KUMP/D: the Kyushu University Multi-media Processor

Taniguchi, Rin-ichiro
Department of Intelligent Systems, Kyushu University

Tomiyasu, Hiroshi
Department of Intelligent Systems, Kyushu University

Kawano, Tetsuo
Department of Intelligent Systems, Kyushu University

Amamiya, Makoto
Department of Intelligent Systems, Kyushu University

KUMP/D: the Kyushu University Multi-media Processor

Hiroshi Tomiyasu   Tetsuo Kawano   Rin-ichiro Taniguchi   Makoto Amamiya

Department of Information Systems, Kyushu University
6-1, Kasuga-koen, Kasuga, Fukuoka 816 Japan

Abstract

High speed image and video processing is a key technology in multi-media applications, and, therefore, currently many hardware accelerators to speed up such processing are developed and used. However, for the next generation advanced multi-media applications in the next generation, such as high quality virtual reality, bidirectional visual interface, etc., the hardware accelerators can not deal with tasks involved in these applications. This is because these tasks consist of complex and irregular computation structures, and, therefore, it is not easy to implement simple hardware accelerators for these tasks.

Considering the above situation, for the next generation multi-media applications, we have been developing a MIMD based multi-media processor, KUMP/D(Kyushu University Multi-media Processor on Datarol-II). KUMP/D is a flexible parallel processor, based on fine grain parallel processing, which is which indispensable in complex and irregular computation, and is also equipped with a specialized I/O network. This I/O network has enough throughput for real time video I/O, providing a mechanism, which supports the synchronization of process executions and real time video frames.

1 Introduction

In multi-media applications, huge computation power is required, especially in image and video processing. To satisfy this requirement, hardware accelerators for image/video processing have been usually used. However, these hardware accelerators are neither flexible nor powerful enough for the next generation multi-media applications, which require advanced bidirectional visual interfaces consisting of complex computer vision and computer graphics algorithms, while the hardware accelerators only provide simple functions such as decoding of compressed video image, image filtering, polygon rendering etc.. Therefore, we should develop a high-performance computing system incorporating advanced real-time computer vision and computer graphics. To achieve this goal, we are developing a MIMD-based multi-processor with special emphasis on high speed processing of image and video streams, because we believe that only MIMD-based multi-processors, especially highly parallel ones, can effectively perform advanced multi-media applications, which involve very complex image and video data handling.

According to these considerations, we have been developing a multi-media oriented multi-processor called KUMP/D (the Kyushu University Multi-media Processor on Datarol-II). KUMP/D is based on Datarol-II architecture [1], which is especially designed for fine grain parallel processing. Fine grain parallel processing is indispensable in advanced multi-media applications, such as "real" [1] virtual reality, in which interactions among many objects in both a real and a virtual world should be calculated and visualized. This is because, to accomplish such computations, complex and irregular, or highly decentralized and autonomous, algorithms should be realized, and these algorithms can be efficiently executed only by fine grain parallel processing. On the other hand, the simple image processings, which have regular computation structures, can be efficiently performed by a simple data parallel architecture and do not require fine grain processing. In KUMP/D, each PE is designed to efficiently execute *threads*, or fine grain processes, with the aid of an efficient thread synchronization mechanism, FMP (Fine grain Message Processor). In addition, KUMP/D has a low latency inter-PE network, which is especially designed for fine grain message communication among threads. Because of these features, KUMP/D provides the high computation power of fine grain parallel processing.

In multi-media processing, the key point of a MIMD machine is the synchronization of a process execution in each PE with a video signal, because in a MIMD-based machine, each processor executes its program asynchronously. The hardware accelerators, on the other hand, are driven by video signal, i.e., their execution phase is controlled by the video signal based clock, and, therefore, there is no need to synchronize the PE execution and the video signal. To solve the synchronization problem, KUMP/D has a specialized high-throughput I/O network, which supports real time video input/output and the synchronization of video frames and processes. In this I/O network, one video frame is divided into a number of fixed size packets, and each PE uses the transmission timings of packets to synchronize every video frame and process. In this paper, we give an overview of KUMP/D, especially illustrating its I/O system design.

[1] "real" here means high reality and high interactivity.

