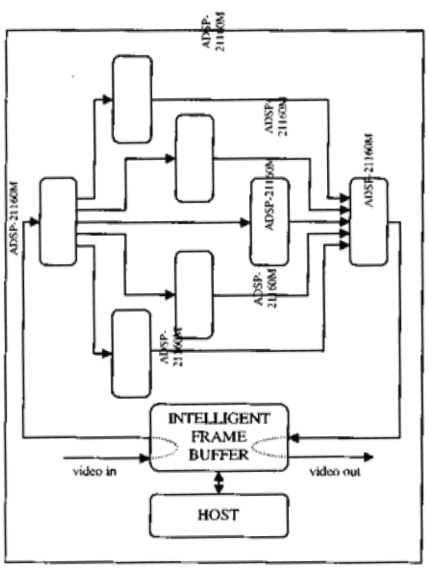

Figure 6 A Video Processing Cluster (VIP-LC-R)

C. Video Processing Linked Cluster (VIP-LC)

The VIP-LC architecture is based on a reduced number of processors (Fig.6) with respect to the constraints of the six link-port DSP design. There are two essential differences between VIP-C and VIP-LC. The first one is the use of ADSP-21160M's as I/O processors since an embedded I/O processing unit exists in each ADSP-21160M. In this case, ADSP-21160M should communicate with the Intelligent Frame Buffer directly. Section IV describes the system calls to be supported by ADSP-21160M for use with this architecture. In this configuration the input and output processors can additionally perform pre- and post-processing activities on the frame data if required. VIP-LC involves the use of link-ports rather than an external memory. The internal memory which is 4M bits and the link port speed which is 80MB/s would accommodate most video processing functions within the 40ms time constraint. Although not shown in Fig.6 a very small size shared memory is used for cluster-wide communication purposes.

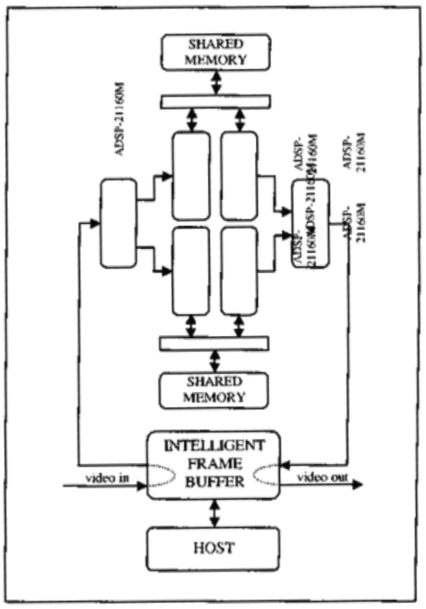

Figure 7 A Video Processing Cluster (VIP-XC-R)

D. Video Processing Cross Cluster (VIP-XC)

In VIP-XC, two ADSP-21160M's act as I/O processors (one input and one output) and also to pre- and post-prosess frame data when required (Fig.7). The remaining four ADSP-21160M's, access two separate shared memory modules simultaneously but in pairs. Access to a shared memory does not yield performance gain unless processing and I/O durations can be overlapped. An enhancement is, therefore, proposed in Section VI for this architecture to improve the performance. Although not shown in Fig.7 a very small size shared memory is used for cluster-wide communication.

among the different VIP-C clusters. This hierarchic order of command (from master through local hosts to the cluster processors) regulates super cluster wide processes as a single SPMD process. The data transfer starts between the intelligent frame buffer and external memory of a cluster through IOP, Fast I/O Bus and cluster IOP. The number of VIP-C clusters depends on the speed of the Fast I/O Bus. It is usual that such constraints stem from the data transfer side of the process since the bottleneck of parallel computing is data transmission rather than computation for most of the video processing algorithms. The Fast I/O Bus should be fast enough to feed clusters with video data so that processors within each cluster should never become idle. The ratio of I/O channel rates and the complexity of the computation would determine the best number of clusters used in a particular implementation. Further details describing such an analysis can be seen in a recent paper [8].

