Abstract

For improving the performance of distributed processor systems (DPS), the communication protocol among processors is very important and has much room to improve. In this paper, we issue the novel protocol and algorithms of communication, and data-receive and data send, respectively, to implement high-speed data transmission between processors of the DPS to archive and application the distance learning systems, respectively.

Keyboards: distributed processor systems (DPS), communication network, discrete topology, and communication handshaking.

1. Introduction

The quality of the distributed processor systems (DPS), also called tightly coupled system, has a decisive impact on the speed, size and cost of the whole architecture. In distributed processor system any processors must be able to access any memory location, even if it physically belongs to another processor. Thus, the communication interconnection schemes are usually employed. The communication machine for distributed processor system allows the messages to be transmitted between execution units of different processor [4,5,8].

The communication interconnection network can be further classified into two types: signal bus and multiple bus systems depending on the number of buses providing interconnection among the processors, called node, of the distributed processor system architectures.

In this paper, we proposed a communication protocol, which is constructed by the multiple buses, to the DPS. The communication latency time is the one of benchmarks of interconnecting processors of DPS [5,7]. The communication delay time is defined to the total time of the request is issued to the desired data are received. In fact, the communication latency imposes a limiting factor on the scalability of the communication mechanism size. Thus, the number of processors of DPS cannot be increased indefinitely without exceeding the tolerance level of the access latency. For a DPS, the desired data of a processor may be accessed through two sources, the local memory (called P-to-M or P2M) or other processor (called P-to-P or P2P) [1].

A communication network can be represented by a communication graph. The topology of the communication graph is an important property, which significantly influences latency on multiprocessors. Thus, there are two-communication models, message-passing model and share-memory model, are discussed and analyzed in this paper. For the message-passing model, the information is exchange through an inter-process communication facility provided by the communication routine [2,3]. In the share-memory model, processes use map memory scheme to gain access to regions of memory owned by other processors.

The recent of this paper is organized as follows. Section 2 describes the protocol of communication for DPS. The communication latency and performance of DPS is analyzed in Section 3. Finally, we remark the conclusions in Section 4.

2. The Protocol of Communication

A DPS is a collection of tightly coupled processors interconnected by a communication network. From this point of view of a specific processor in a DPS, the memory is divided several modules and shared among processors, the rest of the processors and their respective resources, such as data cache, are remote. The designer of a communication network must address four basic issues:

1) Naming and name sources: such as controller, processors, and storage devices, how to process and locate each other to communication?
2) Routing strategies: how are messages sent through the network?
3) Collecting strategies: how do two processes send a sequence of messages?
4) Contention: the network is a shared resource, how do we resolve conflicting demands for its use?
Communication control can take place only if the correct processor access rights are granted. Each processor has a corresponded message queue, which holds a sent message, in the bus decoder of this paper. When the message queue is filled, the sequentially messages are stalled by this message queue. In the communication network architecture, the processors are classified into the requesting processor and the requested processor. The requesting processor, also called receiver, can do one of the following things:

1) receiver can send the request to the bus arbiter while data dependency occurs,
2) receiver can block until the message is received successfully,
3) receiver can ask the bus decoder to hold it temporarily, and
4) receiver can give an error message while the desired data losses.

For one processor of the DPS will get data from other processors. Firstly, The requesting processor must send a request message, which contains the address of the desired data, to the bus arbiter, then to the bus decoder, furthermore to the requested processor. The request message is transmitted to the requested processor via the communication network. If the requested processor is busy, the request message is blocked in the request message queue until the requested processor is free.

The requested processor receives the request message: the requested processor must send a reply signal to the requesting processor. The above process among processors is called handshaking of communication. Implement the handshaking process, then the desired data is transmitted from/to processor through the data queue of the bus decoder. The desired data of the requesting processor may be transmitted from the shared memory or the requested processor, which are called p2M (processor-to-memory) or p2p (processor-to-processor), respectively. The communication protocol of the p2p and p2M are illustrated in Figure 1(a) and Figure 1(b), respectively. The algorithm of communication process of the DPS for this paper is illustrated in Figure 2, too.

The communication machine of this paper is the combination of bus arbiter and bus decoder. The bus decoder is constructed of request buffer and data buffer for storing the desired data from the requested processor, request message register for registering the request when the requested processor is block or busy, and the controller for initializing the requested processor to access the desired data from its local memory. This communication machine allows transmitting the message and data between the executable units of processors.

The structure of the request and data buffer are FIFO (first in first out) architecture to hold the request message and the desired data of the requesting processor. In practice, the buffers can only be a finite length and a point could be reached when the contents are help up because all the available buffer space has been exhausted.

The algorithm of the communication protocol of DPS is shown in Figure 2. For a receive routine, the desired data has been received of the requesting processor if it has been sent to the data buffer of the communication unit. If the receive routine of the requesting processor reaches before the send routine of the desired processor, the data buffer of the communication will be empty and receive routine must wait until the desired data is sent to the data buffer. For the send routine of the desired processor, once the local actions have been completed and the desired data is safely accessed on its way, the process of the requesting processor can continue with the subsequent work and instruction. In this way, we use such receive and send routines interaction between requesting and desired processor to decrease the overhead of the communication time.

In this paper, we will only discuss with the communication protocol of the processor-to-processor (p2p). Let us examine the...
basic message parameters associated with sender and receiver system calls. The message-passing primitives are specified by:

- Create and delete the request buffer and data buffer of the communication unit via the request and data_receiver routines.
- Temporally, storing the transfer status information of requesting processor to the request buffer of the communication unit.
- Send the request signal and receiving the desired data among processors.
- Attach and detach remote processors, such as requesting and requested processors.

