(12) UK Patent Application (19) GB (11) 2 148 563 A

(43) Application published 30 May 1985

(21) Application No 8426840

(22) Date of filing 24 Oct 1984

(30) Priority data
(31) 8328396 (32) 24 Oct 1983 (33) GB

(51) INT CL⁴
G06F 13/14 H04L 11/08

(52) Domestic classification
G4A FG

(56) Documents cited
GB 1440103 GB 1392231 GB 1249209

(58) Field of search
G4A
H4K

(71) Applicant
British Telecommunications plc (United Kingdom), British Telecom Centre, 81 Newgate Street, London EC1A 7AJ

(72) Inventor
Christopher G Miller

(74) Agent and/or Address for Service
Vivian E Irish,
British Telecom, Intellectual Property Unit, 151 Gower Street, London WC1E 6BA

(54) Multiprocessor system

(57) A multiprocessor system consistig of a plurality of slave processors 10 and a master processor 11 each connected by a bus to a common memory 12. Message transfers are implemented by the master which scans one-byte impart areas in the common memory—each slave processor only being able to load a pointer to one said one-byte areas.

The common memory has input areas corresponding to the source slave processors, information being assembled into output queues corresponding to the destination processors.

Connection problems are prevented because once a slave processor has loaded information to its one byte area it must pass control of the bus to another processor. The system is fast because the master does not waste time looking at empty memory.

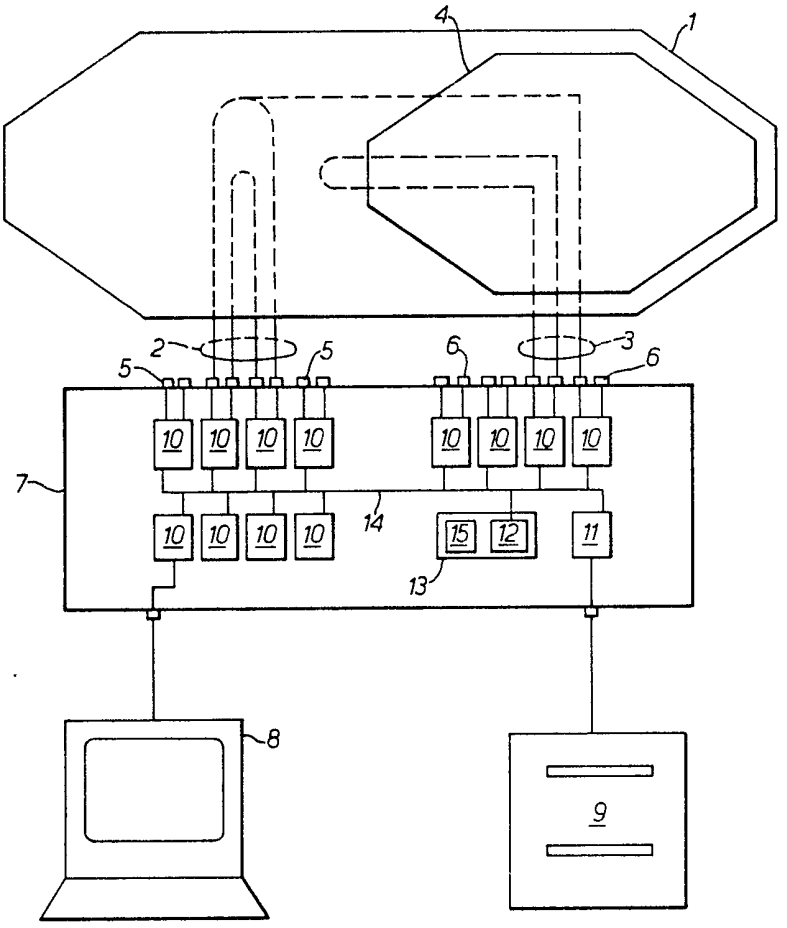

Fig.1.

GB 2 148 563 A

The drawings originally filed were informal and the print here reproduced is taken from a later filed formal copy.