2 Advanced multi-media applications

Before describing our KUMP/D, we discuss some important features of multi-media oriented processors for the next generation advanced multi-media applications. Present popular multi-media applications include video on demand, electronic libraries, simplified virtual reality, etc.. Most of these applications rely on computing power for graphics, such as DCT (discrete cosine transform) in decoding a video stream compressed into a JPEG/MPEG format, translation of axes for polygon rendering, etc., which are fixed and simple algorithms and can be realized by relatively simple hardware mechanisms.

However, as the next generation advanced multi-media applications will become more complex, following three issues will become quite important:

- **high quality virtual reality based on precise visual simulation of real and virtual spaces**
  In both these spaces, the behavior of a lot of objects working under physical laws and their complicated interactions should be calculated. Naturally, such calculations become so complex that to perform them, not only huge computation power but an efficient execution environment allowing for irregular computation structures, such as decentralized and autonomous systems, are required.

- **bidirectional visual interface based on computer vision techniques**
  To simplify the interface between a man (or the outer world of a computer) and a computer, especially when a computer acquires information about the outer world, computer vision techniques are indispensable. In addition, they are also necessary to compress visual data into symbolic data, which extremely reduces the communication overhead in the communication network. Typical examples are human body motion understanding, face expression reading, etc.. It is well known that computer vision, not image processing, requires huge computation power for complex computations.

- **interactive navigator for based on artificial intelligence techniques to assist users**
  This helps non-expert computer users to explore the complicated virtual world and retrieve required information from the enormous amount of stored information. Of course, AI requires massive computation power for huge and complex computations.

One of the most important requirements common to the above issues is the "interactivity" of the system. To achieve high interactivity, the latency of interaction, i.e., the delay between user's action and computer's response should be as small as possible. In displaying images, for example, latencies from user's inputs affects visual reality, and, if these latencies are too large, users will suffer from motion sickness. To avoid these problems , the large amount of computing power needed for the complex and irregular computation structures involved in such tasks as visualization of virtual space and computer vision is an indispensable requirement.

In addition, to achieve high interactivity, a system can efficiently compute interactions among a lot of real world objects, which are observed through interfaces, or sensors, to the real world, and virtual world objects. The behavior of those objects and their interactions are represented in a very large number of concurrent processes, whose efficient executions require a system suitable for fine grain parallel processing, because fine grain processing can reduce the overhead required for process switching, thus directly increasing the interactivity of each process.

According to the above considerations, we have concluded that, for the next generation advanced multi-media applications, we should develop a computing system which has huge computation power and an efficient fine grain parallel processing mechanism. KUMP/D is the realization of our goal, and consists of a lot of processor elements based on the Datarol-II architecture, which provides efficient fine grain parallel processing.

3 KUMP/D

3.1 Overview

Fig.1 shows an overview of KUMP/D. Each processor element(PE) is based on the Datarol-II architecture [1], which is especially designed for fine grain parallel processing. The PEs are connected with each other via a 2-D torus interconnection network. This interconnection network is specialized for fine grain message communication, which includes parameter passing and result value returning. This network has a deadlock-free structure based on its hierarchical packet buffers.

Because the interconnection network is not suitable for long I/O packet transmissions, KUMP/D has a specialized I/O network, which consists of serial links connecting the PEs in a ring structure, for communication of long packets such as video I/O packets and disk I/O packets. One important feature of the I/O network is that it supports a mechanism of the synchronization of each video frame and process execution. KUMP/D has a two-level frame buffer. Each PE has a dual port video RAM, and image data in the video RAM is transmitted to the external frame buffer in the External Video Controller (EVC) automatically through the I/O network. EVC supports a mechanism to change the mapping of image onto PEs.

3.2 PE(Processor Element)

Fig.2 illustrates the structure of a PE of KUMP/D. In parallel processing, one of the most critical issues is the latency problem caused by remote memory accesses and remote procedure calls. To solve this problem, fine grain parallel processing is quite effective.

