1 N－5

The message routing mechanism and kernel system of a parallel graphics accelerator

Tran Cong So Kenji Nakajima Katsuhiro Yamazaki
Faculty of Science and Engineering - Ritsumeikan University

1 Introduction

The VoViAc (Volume Visualization Accelerator) system is a parallel system with distributed memory that is used for Volume Graphics Processing. The VoViAc uses messages to communicate between its processing elements thus message routing mechanism, which affects directly to VoViAc's performance, is an important issue for the VoViAc. We hereby discuss the method that is used to create and route messages in the VoViAc. We also propose the VoViAc's kernel that forms an pseudo-OS while co-operates with other programs in the host computer.

2 System architecture

The VoViAc has 8 processing elements (PEs). Each PE consists of DSP (TMS320C31), local memory (SRAM 256Kbytes) and a router (FPGA XC4010D) (Figure 1). The host computer controls VoViAc's operations. 2D Memory and Display are used to display processed 3D images.

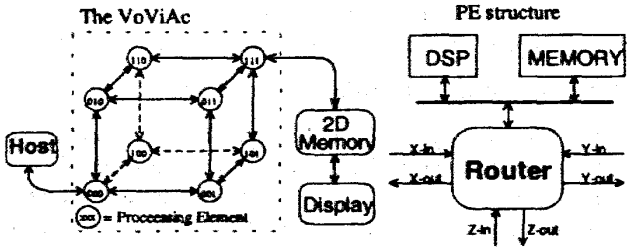

Figure 1: VoViAc architecture

3 Message routing mechanism

Messages carry information between PEs. For processing and routing purpose, a message is created with the following structure (Figure 2):

16	2	4	2	8		
Message ID	T	S	P	Dest Addr	Task	Data

Figure 2: Message structure

Message ID is used to identify the message. T field is the routing Type of the massage. S field is the binary address of the sender node. Destination address field contains 8 bits corresponding to the addresses of destination node(s). P field is the Priority of the message. Task field contains the task that destination node(s) must process. Data field is the data for the task. The message has minimum length at one packet.

Messages are packaged into packet(s) to transmit over the interconnection network (IN) through routers. Messages are later reassembled from packets at the destination node(s). Packet format is shown in Figure 3. Packets have fixed length at 31 bytes. The 1st and 2nd bytes are header of a packet that contains **T, S, P** and **destination address** parameters. The header guides the

2	4	2	8	8	
T	S	P	Dest Addr	PSN	Packet Data

Figure 3: Packet structure

packet to reach the destination. The 3rd byte is always Packet Serial Number (PSN). PSN is used to reassemble packets to form original messages in case the messages required more than one packet to form. The bytes from 4th to 31st are data of the packet.

The Virtual Cut-through Routing (VCTR) (which is mixed from store-and-forward and wormhole routing) scheme is being implemented in the VoViAc. The VCTR scheme allows to reduce the latency and to make dedicated hardware router which isolates the processor from routing workload and, as the result, increases system performance. When collision appears, a decision of which packet will allocate the channel is made depending on packet's priority and the other packets are being buffered or blocked until the channel is free.

The modified E-cube routing algorithm is used to calculate the routing path between the source and the destination. The E-cube routing avoids the deadlock or live lock that may appear during normal routing operation. The next node address $a\{x_a,y_a,z_a\}$ is calculated from present node address $p\{x_p,y_p,z_p\}$ and destination node address $d\{x_d,y_d,z_d\}$ as follows:

$a\{x_a,y_a,z_a\} = p\{x_p+i, y_p+j, z_p+k\}$ with

$i = 1$ if $(x_p \oplus x_d \neq 0)$

$j = 1$ if $(y_p \oplus y_d \neq 0$ and $j = 0)$

$k = 1$ if $(z_p \oplus z_d \neq 0$ and $j = 0$ and $k = 0)$

i, j, k = (0,1) and a, p, d are binary coded addresses

The messages are handled by both the router and kernel. The packet's transmittal over the IN is done by the router while message processing is done by the kernel.

4 The kernel system

The VoViAc kernel is like an operating system for each node except it does not have a file system (Figure 4). During operations, the kernel communicates with a monitor program in host so that forms an pseudo-OS for the VoViAc. The kernel has to supply most basic device drivers, interrupts and functions for applications running on it. At power up, the kernel is the software that the VoViAc runs to check itself and to communicate with the host computer thus a part of the kernel must be contained in ROM devices. The rest of the kernel will be loaded

from the host computer. With the small size of system and memory, the kernel must be **as small and fast as possible**.