2148563

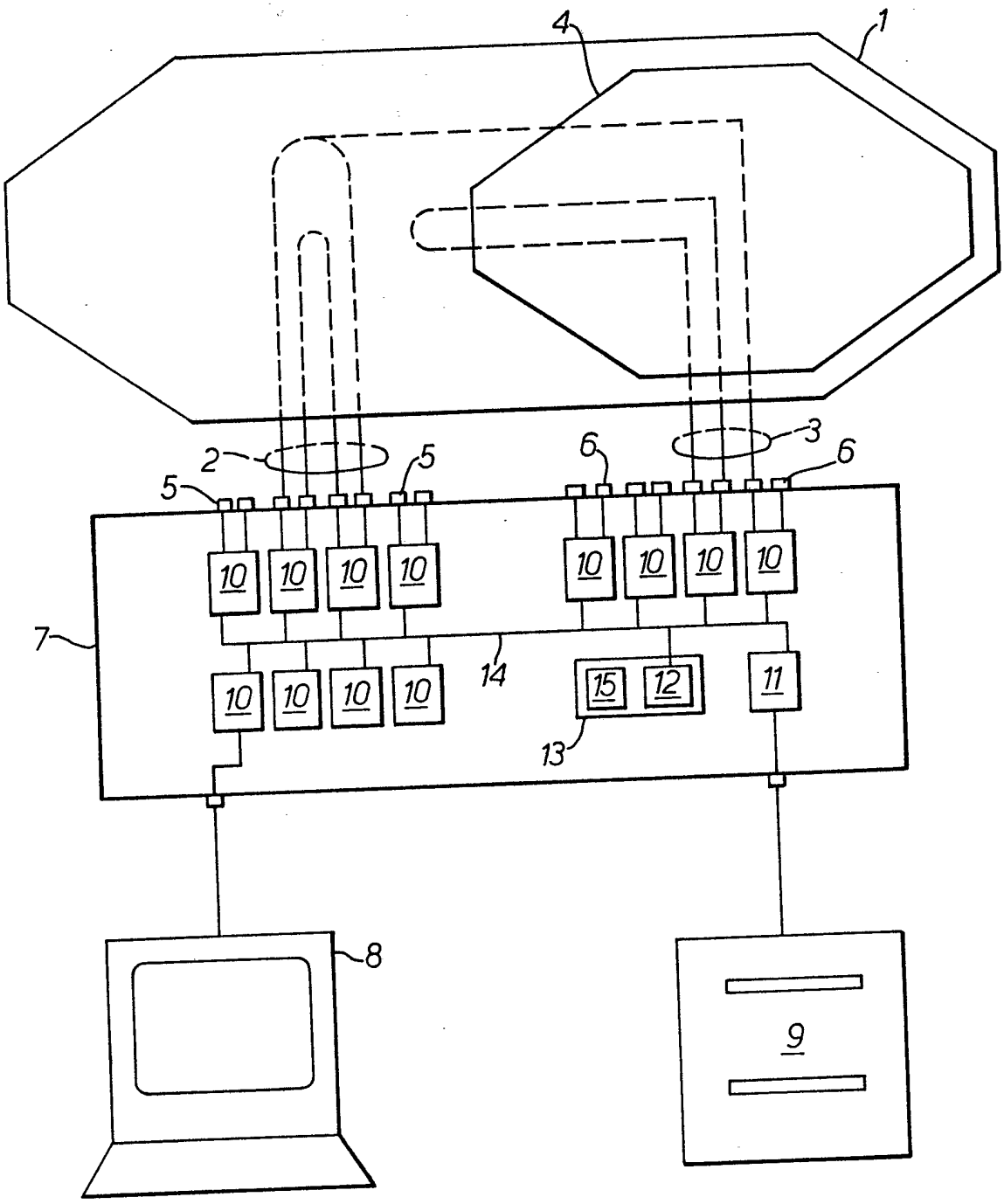

Fig.1.