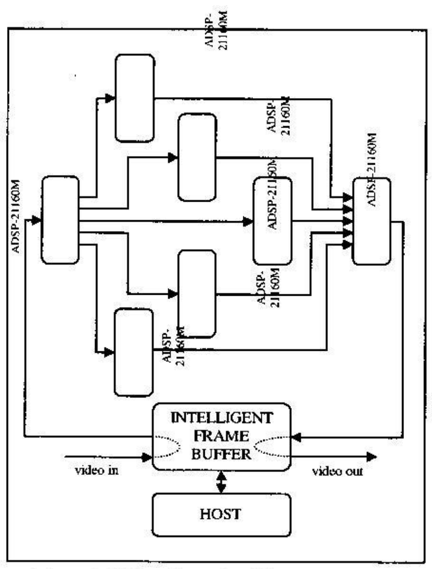

Figure 6 A Video Processing Cluster (VIP-LC-R)

C. Video Processing Linked Cluster (VIP-LC)

The VIP-LC architecture is based on a reduced number of processors (Fig.6) with respect to the constraints of the six link-port DSP design. There are two essential differences between VIP-C and VIP-LC. The first one is the use of ADSP-21160M's as I/O processors since an embedded I/O processing unit exists in each ADSP-21160M. In this case, ADSP-21160M should communicate with the Intelligent Frame Buffer directly. Section IV describes the system calls to be supported by ADSP-21160M for use with this architecture. In this configuration the input and output processors can additionally perform pre- and post-processing activities on the frame data if required. VIP-LC involves the use of link-ports rather than an external memory. The internal memory which is 4M bits and the link port speed which is 80MB/s would accommodate most video processing functions within the 40ms time constraint. Although not shown in Fig.6 a very small size shared memory is used for cluster-wide communication purposes.

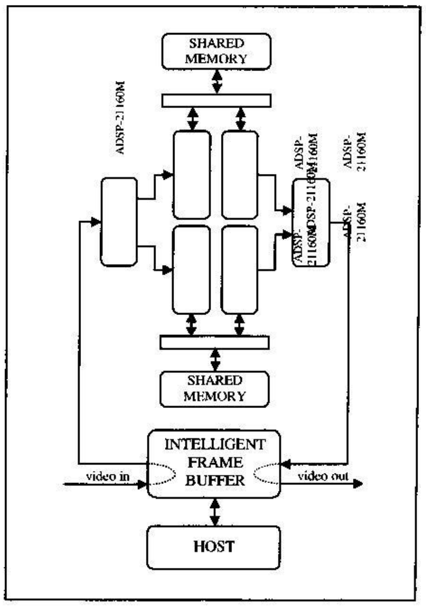

Figure 7 A Video Processing Cluster (VIP-XC-R)

D. Video Processing Cross Cluster (VIP-XC)

In VIP-XC, two ADSP-21160M's act as I/O processors (one input and one output) and also to pre- and post-prosess frame data when required (Fig.7). The remaining four ADSP-21160M's, access two separate shared memory modules simultaneously but in pairs. Access to a shared memory does not yield performance gain unless processing and I/O durations can be overlapped. An enhancement is, therefore, proposed in Section VI for this architecture to improve the performance. Although not shown in Fig.7 a very small size shared memory is used for cluster-wide communication.

IV. FRAME BUFFER-IOP COMMUNICATION

Performance of a digital video processing system is highly dependent upon data I/O management given the substantial amount of data to be transferred between various memory modules. In the proposed generic architectures, ADSP-21160M's are accessing either directly to the frame buffer or a shared memory.

Although write/read operations can be easily implemented at higher levels of programming, a lower level implementation will provide better performance. The first enhancement for data access is a command providing a write followed by a read as such a command will serve to increase the utilisation of the processors. The treatment of peroidicity and continuity of data source as indivisable write-read operation provides a good base for a performance improvement [8]. Therefore any memory access requirement should be supported by a low level indivisable write-read operation.