The send and receive primitives of the communication unit are implemented by the sending and receiving process, respectively. Therefore, the buffer field in send specifies the memory location of the data to be retrieved from the requested processor. On the other hand, the buffer field in receive specifies where the arriving data will be stored before to be read by the requesting processor.

```c
/* The algorithm of interconnection processor communication network*/
Repeat:
    requesting processor has an request item in next-tp of the request buffer of communication unit;
    /* sending the request message to the desired processor */
    send (desired processor, next-tp);
    until false;
    the requested processor process is defined as the needed for the requesting processor;

/*receiving the desired data from the data buffer for the requesting processor*/
/* receive routine*/
receive (requesting processor, next-d);
    requested processor send the data item to the next-d of data buffer;
    until false;
```

Fig. 2. The algorithm of inter-processor.

```
/* data transmission */
send ( )
{
    data_transmitter:
        wait;
        if (data is ready for the requested processor and received the signal of request)
            { data_id is set to active;
                access the desired data from local memory of GPR of requested processor;
                send the desired data to data buffer of the communication unit;
                data_ip = data_ip+1;
                if (data_ip != 0)
                    goto data_transmitter;
                else end;
            } else wait;
        end:
        data_id is reset to valid;
}
```

Fig. 3. The algorithm of data send.

```
/* data receive */
recv ( )
{
    data_receive:
        wait;
        if (data queues in the data queue of communication)
            { the flag of data_valid is set to 1;
                the signal of the received is ready for the requesting processor;
                receive the desired data from data buffer of the communication unit;
                forward the received data to the general-purpose register (GPR);
                data_ip = data_ip-1;
                if (data_ip!= 0)
                    goto data_receiver;
                else end;
            } else wait;
        end:
        the flag of data_valid is reset to 0;
}
```

Fig. 4. The algorithm of data receive.

The structure of proposed DPS is shown in Figure 5, which is consisted of n numbers of processors, m numbers of memory-modules, b numbers of buses, and the communication unit, included the bus arbitration and the bus decoder.
The desired data of the requesting processor may access from one of two paths, other processor via remote produce call (RPC) or shared memory (memory module) via direct-access memory (DMA). In this paper, we only discussed with the data transmission between processor to processor. Thus, we also represented the state transfer machine of the p2p in Figure 6.

If the desired data is accessed from other processor (i.e. requested processor), the communication handshaking starts the place of request data from other for the requesting processor, then initialize to the transition of request queue, furthermore feed to the place of request buffer. If the required processor is busy, the request is temperately queued in the place of request buffer; else the required processor is initiated the transition of request interrupt to access the desired data via the transition of system call.

The desired data are accessed to the requesting processor from the required processor through the transition of received data and the place of data queue. If the required processor does not generate the desired data, the respondent signal of the request is still blocked in the place of ready to receive data, until the requested processor completes the desired data.

3. Performance Analysis

The communication latency is the one of performances of inter-processors architecture. We will analyze the communication latency of this proposed architecture for different number of clients with different size of the request buffer and data buffer. The number of request- and data-buffer are various from 2 entries to 16 entries.

3.1 The Communication Latency of Different Queue-Size

In this section, we compare the communication latency of this proposed DPS in the following cases: 1) the proposed MP has 6 PEs, but with different numbers of request- and data-buffer, 2) the proposed MP has the same size of request- and data-queue, but with different numbers of PE.

Figure 7 shows the communication latency of different numbers of request-buffer in case 1. Referring to Figure 7, we find the lowest latency to cost ratio, is defined to the buffer size is divided to the reduced time ($T_{\text{latency}}$) of each PE is 8-entry, in average. This reason is caused that the request is issued from requesting processor to communication unit, then the communication unit issues an interrupt single to the requested processor, Thus, any requests must expended waiting time in the communication unit. When the request queue size large than 8-entry, such as 12 or 16 entries, the latency time not conspicuously improve for the cost.

Figure 8 shows the communication latency of different numbers of data-buffer in case 1. Referring to Figure 8, we find the lowest latency to cost ratio occurrence in 4-entry. This reason is caused that the desired data of the requesting processing is accessed from required processor to the data-queue of communication unit, then transmitting to requesting processor, immediately. Thus, the desired data not expend any waiting time in the data queue of communication unit.

3.2. The Performance for Different number of Clients

Figure 9 shows the communication latency of different numbers of processor in case 2, which has the same capacitance as request-buffer size and 4-bus. Refer to Figure 9, we find the lowest communication latency while the processor
number is 8. This reason is due to the probability of each bus to be used by processors is 0.5. Thus, the request commands data of the requesting processor do not expend arbitration time to process, and lower time to process handshaking control among requesting processor, communication unit, and requested processors.

Fig. 7. The communication latency of different request-buffer size.

Fig. 8. The Communication latency of different data-queue size.

Fig. 9. The Communication latency of different numbers of Processors with the same of buffer sizes under 4-bus.

5. Conclusions

in this paper, we issue the protocol of communication for the distributed processor system. We also propose the algorithms of data receive and data send when this proposed architecture is exploited to the distance learning system. In order to verify our assumption is realizable, we simulate this proposed architecture in a discrete event system and provide it had improved the performance of the DPS.

References