Each PE of KUMP/D, based on the Datarol-II architecture, executes fine grain processes called *threads*. Here, a thread is a sequence of instructions which can be executed without any suspension caused by instructions involving a long latency period, such as remote

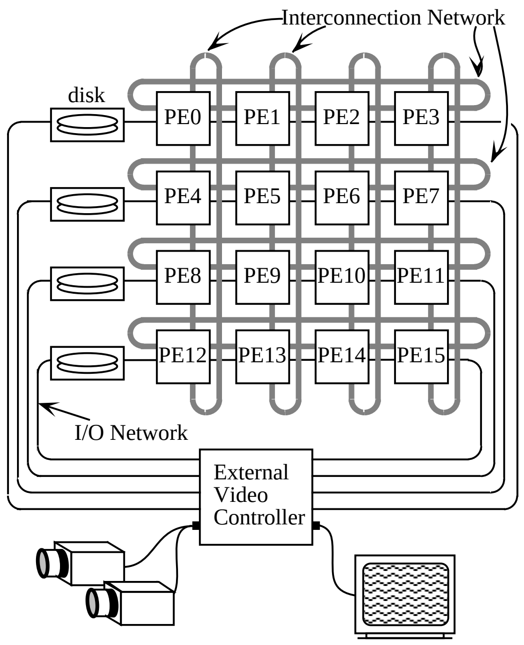

Figure 1: Overview of KUMP/D

memory access. When the parameters of a thread which are required for its execution become available, the thread becomes executable.

As shown in Fig.2, we implement this architecture using a combination of a commercial micro processor, which is referred to as CPU in Fig.2, and an FMP (Fine grain Message Processor). This is because recent high performance micro processors are inexpensive, and can execute thread codes efficiently. In many cases, these processors are faster than custom-made processors designed and developed for specialized applications. However, these commercial micro processors do not have a supporting mechanism for efficient thread activation, which is the most important problem in fine grain parallel processing. Therefore, we have designed the FMP to support the efficient thread activation including message passings among threads. The FMP is mapped in a memory space of the CPU, and when issuing a message, the CPU writes a set of data to a memory address where the FMP is mapped. To reduce the traffic of the memory bus, which is caused by FMP activity, a secondary cache is provided, and the working areas of both processors are separate.

The mechanism of thread activation implemented in the FMP is as follows:

- A thread is executable, when all of its referred variables are available.

- The FMP manages structure data called "Key", each of which consists of a counter showing the number of unavailable variables and an entry point of a thread code. The counter is initialized to be the number of required variables for the thread execution.

- When a variable is written, the CPU sends a "Fire" command, which includes the identifier of a "Key", to the FMP.

- When the FMP receives a "Fire" command, it retrieves a "Key" by the identifier in the command, and decrements its counter. If the counter becomes zero, or if all the variables referred to by a thread become available, the FMP puts the entry point of the thread into the executable thread queue.

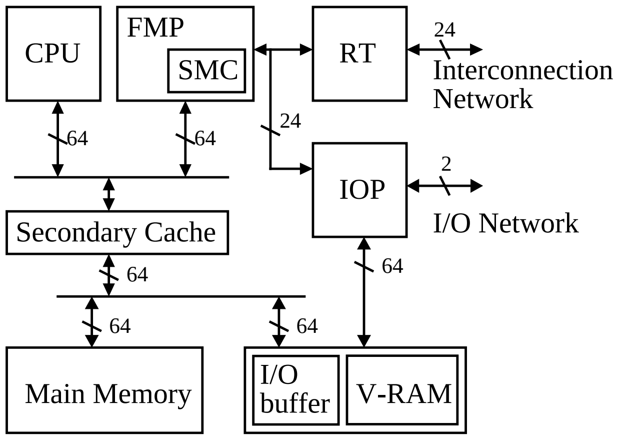

Figure 2: Processor Element(PE)

FMP also directly handles messages from the interconnection network, because the logical structure of these messages is the same as that of messages among threads activated in the same PE. When a PE sends a message, the CPU sends a "Make message" command to the FMP simply by writing few words to the memory. The FMP examines the destination of the message, and if the destination is outside of the PE, the FMP automatically makes a packet and transmit it to the interconnection network. It is not necessary to call complex and heavy message handler, and, therefore, the communication through the interconnection is quite efficient: the latency is low and the throughput is high. In most MIMD based multi-processors, on the contrary, the message handling is executed based on interruption, which often causes a bottle neck, especially in fine grain message handing.

In addition, KUMP/D is equipped with the IOP, which handles I/O issues independently of the CPU. The IOP is also mapped on the memory address space of the CPU, and, to issue an I/O request, the CPU simply writes a set of data to the IOP. The IOP supports real time video I/O, which is one of the most important features in multi-media processing. The IOP transfers, with high speed, image data between dual-port video RAM built in a PE and the EVC. Another important feature of the IOP is the synchronization of process execution with video signal, which will be described in a later section.