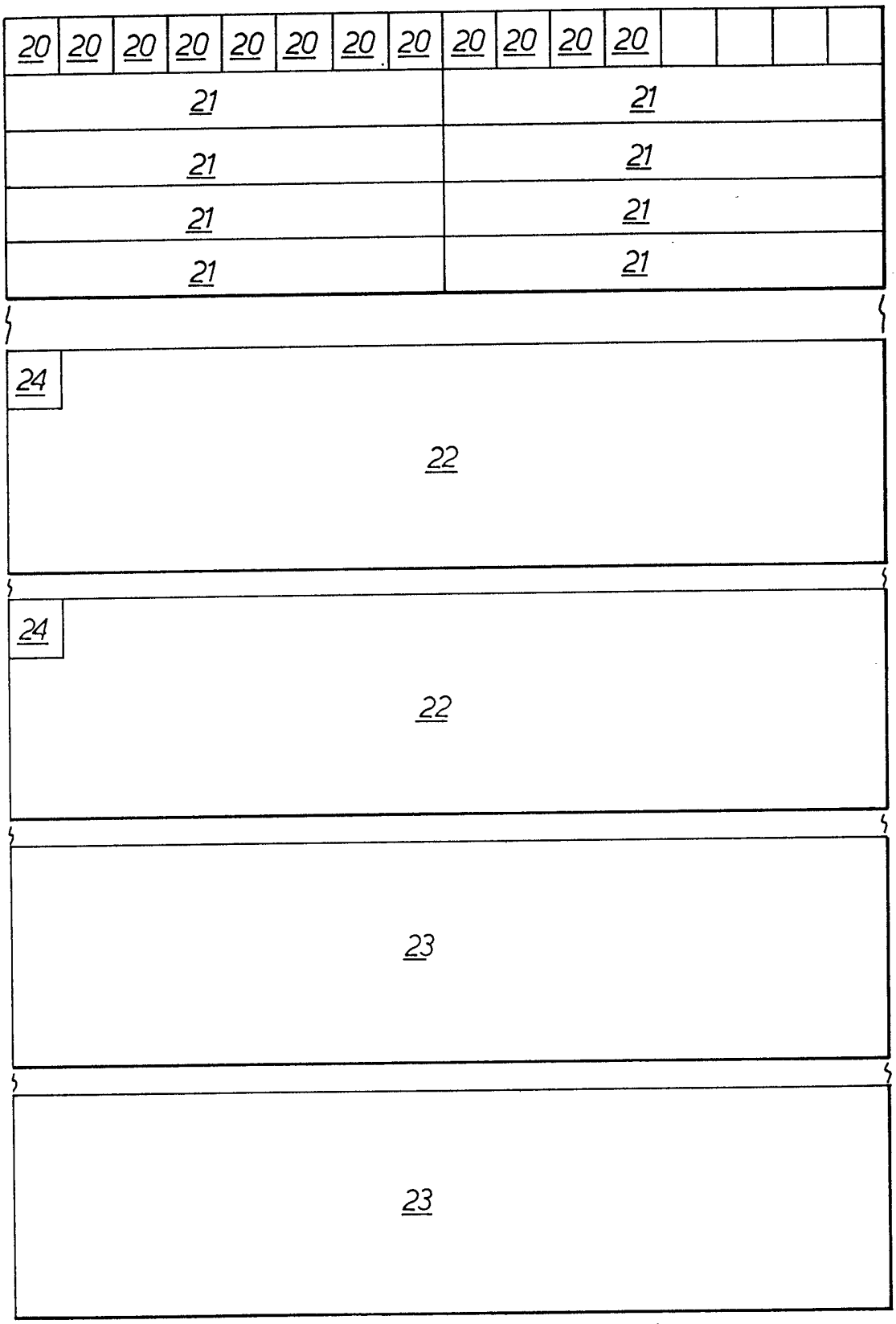

Fig.2.