Considering data streams; more than one stream must be handled. Streams must be identified. In our case names are attached to streams to access them symbolically. Streams can be accessible through smaller indexed units, for this particular case the unit is a frame and the index is a frame number.

Any memory access mechanism should, thus, provide the following set of commands (system calls):

- write-read(device_no, channel_no, r_filename,
- r_frame_number, w_filename, w_frame_number)
- read(device_no, channel_no, filename, frame number)
- write(device_no, channel_no, filename, frame number)
- attach(device_no, channel_no, direction, filename,
- spec1, spec2, ...)
- detach(device_no, channel_no, filename)
- setspec(device_no, channel_no, direction, filename,
- spec_code, spec_val1,...)
- sync(device_no, channel_no, filename, channel id,
- filename, synctype)
- fastforward(device_no, channel_no, filename, speedup)
- rewind(device_no, channel_no, filename, speedup)
- skip(device_no, channel_no, filename, ratio)
- repeat(device_no, channel_no, filename, ratio)
- setframe(device_no, channel_no, filename,
- frame_number)
- getframe(device_no, channel_no, filename)

write-read sets the device and channel for communication and for given write and read files and frame numbers a write operation is followed by a read without any interruption
read and write are basic commands for reading and writing, where a specific value of frame number should imply continuous real-time reading rather than specific frame numbers.

attach and *detach* commands attaches and detaches video streams to I/O channels. Channel control is possible using these commands. Cluster host or master could decide to select directions, channels, and IOP's independently.

setspec is to change/set specifications related to the sequence. For example, frame size, video type (PAL, NTCS, YUV, RGB, etc).

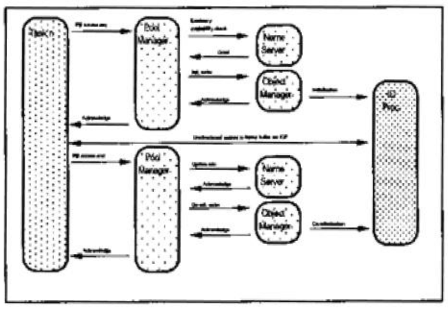

Figure 8 Standard frame buffer access protocol

sync command is for synchronising two video sequences for processing. There are two ways to synchronise them, with respect to time or with respect to frame.

skip command is to skip a frame from a sequence within a given ratio (e.g. 3 in 5 frames). QoS (Quality Of Service) issues will require such a command to reduce the I/O and processing time.

repeat command is to duplicate the last frame in the sequence.

fastforward and *rewind* is to skip a number of frames in given direction. The number of frames is determined by the speedup value desired.

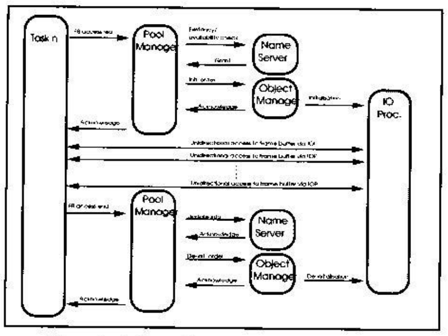

Figure 9 Dedicated frame buffer access protocol

setframe is to make IOP read the sequence beginning from a particular frame.

getframe is to determine the position of the last frame in the sequence.

V. THE USE OF DEDICATED I/O CHANNELS AND FRAME BUFFER MANAGEMENT

Previous experiences has shown that dynamic allocation of communication channel connection of client processors to the frame buffer may not be fast enough to cope with real-time constraints of digital video processing due to the high cost of channel switching [12]. The cost of channel switching is not only due to hardware switch propagation delay but also due to a hierarchy of software communication starting from the request of data form a particular processor producing an address and settling down for DMA under the control of the host and for larger clusters, the masters.