3.3 Interconnection network

For the scalability, as the inter-PE network of KUMP/D, we have chosen a 2-D torus network[3], which is also suitable for mapping images on PEs. In KUMP/D, most of the packets such as parameter passings are short, and their communication patterns are relatively random. Actually the maximum size of a packet is only 72bits long. Therefore, the interconnection network of KUMP/D is specialized to handle these short and random packet communications, and its packet buffer size has been reduced to minimum. Since the packet buffer is customized for these short packets, long packets cannot be handled. The long size packets which are required in transferring image data are supported by the I/O network. We will discuss the details of I/O network in a later section.

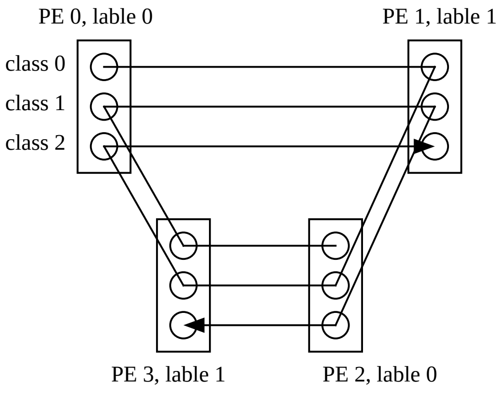

Figure 3: Hierarchical Packet Buffer

One important feature of this interconnection network is that it becomes deadlock-free due to the hierarchical packet buffers shown in Fig.3. This structure can be realized because the size of the packet buffers can be minimized, and actually each PE has only 9K byte packet buffers. The structure of a hierarchical packet buffer and that of the router chip of this network are shown in Fig.4. In X-Y routing, the number of the packets routed in the X direction is larger than that of the packets routed in the Y direction. Therefore, the router of the interconnection network has three pseudo-ports in the X direction. In the prototype version of KUMP/D, the packet buffer size is reduced by using a negative hop algorithm[4]. These buffers produce multiple paths, therefore, conflict among packets does not occur and the interconnection network can avoid deadlock.

3.4 Structure memory

KUMP/D has I-structure memories, which support several important high level memory access methods, including simple one read/one write accesses, stacks,

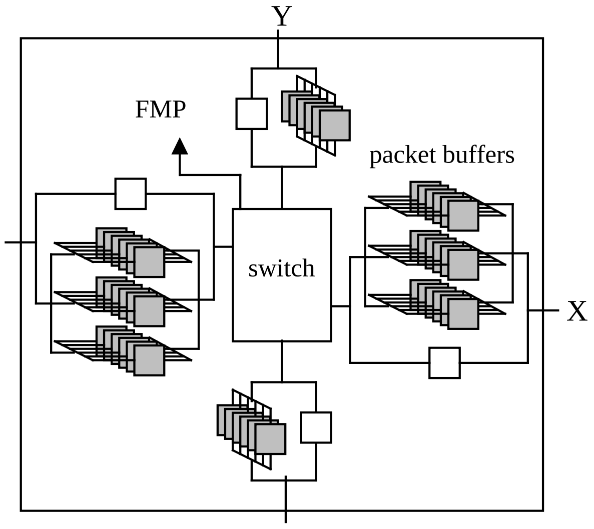

Figure 4: Structure of Router

and queues, which are indispensable in efficiently handling complex data structures. Particularly queues are quite useful for multi-media tasks like video stream processing. In the I-structure adopted in KUMP/D, each memory cell has three tag bits and one special bit. The three tag bits indicate the state of the memory cell: *empty* or *valid*, *list* or *element*, and *pending read-request* or *waiting for read-request*. Using these tag bits, the Structure Memory Controller (SMC) supports the structures of one read/one write accesses, queues and stacks. The special bit is used for locking operations, which is used when the SMC executes exceptional operations and inhibits ordinary memory accesses.

All accesses to the structure memory are originated by the memory access packets. When a read packet is received, the SMC checks the tag bits. If the tag bits show that the cell has valid data, the SMC reads the data and returns two packets: one of them contains the data stored in the structure memory; the other is the "Fire" command packet. If the tag bits show the cell is empty, the SMC stores this read-request. Packets for write memory accesses are handled similarly. When a write packet is received, the SMC examines the tag bits, and if the cell has already stored a read-request, the SMC sends the data to the requested address stored in the cell and sends a "Fire" command in order to activate a thread issuing the read-request. If the cell is empty, data in the write packet is written into the cell.