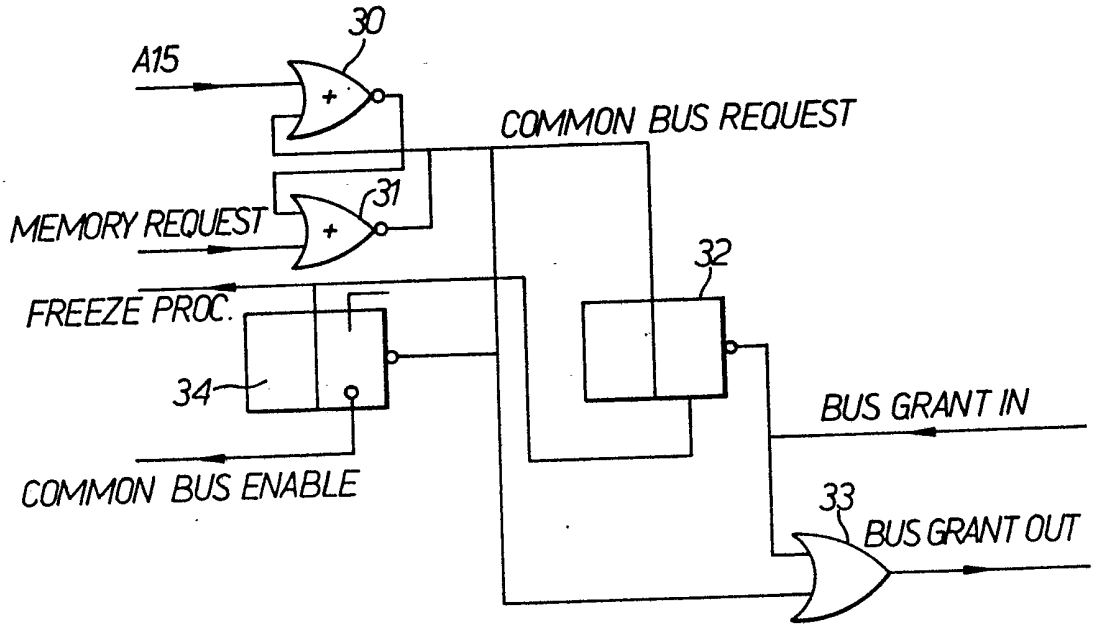
30
A15
COMMON BUS REQUEST
31
MEMORY REQUEST
32
FREEZE PROC.
34
BUS GRANT IN
33
BUS GRANT OUT
COMMON BUS ENABLE

Fig.3.

SPECIFICATION

Multiprocessor system

This invention relates to a multiprocessor system in which a bus connects each processor to a common memory.

Multiprocessor systems employing common memory for the transfer of information between processors are known. For example, in British patent application 2 112 146 a system is disclosed in which information is transferred in response to interrupts. In an article by R J Bowater et al in the IBM technical disclosure bulletin Volumn 22, No 11, of April 1980 a system is disclosed in which areas of memory are scanned in order to locate information which is to be transferred.

To ensure satisfactory operation of the above systems, while making efficient use of the common bus, contention for the bus must be resolved to ensure that all processors have access to the common memory. The rate at which each transfer is preformed (for a given clock frequency) is determined by the number of instructions required to perform the transfer. It is therefore an object of the present invention to provide an improved multiprocessor system which provides fast information transfer while allowing each processor to have access to the common bus.

According to a first aspect of the invention there is provided a multiprocessor system, comprising a master processor, a plurality of independently controlled slave processors, a bus for connecting the slave processors and the master processor to a common memory, said common memory including:

(a) a plurality of message buffers for storing messages in which each message includes a label which identifies a destination slave processor,

(b) an input area associated with each slave processor such that information is supplied to an input area only from its associated slave processor, and each area occupies the minimum space required to identify one of said message buffers,

(c) an output queue associated with each slave processor such that the contents of an output queue are only read by its associated processor,

the system arranged to transfer information from a source slave processor to a destination slave processor, in which:

(e) a source slave processor supplies a message to an addressed message buffer and either prior or subsequent to supplying said message said first processor loads to its associated area a message buffer indicator which indicates the position of said addressed buffer,

(f) the master processor scans all the input areas to detect a message buffer indicator, reads the label stored within an indicated message buffer, and supplies at least part of the message to the output buffer associated with the destination processor, and

(g) each slave processor scans and reads messages from its associated output queue.

In a preferred arrangement a slave processor must request use of the bus from the master processor before said slave may access the common memory. The master processor may issue a bus acknowledge signal to a highest priority slave processor after receiving a bus request signal from any of the slave processors if the master does not itself require to access the common memory. Preferably the highest priority slave processor will relay a bus acknowledge signal to the next priority slave processor if said highest priority slave does not require to access the common memory and so on until all the slaves have received a bus acknowledge signal.

Preferably, after reading a message the master processor replaces the label which identifies the destination processor with a label which identifies the source processor before writing the whole of said message to the associated output queue of the destination processor. Each message may include a chain pointer which may identify a further message which is to be supplied to the associated output queue of the destination processor.

In a preferred arrangement the common memory also includes a plurality of data buffers into which data is supplied by a source slave processor and ownership of the data buffer is transferred to a destination slave processor. Preferably each message also includes a data pointer which may identify a data buffer to a destination processor such that on receiving said message said destination processor takes over the ownership and has sole use of said data buffer.