Therefore, the frame buffer management is one of the most important issues that easily influence the overall performance of the architecture. The frame buffer acts as a mounted single level directory (disk space), which enables us to make memory allocation and memory management on the principles of a disk management. Any video sequence should be created (allocated) first prior to its use. No pre-emption/swap space is available if there is not enough space to create the specified file. Processes can link this file through I/O Processors. It is also possible to link to file via different IOPs, which provides a good way of sharing a frame buffer.

Consider that there is some control software running on the host (or on the master) controlling the processor pool. This software consists of a few modules such as a pool manager, a name server, and an object server just to name the ones we are interested in for this case. The actual accesses to the frame buffer are controlled by the name server, which manages the name table, availability and access rights. This authorisation mechanism is essential to prevent conflicting access requests from clusters.

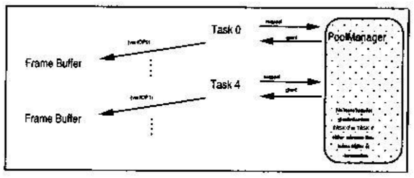

Figure 12 Dedicated execution mode

The protocol for accessing frame buffer is as follows. A process sends a message to pool manager to get/put data from/to a file whose name is submitted literally. The pool manager asks name server about the existence, (if it exists) the physical address, and availability of the file. After having the relevant information about the file, it initialises the dedicated I/O Processor for the defined task and sends an acknowledgement to the processor. Having received the acknowledgement, the processor directly read/write from/to the frame buffer and never leaves the control until it finishes its transfer. At the end of the transfer it sends a massage to pool manager to release the dedicated I/O Processor (Fig.8).

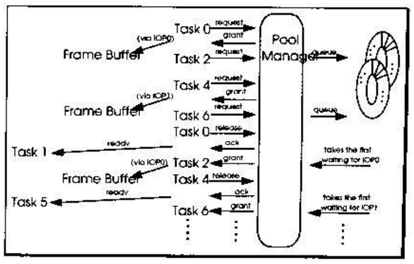

Figure 11 Shared memory execution mode

Although this is a generic protocol for a parallel machine non-shareable source management, initialisation and communication itself, takes a long time. In order to reduce this delay another dedication-based protocol was implemented. In that protocol, a process is allocated the use of an I/O Processor throughout its lifetime (Fig.9). The protocol explained above takes place only once. And process accesses frame buffer via the dedicated I/O Processor without requesting or waiting for any grant from pool manager. Accessing through only one process running in the same cluster is the constraint of this protocol. On top of these two protocols, various kinds of video sequence access schemes were implemented.

Scheme 1
Assume that there are eight VIP-C or VIP-LC constituting a super cluster. The first protocol is used and eight clusters access the frame buffer in turn. This is called generic, parallel execution mode (Fig 10). All requests are handled via queues.

Scheme 2
Assume that there are two VIP-XC constituting a cluster. Since every working processor pair will be served by I/O Processors individually we can think of four parallel tasks running over eight ADSP-21160M's. The first protocol is used and each working processor pair accesses the frame buffer in turn. By making use of the shared memory, it is possible to run processes dealing with the same frame on the same board. This is called generic, shared memory execution mode (Fig 11).

Scheme 3

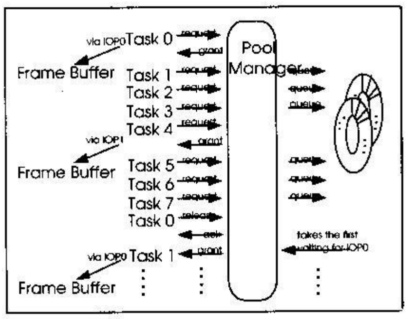

Figure 10 Generic parallel execution mode

Assume that there are two VIP-C or VIP-LC constituting a super cluster. The second protocol is used and two processes access the frame buffer. This is called dedicated simple execution mode (Fig.12). The number of the clusters depends on the I/O rates of these connections. It is assumed that I/O bus is fast enough to drive two I/O Processor virtually concurrently.