3.5 I/O network

Since in multi-media processing the performance of I/O operations is quite important, we have designed the I/O network of KUMP/D, considering the characteristics of its I/O operations, to achieve high speed I/O operations[2]. The characteristics of the I/O operations considered are as follows:

- Almost all the I/O data transfers, such as disk I/O, video scan line data transfers, image data transfers of rectangular windows, etc., are block data transfers. In the case of disk I/O, the packet size is at least the size of a disk block. In the case of video I/O, the size becomes a few K bytes.
- Communication patterns are simple compared with messages among threads. Disk I/O packets are transmitted from PEs to disk units or vice versa. Video output packets are transmitted from PEs to the external video controller.
- Output timing of video packets is fixed. That is, video packets must be transmitted at a video rate, and be synchronized with video frames.

If these long I/O packets are transmitted through the interconnection networks, they will cause serious problems, because the interconnection network of KUMP/D is specialized for short packets.

- A burst of long packets require a large band width. Therefore, an interconnection network which handles long packets must have a much larger band width than one which handles only short packets.
- Long packets occupy a long path and block other packets. Since packet buffer size is minimized in the interconnection network of KUMP/D, long packets probably occupy more than two paths, and other packets can not be transmitted through these blocked paths.
- The interconnection network has to guarantee that each video packet reaches the external video controller by the time for its display. This means the priority of video packet transmission is should be high. If video packets are transmitted through the interconnection network, the interconnection network must support a certain priority control, which requires complex protocol and complex hardware mechanisms. They often badly affect the throughput of the interconnection network.

To solve these problems, KUMP/D is equipped with an I/O network in addition to the inter-PE network. We have carefully designed the I/O network of KUMP/D, paying special attention to the characteristics of its I/O operations, particularly the synchronization of process execution with real time video frame, to achieve high-speed I/O operations. Details of these mechanisms will be described in the following section.

4 I/O System

4.1 Overview of I/O Network

The I/O network consists of several ring channels, each of which connects the external video controller, a set of disks, and PEs on the same same row of the 2-D torus interconnection network. Each channel of the I/O network is a high-speed serial line, which is widely used in I/O devices, because the serial link reduces the amount of wiring and makes it easy to handle cables.

Because the communication patterns of the I/O packets are simple, the I/O network operates as follows:

- Each packet is routed in a single direction.
- Each packet flowing in the serial channel contains a destination address designating a PE or an I/O device to which the packet is delivered.
- Each PE or I/O device examines the address part of a packet, and if the address is identical to that of the PE or the I/O device, it takes the packet.
- When the packet taken into the PE is not a broadcast packet, the PE transfers a null packet, which can be used for any data transfer at the succeeding PEs. This is because the packet itself serves as a timing signal for the video signal.

In each PE, I/O operations are controlled by the IOP and the FMP. The IOP writes the packet data into the I/O buffer, and sends a signal to the FMP, which activates threads according to this signal.

4.2 Design Parameters of the I/O Network

In this section, we calculate detail of required parameters of the I/O network based on the following variables[2]. In the following discussion, we suppose image data is equally distributed to each PE.

S	Speed of each serial lines; S[byte/sec].
N_s	Number of serial lines.
W	Amount of Data of video frame; W[byte/frame].
F_w	Frequency of video frame rate; F_w[Hz].

Among the above variables, the following equation must be satisfied, which secure the band width for video data transmission.

$$S > \frac{2WF_w}{N_s}$$

Video occupation rate, VOR, which indicates the proportion of video data amount to the total amount of transferred data in the I/O network, and the capacity of data transmission except for video data, C[bytes/sec], become as follows:

$$VOR = \frac{2WF_w}{N_s S}$$

[2]Actually, there is an overhead, which is caused by packet headers. However, it does not change the result so much, because the size of a header is less than 1 percent of the size of a packet: actually a four-byte header is attached to a 2 K byte packet.

$$C = S - \frac{2WF_w}{N_s} = (1 - VOR)S$$

Since S has a physical performance limit, the minimum N_s is established as follows:

$$N_s \geq \frac{2WF_w}{VOR \cdot S}$$

Actually VOR should be less than or equal to 1. If VOR is equal to 1, there is no room to transfer data other than video data. The smaller the VOR is, the more room for other data transfers in the I/O network, but the required speed of serial link is higher.