In a preferred arrangement messages are written into an output queue in a cyclic manner and the master processor records the position where the last write occurred to each of said queues. Preferably each slave erases information from its associated output queue thus leaving the positions blank once said data has been read by said slave. A slave may then stop reading from its associated output queue after detecting one blank. Preferably a slave may scan information in its output queue without erasing said information in an attempt to locate a specific message and on detecting said message said slave will return to the unattended information in said queue. A slave may also periodically read a predetermined location in its associated output queue out of sequence and on detecting a crash command at said predetermined location said slave processor will attempt to retain syn-

chronisation with the master processor.

Preferably the input areas consist of single consecutive memory locations and the maximum number of message buffers is determined by the word length of said locations.

The invention will now be described by way of example only with reference to the accompanying Figures of which:

Figure 1 is a schematic representation of a multiprocessor testing apparatus for testing a packet switching network;

Figure 2 is a partial memory map for a common memory which forms part of the multiprocessor testing apparatus; and

Figure 3 is a circuit for allowing each processor of the multiprocessor system to obtain access to the common memory.

A packet switching network 1 is shown schematically in Fig. 1 having four synchronous terminals 2 and three asynchronous terminals 3. Information transferred over the asynchronous terminals must be interfaced to the packet switching environment by means of packet assembler/dissemblers shown generally by 4. Each synchronous and asynchronous terminal is respectively connected to a synchronous port 5 or an asynchronous port 6 of a multiprocessor testing system 7. The testing system includes a VDU/keyboard terminal 8 and a disc drive 9 allowing test programs to be loaded which generate a plurality of calls over a wide range of speeds and record the response of the network 1 to said calls.

The multiprocessor testing system 7 consists of twelve slave processors 10 (each based on a Zilog Z80 microprocessor) and a similar master processor 11. Ports 2 and 3 are grouped into pairs and each pair is interfaced to the system 7 by one of the slave processors 10. Of the remaining four processors one controls the operation of the VDU and keyboard 8, one implements CCITT X25 level 2 protocol, a third formats information for displaying on the VDU, and a fourth provides X25 level 3 and organises network tests. The master processor 11 controls the operation of the disc drive 9 and performs a central role with respect to the movement of information within the multiprocessor system.

The master processor 11 and each slave processor 10 may access 64K memory positions. Half of this addressable memory takes the form of local memory which is unique to each processor. The remaining 32K is used to address common memory 12 located on a common services board 13. Each processor 10,11 has access to the common memory 12 over a common bus 14—the bus is essentially controlled by the master 11 and each slave 10 must request use of the bus 14. The common services board 13 also includes a clock 15 (which generates timing signals for all of the processors 10,11 in the multiprocessor system 7) and may include another Z80 microprocessor for organising the functions it is required to perform.

The local memory of each slave processor 10 and the local memory of the master processor 11 consists of 2K of ROM and 30K of RAM. Application programs are loaded into the RAM from the disc 9 while the instructions held by the ROM control the transfer of information within the system 7. The multiprocessing system 7 may therefore operate in accordance with the present invention before application programs are loaded to the processors. The software of each processor uses the common memory for transferring information by arranging said common memory into a plurality of blocks, however, these blocks do not determine the physical arrangement of the common memory which is 32K of standard RAM.

A partial memory map for the common memory is shown in Fig. 2. The memory has twelve one-byte input areas 20, a plurality of eight-byte message buffers 21, twelve sixty-four-byte output queues 22 and a 255 data buffers 23.

Each of the twelve output queues 22 is uniquely associated with one of the slave processors 10. Information is supplied to a slave processor 10 by loading said information to its associated output queue. However information can only be loaded into an output queue by the master processor 11. The master processor knows the location in each output queue 22 where it last loaded data. The next load therefore immediately follows on in a cyclic fashion.

The master 11 has control over the common bus 14 and slave processors 10 must request bus access. Any of the slave processors 10 may issue a bus request at any time which is ignored by the master if the master 11 is itself accessing the common memory. When the master 11 no longer requires the common memory it issues a bus acknowledge signal which is supplied to a slave processor 10 having the highest priority. This slave processor 10 will then access the memory or pass the bus acknowledge signal to the next slave processor and so on. The arrangement is known as a daisy chain and the circuit for performing this operation is described below with reference to Fig. 3.