Scheme 4

This scheme was also developed for VIP-XC super clusters in which the second protocol is used, and by using shared memory, the number of processors in scheme 3 was doubled. It is called dedicated, shared memory execution mode The above four frame management schemes indicate that the dedicated approach is preferred in which channels are set once and kept untouched throughout the process. The use of shared memory increases the processing power. However, introduction of a shared memory gave rise to new problems. The solutions discussed are particular to the two-processor architecture (Section VI). Equivalent solutions may be derived for multi-processor systems having more than two processors. If it is impossible to set a dedicated channel, a FIFO buffering system is essential to cover the probable latency. The I/O Processor should be equipped with a reasonable size of the internal FIFO buffer as shown in the architectures above.

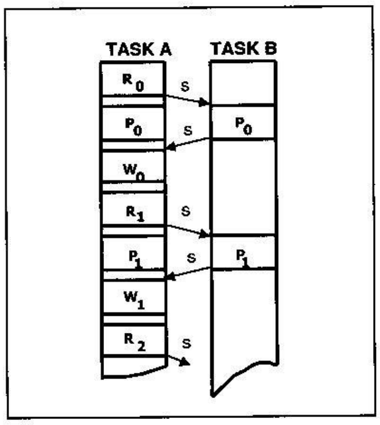

Figure 13 Usual flow of processors and signalling

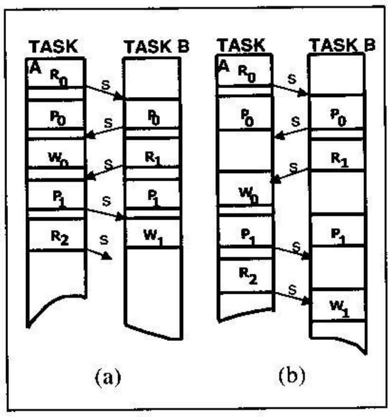

Figure 14 Virtually concurrent processing scheme

VI. SHARED MEMORY ACCESS FOR THE FASTEST THROUGHPUT

Consider an application that reads a frame from a frame buffer, processes it using two DSPs having a shared memory and writing the result back to frame buffer. This is a standard process for the VIP-XC type architecture. Assume that processing time is less-than or equal to the I/O time. The ultimate aim is to utilise both processors and I/O channels.

The usual approach for two processors sharing a memory is shown in Fig.13. Assume that Task A runs on Processor A and Task B on Processor B. Each processor processes different parts of the frame. A reads, then signals B indicating that memory access is permissible. B accesses the memory and signals A. At this point A has completed processing and writes back into memory. The processing continues in this fashion until the entire frame has been processed. Fig.13 indicates that inefficiency is possible as B can potentially remain idle while A writes/reads to/from frame buffer.

In order to increase the utilisation of B, the following scenario depicted in Fig.14a appears to offer an adequate solution at first sight. A reads a frame, signals B, they process together, B signals A, then B reads the second frame, while it lets A to write first frame back to frame buffer, and so forth. Although it seems that the utilisation of B improves, considering the shared resource (IOP/Frame buffer), they have to wait each other as concurrent access to IOP/Frame buffer is not permissible. (Note that depending on the architecture IOP and/or the frame buffer become unsharable common resource.) As shown in Fig.14b, the actual flow includes mutual exclusion while accessing to the IOP/frame buffer. They can process concurrently using their own memory banks however I/O is always virtually concurrent.

The actual solution to that problem is overlapping processing and frame buffer accessing sessions of the different processors. However, three buffers and a switch mechanism are required to enables processors to access any of the three buffers in turn depending on mutual signals.

A solution with dependencies (signals and waits) is proposed in Fig 15. Ignore the first four idle stages of Task A. Similarly ignore the dummy process P-1 between to consecutive readings of R0 and R1 of Task B. (These dummy slots - frame read time - are negligible considering a stream of video.) Task A initially waits on P0 to be read by Task B. Whenever it finishes processing frame #0, it waits on the Task B to process the same portion. Task A writes back and send a signal to Task B indicating that frame #0 is finished, therefore a new frame (frame #4) can be read. At any time, there are three frames accessible in this flow. Since read cycle of Task B corresponds to a process cycle of Task A, and write cycle of Process A corresponds to a process cycle of Task B. After every two units of time, a new frame is produced. A hardware solution for such an approach will definitely improve the throughput.