4.3 Synchronization with video frame

Since in a distributed memory parallel processor, the processing time of each PE, which operates asynchronously, is different, a mechanism to synchronize process execution with video frame timing is required for real time multi-media processing. In KUMP/D, the external video controller(EVC), which consists of frame buffers and control modules for video data display and video input, constantly sends fixed-size packets through the I/O network channels.

As in Fig.5, one video frame, i.e., one vertical sweep period of the video signal, is divided into M slots, and to each slot one fixed-size packet on the I/O network is allocated. To secure the maximum period of video data processing, the first m packets are reserved for data transfer from an external video input device (such as a video camera) to PEs and the last m packets are reserved for data transfer from PEs to an external video output device (such as a video display). Other packets can be used for any kinds of data transfer such as disk I/O. Even when there is no request for the I/O network, null packets are transferred for synchronization. These fixed-size packets simplify the routing algorithm, and are useful for the synchronization with video frame. In addition, each packet has a current video frame number, which is used for checking errors in real time video processing. In order to execute programs synchronous to a video frame, a mechanism to activate a computation body upon the receipt of a video input packet is realized in KUMP/D.

4.4 Activation Control by I/O Packet

KUMP/D provides data driven program execution on the basis of a mechanism of thread activation by I/O packets. The data driven program execution is quite useful, for example, for those programs in which each PE starts image processing whenever it receives a video input packet. To realize the packet driven mechanism, the IOP uses a thread activation mechanism provided by the FMP.

When the IOP receives a packet, it writes data of the packet into the I/O buffer and sends a “Fire” command to the FMP. If an input request for the packet is already issued, a thread which issued the I/O request is activated by the “Fire” command and read the data in the buffer. If the input request is not yet issued, the thread execution is suspended until the input request is issued. Thus, the thread activation by I/O packets is realized. Since multiple “Fire” commands can be

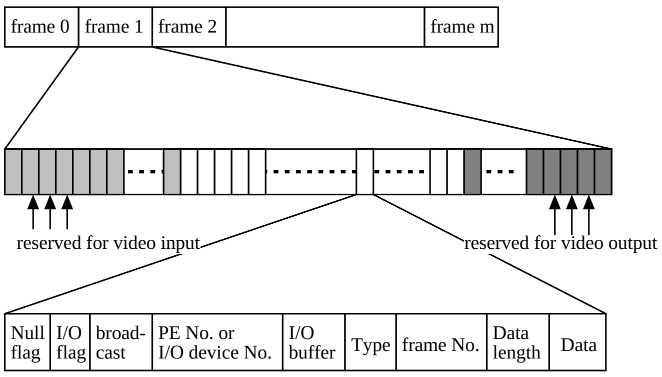

Figure 5: Time Slicing in Video Frame

stored in a queue established in the structure memory, stream I/O data, such as video stream data, can be easily and efficiently processed in each PE without any additional hardware mechanisms.

4.5 Distributed Frame Buffer

KUMP/D supports a distributed frame buffer consisting of dual-port video RAMs built in PEs, and frame data is transfered between the distributed frame buffer and the external video controller through the I/O network. Mapping of a video frame to video RAMs can be reconfigured by the EVC, is configurable. In the prototype version of KUMP/D, the EVC has three sets of frame buffers (see Fig.6): one for video capturing, one for video displaying, and one for auxiliary usage such as translation of the mapping. In addition, the EVC has a synchronization register, which is used for the synchronization of PEs in real time video processing.

Each buffer plane is divided into several banks, each of which is connected to one I/O link via one of the two ports of video RAM. In order to make it possible to re-configure the mapping between an image plane and a frame buffer plane, and equivalently the mapping between an image plane and PEs, a mapping table, or an address translation table, by which, the CRT controller can change the accessing order of video RAM, is provided between the CRT controller and the other port of the video RAM. This table is set at the system configuration time. Some possible frame buffer configurations are shown in Fig.7.

The drawback of the double buffer mechanism is latency, or the delay in displaying processed images, which is four video frames: one for video capturing, one for data transfer from the EVC to the PEs, one for data transfer from the PEs to the EVC, and one for video displaying. Of course, naive processings such as point operations of images can be realized with less delay. Actually, a four video frame delay is not a serious limitation for any actual applications. We have considered that the flexibility is more important.