When a slave processor 10 has control of the bus said slave will read any information waiting in its associated output queue 22. Like the master processor stores the position in each output queue where it last implemented a write so each slave processor knows the last byte to be read. After reading information from a memory location in its output queue a slave processor erases that information, ie it loads blanks. On implementing a read to its output queue a slave processor will read information until it encounters a blank. Therefore if no information is waiting in an

output queue the associated slave processor will only read one byte and then pass a bus acknowledge signal to the next slave processor in the daisy chain.

The above details how information is transferred from the master 11 to a slave 10. This procedure is also followed when a slave communicates with another slave which ensures the master has control over all information transfers. If a slave 10 is required to send information to the master 11 or to another slave it does so using one of the message buffers 21. The master 11 determines which message buffers 21 are available for each slave and more message buffers 21 may be requested by a slave during operation of the system 7. If a first slave (a source) wishes to send a message to a second slave (a destination) then the source slave processor 10 requests use of the common bus 14. When the source slave receives a bus acknowledgement signal it writes an 8-byte message into a message buffer 21, and then writes a message buffer indicator to the area 20 associated with the source processor. The source processor only has one associated area which ensures that control of the bus will be passed to another processor.

Each 8-byte message is generated in accordance with a defined structure. The first byte is a chain pointer which allows message buffers to be chained thus maintaining queues of messages. The maximum number of messages which may be chained is restricted to the number of message buffers 21 which are available to a slave processor. Therefore each slave will have to pass control of the bus to the master 11 to ensure that said slave has free buffer space.

Byte 2 is a data pointer which uniquely defines one of the 255 64-byte data buffers 23. The bulk of any information which is to be sent from a source slave processor 10 is placed in a data buffer 23, for example a packet to be sent or a packet received from line. In this embodiment each data buffer has 64 bytes but essentially they may be of any suitable length. When a destination processor receives a data pointer it takes over the ownership of the data buffer identified by the pointer and may then access the information contained within said buffer. Therefore the bulk of information transferred between processors does not have to be physically transferred from one memory location to another memory location.

Byte 3 is a message destination label which identifies the destination slave processor 10. On reading this byte the master processor knows which output queue 22 the message is to be written to, ie the one associated with the destination processor. However before the message is written to an output queue 22 the destination label is replaced with a source label which identifies the source processor to the destination processor.

The fourth byte describes the message function, for example, take over the data buffer 23 identified by the data pointer (byte 2), or send out a test frame etc. Bytes 5 to 8 are message parameters the meaning of which is determined by the message function.

Any of the slave procesors 10 may also be provided with the facility of looking ahead in their output queues 22. If they are waiting for a specific message, for example the answer to a question, then they may scan and ignore several messages until the awaited message is found. The message is then acted upon, a market left to the effect that this message must not be read again, and the slave processor returns to the start of the previously ignored messages. Of course a slave 10 cannot look further than the 64 bytes of the cyclic output queue 22 and messages are not erased while they are being scanned.

In addition to organising the transfer of information, as described above, the master processor 11 must also ensure that all the slaves are operating in synchronism. On detecting that something has gone wrong with a slave processor the master loads a one byte crash command to a predetermined position 24 in the slaves' associated output queue. Each slave is arranged to periodically scan its respective predetermined position and on finding a crash command will set about regaining synchronisation with the rest of the system while temporarily halting its external activities.

A slave processor 10 calls for memory access on (to be precise just after) the rising edge of a clock pulse and the common services board 13 provides a bus acknowledgement signal on the falling edge. Since by the time the bus acknowledgement signal is available all processors that require a common memory access will have asserted their requirement, this allows contention for use of the bus to be solved by the daisy chain arrangement.

The Bus control logic for each slave processor 10 is shown in Fig. 3 and is designed to request use of the common memory bus only when necessary (ie the processor is attempting to access the common memory) and to hold the processor frozen until the bus becomes available. When a slave processor sets its address line A15 high, indicating a requirement for a memory access to the common memory, the output of gate 30 will be forced low. When the memory request signal from the slaves' CPU goes active (low), half a clock cycle later, the output of gate a 31 will be allowed to go high which will clock a zero into a D-type bistable 32. The output of gate 31 being high will automatically force the output of a gate 33 high, thereby making this processor's bus grant out signal false which will inhibit any processors of lower priority from accessing the bus. When "Bus grant in" goes

low (and this will depend on the common services board being ready for an access at the same time as no other processors upstream requiring access) bistable 33 is preset. This transition will clock a zero into a latch 34 (since the output from gate 31 and hence the preset input is high) which means that common bus enable becomes true. Note that during the period when bistable 33 has its Q output low the CPU is held in a wait condition. When released the processor will operate the necessary signals to achieve the required memory access and when finished the memory request signal will go high again which forces the output from gate 31 low, presetting latch 34 and re-enabling the bus-grant daisy chain via gate 33. A little later the A15 line from the CPU will go low again and the bus is released for the next cycle.