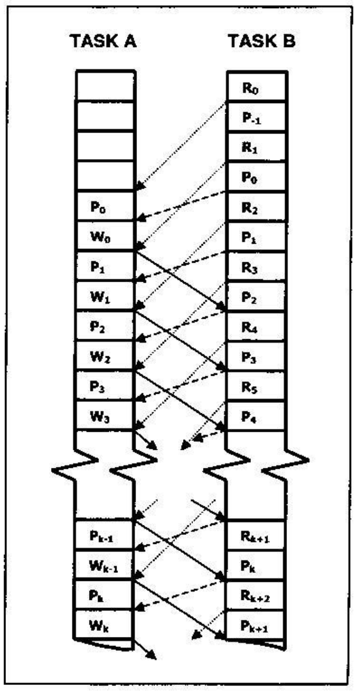

Figure 15 Three buffer scheme

VII. CONCLUSION AND FURTHER RESEARCH

We have proposed several new architecture designs using state-of-the-art approaches and based on the use of ADSP-21160M clusters. Within these scenarios, we have addressed three main problem areas, namely frame buffer-IOP communication requirements, frame buffer management and shared memory management. For each of these areas, we have proposed enhancements and solutions that would achieve the required level of performance for a viable real-time multi-streamed video processing system.
The SIMD architecture of the ADSP-21160M is not directly exploited since our approach is based on course-grained, loosely coupled SPMD processing. However exploitation of the SIMD features of the processor should be investigated further to achieve the best out of the technology.

The scalability issues relating to the proposed VIP-LC and VIP-XC architectures have not been addressed sufficiently. Further investigation is essential on this aspect to cope with different video data sizes.

VIII. REFERENCES

[1] Whitman S., Hansen D.C., Crockett T.W., Recent Developments in Parallel Rendering, IEEE Computer Graphics and Applications, July 1994.

[2] Singh J.P., Gupta A., Levoy M., Parallel Visualisation Algorithms: Performance and Architectural Implications, IEEE Computer, July 1994.

[3] Gibbs S, Arapis C, Breiteneder C, Lalioti V, Mostafawy S, Speier J, Virtual Studios: An Overview, IEEE Multimedia, January-March 1998.

[4] Blondé L, Buck M, Galli R, Niem W, Paker Y, Schmidt W, Thomas G, A Virtual Studio for Live Broadcasting: The Mona Lisa Project, IEEE Multimedia, Vol. 3 No. 2, Summer 1996.

[5] Proceedings of the European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production, Paker Y. and Wilbur S, (Eds.), Springer, November 1994.

[6]

[7] Young E S, Schmidt W, Altilar D T, Paker Y, Ampa Architecture Specification, Project Deliverable 1, Nov 1996.

[8] Altilar D T, Paker Y, An Optimal Scheduling Algorithm for Parallel Video Processing, Proceedings of International Conference on Multimedia Computing and Systems'98, Austin Texas USA, 245-258, July 1998.

[9] Altilar D T, Paker Y, Minimum Overhead Data Partitioning Algorithms for Parallel Video Processing, DD11 - Eleventh International Conference on Domain Decomposition Methods, The University of Greenwich, July 1998.

[10] ADSP--21160 SHARC DSP DSP Hardware Reference, Analog Devices, Mass. USA, November 1999.

[11] [ADSP--21160 SHARC DSP DSP Microcomputer Data Sheet Revision 0, Analog Devices, Mass. USA, 2001.

[12] Altilar D T, Paker Y, Sahiner A.V., A Parallel Architecture for Video Processing, Proceedingsof International Conference on High Performance Computing and Networking (LNCS - 1225) Vienna Austria, April 1997.