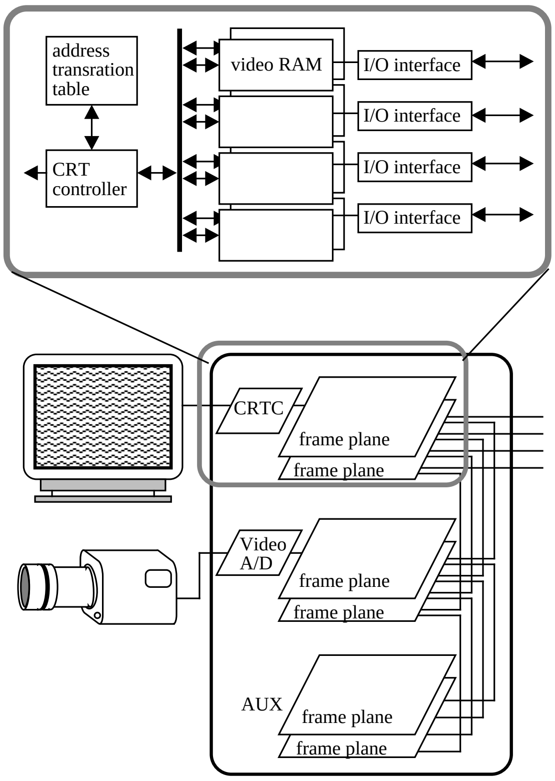

Figure 6: External Video Controller

4.6 Real Time Video Processing

An important issue other than the latency problem described is the throughput of the system. If the speed is too slow, some video frames will be lost. In order not to lose any video frames, processing by PEs should be done within a period as follows(see Fig.5):

$$T_p = \frac{1 - VOR}{F_w}$$

This condition comes from the limitation, that the calculation in each PE should be finished by the time the packet reserved for video output arrives at the PE.

However, if the above condition is not satisfied, i.e., if delays of processing occur, KUMP/D compensates the delay in several ways. In recovering the delay, it is the most important issue to synchronize all PEs with video frames, because the processing time of each PE is different from each other depending on data allocated to the PE. At present, KUMP/D provides the following ways of synchronizing computation at PEs with the video signal.

(1) **No synchronization**
In KUMP/D, each PE which has finished its calculation asynchronously transfers the result to the I/O network. In this mode, the EVC displays the received data, without examining whether all the frame data have arrived or not. In real time video processing, that is, when captured video data are supplied to PEs, the video data is invalidated and is not processed if the PE has not finished processing the previous video frame. Although the visual quality of this mode might not be as good as the other possible alternatives, this is the simplest way.

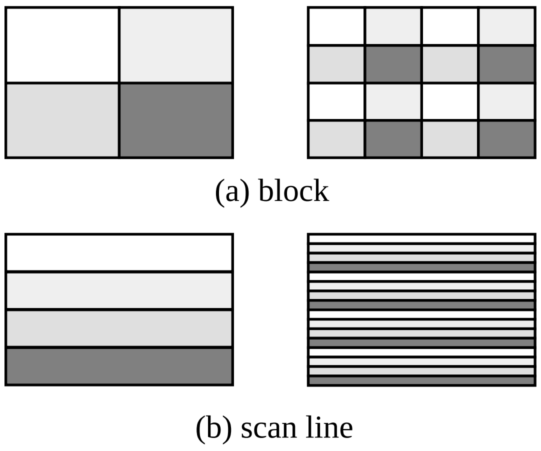

Figure 7: Mapping of an Image on PEs

(2) **Complete synchronization**
In this mode, the external video controller does not switch a frame buffer plane for display until it receives all the data from each PE. The synchronization is achieved by counting the number of valid bits in the synchronization register of the EVC, each of whose bits corresponds to each PE and is set when the frame data from the PE arrives. As a result, in real time video processing, one frame out of several continuous frames is processed and displayed in turn. The EVC stops supplying video data to the PEs until the synchronization is achieved, and, consequently, loses several video frames depending on the computation load of the PEs. This suspension of data transfer is controlled by the synchronization register.

(3) **Progressive Display**
This mode is a mixture of mode 1 and mode 2. As soon as the data arrives at the EVC, the received data is displayed. In addition, until all the data is received, already received data are repeatedly displayed. This is done by the following mechanism:

- The IOP repeatedly transfers the content of video RAM to the I/O network, whether the content is updated or not, until the synchronization is achieved in the EVC. PEs which have not finished their computing transfer null video packets.