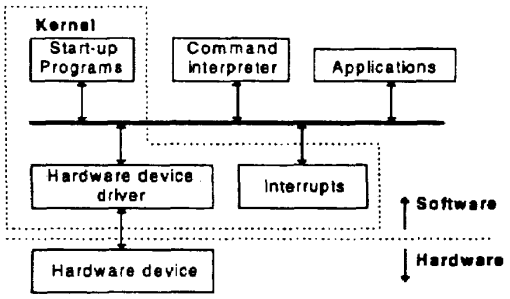

Figure 4: Software environment of VoViAc.

4.1 Start-up programs

At power-up, the start-up programs are executed to setup the VoViAc. The start-up process includes several specific procedures:

- Node setup procedure configures the hardware of each node, checks hardware functionality and indicates failures code if any for each node.
- System setup procedure checks the IN statuses in all nodes to create overall status variables.
- Kernel loading procedure loads the other part of kernel from host. It also completes environment variables such as interrupt table and software environment statuses. At last, this procedure loads the command interpreter into the VoViAc.

4.2 Hardware device drivers

Hardware device drivers are used to isolate the hardware and application programs. In the VoViAc kernel system, we have the following device drivers:

- ***Router device driver (RDD)***: This driver supports for interfacing with the router. This driver exists in all nodes. An application interacts with router device driver to send and receive messages over the IN and moreover do synchronization between processes.

In sending, an application sends a message request to the RDD. The RDD creates the message structure and sends the message to the bottom of outgoing queue. Meanwhile, the RDD packages the top message into packet(s) and orderly sends packets to the router. The router then sends packet(s) over the IN to destination. In receiving, the router receives packets and sends them to the RDD. The RDD reassembles the original message from packets with information in the header and the PSN and then puts it to the bottom of incoming queue. Meanwhile, the RDD processes the top-level message and creates an event to the application.

In application point of view, the RDD provides a message passing interface for application. The RDD supports for blocking (synchronous) and non-blocking (asynchronous) message send and receive. In blocking mode, the RDD waits for the completion of sending or receiving then return control to the calling process. On the contrary, in non-blocking mode, the RDD returns control to the calling process before the actual sending or/and receiving has been completed.

The RDD supports for point-to-point and collective communication between nodes. Point-to-point communication does communicate between a pair of nodes while collective communication takes place among all nodes. Collective communication includes: one-to-all, all-to-one, gathering, scattering, all-gather and all-to-all. Also, the RDD has ability to do barrier synchronization among processes. For the specific purpose of the VoViAc, the RDD does not support for topology reconfiguration, group and context of processes.

- ***Host communication device driver***: Only the root node has to use this driver to communicate with the host computer. This driver supports the command interpreter program in VoViAc to communicate with the monitor program in host to provide application loading, debugging, test and benchmark. Also, this driver supplies a simple file system for applications (while co-operates with the monitor program).

4.3 Kernel's Interrupts

The kernel has several functions supporting for applications. These functions are provided by interrupts. Most of hardware interrupts are used to implement the above drivers. Some other important functions are implemented by software interrupts completing the kernel. The application initiation and termination interrupt is used to initiate and terminate an application. The memory management interrupt supports for memory allocation, memory de-allocation, memory de-fragment, etc. The kernel also allows some software exception handling such as overflow, divide by zero and so forth.

5 Conclusion and further research

The VoViAc system is only a graphics accelerator so that its size is small and it requires the highest speed. The way, that we tried to implement message passing mechanism and the kernel, deals with these problems. The message structure, routing scheme and routing path calculating algorithm is chosen for implementing a fast and simple router with which we can pass almost routing workload to hardware. The kernel was designed to fit into small memory of the VoViAc but still supports for most of important functions and guarantees simple and reliable operations. Currently, we have been finishing the router design (on FPGA, using Mentor Graphic EDA tools), the host communication driver and setup programs. Also, we are under way programming the kernel (RDD) and the monitor program in the host computer. Further research needs to be conducted on problems of the kernel such as protecting environment for application, data type supporting for message passing interface and so forth.

References:

[1]Y.Yamazaki and K.Yamazaki: Design of a Volume Visualization Accelerator. IPSJ 51st National Convention Record, 4S-11, 1995.

[2]T.C.So, K.Nakajima and K.Yamazaki: Message in VoViAc system: Structure, routing and handle. The Kansai-section Joint Convention of Institutes of Electrical Engineering Record, G304, 1997.