CLAIMS

1. A multiprocessor system, comprising a master processor, a plurality of independently controlled slave processors, a bus for connecting the slave processors and the master processor to a common memory, said common memory including:

(a) a plurality of message buffers for storing messages in which each message includes a label which identifies a destination slave processor,

(b) an input area associated with each slave processor such that information is supplied to an input area only from its associated slave processor, and each area occupies the minimum space required to identify one of said message buffers,

(c) an output queue associated with each slave processor such that the contents of an output queue are only read by its associated processor,

the system arranged to transfer information from a source slave processor to a destination slave processor, in which:

(e) a source slave processor supplies a message to an addressed message buffer and either prior or subsequent to supplying said message said source processor loads to its associated area a message buffer indicator which indicates the position of said addressed buffer,

(f) the master processor scans all the input areas to detect a message buffer indicator, reads the label stored within an indicated message buffer, and supplies at least part of the message to the output buffer associated with the destination processor, and

(g) each slave processor scans and reads messages from its associated output queue.

2. A multiprocessor system according to claim 1 in which a slave processor must request use of the bus from the master processor before said slave may access the common memory.

3. A multiprocessor system according to claim 2 in which the master processor issues a bus acknowledge signal to a highest priority slave processor after receiving a bus request signal from any of the slave processors if the master does not itself require to access the common memory.

4. A multiprocessor system according to claim 3 in which the highest priority slave processor will relay a bus acknowledge signal to the next priority slave processor if said highest priority slave does not require to access the common memory and so on until all the slaves have received a bus acknoledge signal.

5. A multiprocessor system according to any of claims 1 to 4 in which after reading a message the master processor replaces the label which identifies the destination processor with a label which identifies the source processor before writing the whole of said message to the associated output queue of the destination processor.

6. A multiprocessor system according to any of claims 1 to 5 in which each message includes a chain pointer which may identify a further message which is to be supplied to the associated output queue of the destination processor.

7. A multiprocessor system according to any of claims 1 to 6 in which the common memory also includes a plurality of data buffers into which data is supplied by a source slave processor and ownership of the data buffer is transferred to a destination slave processor.

8. A multiprocessor system according to claim 7 in which each message includes a data pointer which may identify a data buffer to a destination processor such that on receiving said message said destination processor takes over the ownership and has sole use of said data buffer.

9. A multiprocessor system according to any of claims 1 to 8 in which messages are written into an output queue in a cyclic manner and the master processor records the position where the last write occurred to each of said queues.

10. A multiprocessor system according to claim 9 in which each slave erases information from its associated output queue thus leaving the positions blank once said data has been read by said slave.

11. A multiprocessor system according to claim 10 in which a slave will stop reading from its associated output queue after detecting one blank.

12. A multiprocessor system according to claim 10 or claim 11 in which a slave may scan information in its output queue without erasing said information in an attempt to

locate a specific message and on detecting said message said slave will return to the unattended information in said queue.

13. A multiprocessor system according to any of claims 9 to 12 in which each slave processor periodically reads a predetermined location in its associated output queue out of sequence and on detecting a crash command at said predetermined location said slave processor will attempt to retain synchronisation with the master processor.

14. A multiprocessor system according to any of claims 1 to 13 in which the input areas consist of single consecutive memory locations and the maximum number of message buffers is determined by the word length of said locations.

15. A multiprocessor system as described herein with reference to Figs. 1 to 3.

16. A packet switching network testing system as described herein with reference to Figs. 1 to 3.

Printed in the United Kingdom for
Her Majesty's Stationery Office, Dd 8818935, 1985, 4235.
Published at The Patent Office, 25 Southampton Buildings,
London, WC2A 1AY, from which copies may be obtained.