- The EVC continuously switches the planes of the frame buffer.
- When the computation in each PE is finished. The IOP sets the synchronization bit in the video output packet. The same synchronization system as that of mode 2 is used.

Several modified methods of progressive display, such as line-wise progressive display and quad-tree like progressive display, can be also realized by modifying the order to update video RAM cells.

It is not clear which mode is most suitable for actual video processing, because it depends on applications and applied video data. We need to investigate the most suitable mode in various applications using various simulations. Anyway, users can specify the most favorable synchronization mode depending on their application.

5 Prototype System Design

At present, we are developing a prototype system consisting of 16, or 4×4, processor elements. The specifications of this prototype are shown in Table 1. A Pentium (66MHz clock) is used as the CPU in the processor element. The FMP and the network router will be constructed with FPGAs in order to be able to examine several alternative methods. The main memory can consist of up to four 72-pin SIMMs. In the first prototype PE, the memory size will be 8M bytes. The size of the secondary cache is 256K bytes, and the video RAM is 1M bytes.

Each path of the interconnection network is 24 bits wide, and the maximum size of a packets is 72 bits, depending on the content of the packets. The clock rate of the interconnection network will be around 33MHz. Right now the video system supported by this prototype is only NTSC: 640 × 480 pixels/frame and 30 frame/sec. Since the speed of an I/O serial link is about 33M bytes/sec, and there are four serial lines, VOR will be around 0.41, in other words, the throughput of the I/O network, except for video I/O, will be around 19.5M byte/sec.

We will support an HDTV level video system in the next version by extending this prototype. It is not difficult to increase the number of PEs to achieve higher performance because KUMP/D has a scalable 2-D torus interconnection network. The only thing that we should re-configure is its I/O system. This is because HDTV requires high speed video data transfer, which possibly causes the lack of the channel capacity of the I/O serial links. In such a case, we should double the number of serial links in a ring, i.e., we should establish two serial lines, or a two-bit parallel link, for each I/O links instead of one serial link. The decision will be made based on a criterion described in 4.2.

6 Conclusion

In this paper, we have given an overview of KUMP/D, an MIMD based multi-processor for multi-media oriented applications. KUMP/D consists of processor elements which are based on the Datarol-II architecture and efficiently handles asynchronous fine grain parallel processes. The interconnection network of KUMP/D has a deadlock-free structure and is especially designed for fine grain message communications.

Moreover, to solve the synchronization problem, caused by asynchronous process execution in MIMD based multi-processors, KUMP/D provides an I/O network, which is specialized for long packet transmission, and which supports disk I/O and real time video I/O. In this network, one video frame is divided into fixed-length packets. Using these simple fixed-length packets, the I/O network secure enough band width to handle the video stream, and supports the synchronization of video frame and thread execution.

Because of this approach KUMP/D has high throughput and high flexibility necessary for the next generation multi-media applications, which require complex computer vision and computer graphics algorithms. Currently we are developing a prototype system of KUMP/D with 16 processor elements. Hereafter we will precisely evaluate its performance in various actual multi-media tasks.

Number of PEs	16
Structure of the interconnection network	2-D torus
CPU	Pentium(66M Hz)
Main memory of each PE	8 [M byte]
Throughput of the interconnection network	99 [M byte/sec]
Structure of the I/O network	ring
Throughput of the I/O network	33 [M byte/sec]
Supported video system	NTSC

Table 1: Specifications of Prototype

References

[1] M. Amamiya, and R. Taniguchi, “Datarol: A Massively Parallel Architecture for Functional Language,” Proc. SPDP, pp. 726-735 (1990).

[2] A. L. Narasimha, Reddy and James C. Wyllie, “I/O Issues in a Multimedia System,” IEEE COMPUTER, pp. 69-74 (1994).

[3] W. J. Dally “Performance Analysis of k-ary n-cube Interconnection Networks,” IEEE Transactions on Computer, Vol. 39, No. 6, pp. 775-785 (1990).

[4] I. S. Gopal, “Prevention of Store-and-Forward Deadlock in Computer Networks,”IEEE Transactions on Communications, Vol. COM-33, No. 12, pp. 1258-1264 (1985).